Dual Eye-Tracking Methods for the Study of Remote Collaborative Problem Solving

THÈSE N^{O} 5232 (2011)

PRÉSENTÉE LE 21 DÉCEMBRE 2011

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

CRAFT - GESTION

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Marc-Antoine NÜSSLI

acceptée sur proposition du jury:

Prof. P. Fua, président du jury
Prof. P. Dillenbourg, Dr P. Jermann, directeurs de thèse
Prof. A. Billard, rapporteur
Prof. R. Dale, rapporteur
Prof. A. Duchowsky, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2011

On fait la science avec des faits, comme on fait une maison avec des pierres: mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison.

— Henri Poincaré

Acknowledgements

First of all, all my gratitude goes to both my supervisors, Pierre Dillenbourg and Patrick Jermann. Pierre has offered me the chance to conduct this research and has supervised me all along this complex project; he gave me the freedom and his trust to explore my own ways. Patrick was an invaluable companion that helped and supported me through the long and difficult journey that this thesis represents; without him, this work would have never reached this maturity.

Second, I would like to thank wholeheartedly former colleagues that participated closely to some aspects of this work. In particular, a special thank to Mirweis Sangin for all passionate debates and insightful advices as well as for his direct contribution to this work. A great thanks to Mauro Cherubini, Gaëlle Molinari and Weifeng Li for their close and valuable collaboration in this project.

Many thanks to all my colleagues and friends at CRAFT, who contributed more or less directly to this work through their advices and support: Frédéric Kaplan, David Bréchet, Olivier Guédat, Guillaume Zufferey, Quentin Bonnard, Andrea Mazzei, Hamed Seiied Alavi, Sébastien Cuendet and Son Do Lenh Hung. A particular thank to Florence Colomb for her constant availability and her willingness to help.

Also, I would like to thanks colleagues over the world who contributed to this work through insightful discussion and constructive comments: Halszka Jarodzka, Ignace Hooge, Marcus Nyström, Kenneth Holmqvist, Yvonne Kammerer and Prof. Daniel Richardson and many others.

I would like also to thank Amanda Harrington Legge for her help in the proofreading of parts of the manuscript.

Finally, I would like to thank my family and friends for their support, love and friendship during these years of intensive work and last but not least, a big thank you to my love, Christelle, who transformed my life and gave me the strength to finish this work.

This thesis was funded by the Swiss National Science Foundation (grant #K-12K1-117909).

Lausanne, 28 September 2011

Abstract

Applied eye-tracking has been extensively used for the study of psychological processes. More recently, some researchers have used this technique to study the interaction between people by tracking and analyzing eye-movements of two persons synchronously. However, this is generally accomplished by observing people in simple controlled settings. In this thesis, we use a similar methodology, dual eye-tracking, to study people in more natural, semantically richer, tasks with the aim of identifying dual eye-movement patterns which reflect collaborative processes. Eye-tracking, and more globally eye-movements analyses, is a complex domain involving several methodological issues, which have not yet been satisfactorily solved. This work is also an attempt to offer solutions to several of these issues.

The first part of this thesis is dedicated to **the improvement of the methodology of eye-tracking data analysis**. We present several developments pertaining to the general methodology of eye-tracking. More specifically, we identify problems and offer solutions to the following aspects: fixation identification, systematic position errors correction and hit detection. We also tackle more specific questions concerning the method dual eye-tracking. We present issues that arise in dual eye-tracking data collection and analysis and propose some solutions. On the technical side, we deal with the question of the synchronous recording of two streams of eye-movements and on the analytical side, we extend gaze cross-recurrence, a measure of eye-movements coupling, to complex realistic collaborative tasks.

The second part is devoted to **experimental studies of collaboration through the use of dual eye-tracking methods**. We first present four exploratory studies which allowed us to set up the stage by identifying interesting phenomena and experimental difficulties. The main results of these experiments revolve around the relationship between gaze and speech. In these respects, we extended some results found in the literature to more natural settings and we developed a computational model to make actual predictions about dialogue and visual references. Finally, the main study of this thesis is about computer program understanding. This study is composed of two experiments: a solo programming experiment, from which we identified gaze patterns of a single programmer and a pair-programming task in which we explored how gaze patterns during program comprehension are affected by collaboration.

Keywords: Eye-tracking methodology, Fixation identification, Applied eye-tracking, dual eye-tracking, Collaboration, CSCW

Résumé

L'oculométrie (eye-tracking) appliquée a été utilisée de manière intensive pour l'étude des processus psychologiques. Plus récemment, certains chercheurs ont utilisé cette technique pour étudier l'interaction entre des personnes en mesurant et en analysant les mouvements oculaires de deux personnes. Toutefois, cela a généralement été fait dans le cadre de tâches simples et contrôlées. Dans cette thèse, nous utilisons une méthode similaire, l'oculométrie double (dual eye-tracking), pour étudier des personnes dans des tâches plus naturelles et sémantiquement plus riches. L'oculométrie, et plus généralement l'analyse des mouvements oculaires, est un domaine complexe impliquant plusieurs questions méthodologiques qui n'ont toujours pas été résolues de manière satisfaisante. Ce travail est également une tentative pour offrir des solutions à plusieurs de ces questions.

La première partie de cette thèse est dédiée à **l'amélioration de la méthodologie de l'analyse de données oculométriques**. Nous présentons plusieurs développements concernant la méthodologie générale de l'oculométrie. Plus spécifiquement, nous identifions des problèmes et proposons des solutions sur les aspects suivants : identification des fixations, correction des erreurs systématiques de positions et détection de correspondances. Nous nous attaquons également à la méthodologie spécifique de l'oculométrie double. Nous présentons les questions qui apparaissent dans le contexte de la collecte et l'analyse de données oculométriques double et nous proposons des solutions. Sur le plan technique, nous traitons de la question de l'enregistrement synchronisé de deux flux de données de mouvements oculaires et sur le plan analytique, nous étendons la récurrence croisée des regards (gaze cross-recurrence), une mesure du couplage de mouvements oculaires, à des tâches réaliste et complexes.

La seconde partie est dévouée à **des études expérimentales de la collaboration à travers l'utilisation de méthodes d'oculométrie** double. Nous présentons d'abord quatre expériences exploratoires qui nous ont permis d'identifier des phénomènes intéressants et des difficultés expérimentales. Les résultats principaux tournent autour de la relation entre le regard et la parole. A cet égard, nous avons étendu des résultats existants dans la littérature à des conditions plus naturels et nous avons aussi développé un modèle informatique pour faire des prédictions sur le dialogue et les références visuels. Finalement, l'étude principale de cette thèse est sur une tâche de compréhension de programme informatique. Cette étude est composée de deux expériences : une expérience individuelle, à partir de laquelle nous avons identifié des motifs de regards d'un programmeur seul, et une expérience de programmation en pair, dans laquelle nous avons exploré de quelle manière la collaboration affectait les motifs de mouvements oculaires.

Mots-clés : Méthodologie de l'oculométrie, Identification de fixations, Oculométrie appliquée, Oculométrie double, Collaboration, CSCW

Contents

1 Introduction

Several researchers argue that cognition is distributed between a person and its environment, including its social environment and we often speak of joint cognition to refer to several people that interact so closely that they start to form a single, distributed, cognitive system. In the current work, we aim to make a step in the study of such a distributed cognitive system that arises when people collaborate to perform some joint activity. More specifically, we explore how socio-cognitive processes underlying joint cognition are reflected in the eye-movements of the collaborators. We are interested in finding patterns in collaborators' eye-movements that are related to the type or the quality of interaction. Incidentally, we explore whether, and how, eye-movements reflect collaborative processes which people are engaged in. While previous studies show that eye-movements reflect cognitive processes to some extent, this is certainly not the case for all cognitive activities (for example, thinking about abstract mathematical concepts) and thus, it is legitimate to ask whether collaborative processes may also be revealed through eye-movements. Previous works have already shown that, to some extent, eye-movements of people interacting verbally inform about the level of mutual understanding, or grounding (Richardson and Dale, 2005; Richardson et al., 2007). In the current study we question first whether such results can be extended to more ecologically valid situations and secondly whether we can find other relationships of this kind.

The general goal of this study is twofold. On the one hand, we want to gain insight about how cognition, and more specifically social cognition, works. Indeed, by finding relationships between collaborative processes and eye-movements, we could understand better the mechanisms behind social cognition. This is of a general interest for cognitive sciences as it could help to build better models of cognition. In this respect, we are interested in interpreting the identified gaze patterns in terms of cognitive processes. On the other hand, this work also has an applied aspect, which is to make actual predictions about the collaboration processes from the partners' gazes. Such predictions could then be used by computer-supported collaborative tools to improve the interaction between collaborators. Indeed, considering the rapid development of eye-tracking techniques, it is likely that within the next decade, it would be possible to use any computer equipped with a simple web-cam as an eye-tracker. Hence, it is reasonable

to start developing applications that would take advantage of eye-movement data in order to enhance the user experience. Contrary to other approaches such as the COGAIN project[1] for disabled people in which eye-tracking is used as a control device (for example, to replace or extend mouse cursor control), our idea is to take advantage of natural eye-movements. By being able to identify patterns in eye-movements that predict collaboration states, we could have tools that adapt to the users. Typically, we refer to awareness tools or groups mirrors which aim to reflect the state of collaboration or the state of the partner in order to enhance the collaborative activity.

Finally, the relationship between gazes and collaboration is investigated at different levels of temporal granularity. At one extreme, there is the macroscopic level, which encompasses a whole interaction. We ask whether it is possible to predict the general quality of the interaction from the complete gaze data of both partners. At that level, we compute some global indicators summarizing the whole gaze data, generally disregarding the temporal aspect. At the other extreme, there is the microscopic level, which is a single interaction episode such as the understanding of a specific utterance or group of utterances. At such a low level, we examine the precise temporal flow of eye-movements to see how it is related to utterance understanding or utterance elaboration level, for example.

1.1 Eye-tracking methodology

An important part of this work deals with the possibility of measuring eye-movements in a reliable way. Whether we want to build theories about cognition or whether we want to make actual predictions to be used in computer systems, it is of primary importance to have reliable eye-movement measurements that are independent of the eye-tracking system and of the subject or situation. Indeed, measuring eye-movements in a reliable way is a difficult process because it is generally based on indirect measurements. More specifically, most current eye-trackers are video-based systems that use the corneal reflection of a focused light, typically an infra-red LED. Such systems are affected by physiological aspects (e.g. eye dryness) which can affect the corneal reflection phenomenon and by technical aspects (e.g. glasses, luminosity) that can hinder computer vision algorithms used to infer the gaze position from video images. These are sources of noise, inaccuracies, and even risks of loss of data that need to be taken into account. Finally, as we will see, there are multiple preprocessing steps that need to be performed between the raw data provided by the eye-tracker and the statistical analysis relating eye-movements to cognitive processes. It is necessary to take into account the various sources of errors cited above to perform this preprocessing in an adequate way. In addition to the fact that these issues involve complex technical problems, eye-tracker manufacturers generally partially ignore these problems and claim that their systems provide data close to perfection. For these reasons, researchers are often led to ignore these facts which hampers the collection of reliable data and, consequently, of sound analyses. We set out to bridge this methodological gap in this thesis. In chapter 3, we address several methodological issues with

[1] COGAIN is a network of excellence on Communication by Gaze Interaction (see http://www.cogain.org/).

respect to gaze data analysis in order to draw stronger conclusions about our primary focus of interest: the discovery of dual gaze patterns reflecting collaborative activities.

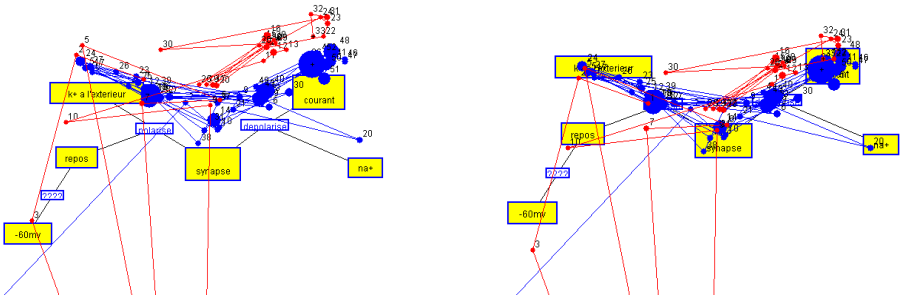
1.1.1 Fixation identification

One of the most important methodological issues that has to be addressed is the identification of fixations, and possibly of saccades[2], from the raw data provided by the eye-tracker. Indeed, eye-movements form specific patterns that exhibit two main dynamics, fixations and saccades, while eye-tracker systems simply measure the position of the gaze at specific moments in time. Thus, we have to identify the corresponding eye-movements from these raw measurements. Salvucci and Goldberg (2000) have summarized the main algorithms that have been developed by researchers to accomplish this task; however, there is no commonly agreed solution to this question and this is still an open problem. Moreover, there is no good, objective way to assess whether one algorithm performs well or not and the evaluation of the results is often left to subjective interpretation. This is an important issue as it hinders reliable comparisons between different studies and therefore, replication of results. What makes fixation identification difficult is that there are different types of algorithms. Furthermore, each of these algorithms depends on several parameters. Selecting the best algorithm and parameters greatly depends on the characteristics of the raw gaze data, such as its sampling frequency or noise level, which is in turn influenced by the eye-tracking device but also, to some extent, by other factors related to the experimental condition or the subject. The algorithm and parameters selected may have a large influence on subsequent analyses. Blignaut (2009) has shown how changes in the parameters can dramatically alter the result of the identification and Shic et al. (2008a) have shown that it can even invert the conclusions drawn from a subsequent statistical analysis done on the identified fixations. In section 3.2, we present several analyses and developments related to this topic. First, in line with the works of Blignaut and Beelders (2009) and Blignaut (2009), we explore the influence of these algorithms and parameters on the fixation identification more deeply and we also identify external factors that may affect this process. Finally, following the work of Nyström and Holmqvist (2010), we develop a new adaptive algorithm that takes into account variations of noise level and possibly of sampling frequency. The originality of our approach resides in the fact that we develop an objective measure of fixation identification quality and use it to optimize our adaptive algorithm.

1.1.2 Systematic errors correction

The second methodological point that we address concerns the accuracy of the measured positions. Indeed, qualitative analyses reveal that the measured locations are often shifted from the actual gaze position. This fact has been confirmed by other researchers (Hornof and Halverson, 2002). In the case of natural images, it may be hard to determine whether measured gaze locations are correct or not, but, in the case of artificial stimuli, such as computer software

[2] As we will see, saccades identification is often considered to be impossible with low temporal resolution eye-trackers, such as the one used in the present work.



(a) Raw fixations (b) Corrected fixations

Figure 1.1: Fixations of two subjects, red and blue, as measured by the eye-tracker and after correction of systematic accuracy errors

or texts, it is often very clear that there are errors in the measured locations. Indeed, such stimuli frequently consist of a few areas of interest disposed over a blank region and thus, if we assume that people look at the stimulus in an intentional way, it is very likely that most of their gazes fall on the objects and not in the blank regions. Hence, when we see measured gazes such as in figure 1.1a, it is clear that there is an upward shifting of their actual fixations and it seems obvious that the real fixations should instead be positioned lower like in figure 1.1b. It is very interesting to note that there is clearly some systematic trend in these accuracy errors. More specifically, these errors are not necessarily systematic across recordings, but, within a specific experimental trial, it appears that there are generally the same. The example presented in figure 1.1 demonstrates the importance of correcting such errors for the rest of the analyses. Indeed, we generally want to relate each fixation to objects of the stimulus and thus, if we take the uncorrected data, a lot of fixations will be incorrectly mapped to the wrong objects. It is important to see that it is not simply a matter of being more tolerant and allowing fixations to fall further from the objects because in the uncorrected data, some fixations are closer to another object than to the actual fixated object. In section 3.3, we present an automatic systematic errors correction algorithm that we developed to solve this important issue. Our approach follows the same line of thought as Hornof and Halverson (2002), postulating that we generally look at the objects and not at the blank regions in order to estimate the best correction to apply. We also explore how much confidence we can have in such an estimation and also whether these systematic errors may vary over time and possibly over space. Finally, we propose two variations of our algorithm, one optimal offline method that can be used when analyzing data a posteriori and a probabilistic optimization method that is fast enough to be used in real-time applications.

1.1.3 Hit detection

Finally, we also address hit detection concerns the question of how to map gaze data to stimulus objects. In order to perform semantically meaningful analyses of eye-movements, it

is necessary to relate each fixation to one or several objects of the stimulus, also called Area Of Interest (AOI). Usually, researchers simply assign the AOI on which the fixation falls, or the closest AOI if the fixation is in a blank part. While this seems to be a reasonable way to do it, it may be quite arbitrary in certain situations. Depending on the size, number, and arrangement of the AOIs, it is difficult to assign each fixation to a single object unambiguously. This is the case, for example, when the distance between the AOIs is close to the precision of the eye-tracker. In such situations, fixations falling between AOIs or on the edge of an AOI could actually be on either object. Another problem with the classical method is that each fixation is always linked to a single object while it is clear from the literature that we can actually see more than a single object with only one fixation. In section 3.4, we address these issues by proposing a probabilistic model which assigns fixation probabilities to several objects for each fixation. Through such a model, we intend to account both for the imprecision of the eye-tracking device and for the fact that one fixation may actually be used to get information from more than one object. We also discuss possible biases, due, for example, to object sizes, which can arise during hit detection.

1.2 Dual eye-tracking methodology

The study of eye-movements in a collaborative context is a novel approach that has almost never been explored, except for a few exceptions such as Richardson and Dale (2005) or Pietinen et al. (2008). For this reason, we think that it is important to discuss and explore the technical requirements for such studies. Moreover, the analysis of dual eye-movements, i.e. the gazes of two collaborators, in complex ecologically valid tasks, raises challenges that have to be addressed. Chapter 4 of this thesis is devoted to these topics.

1.2.1 Technical issues

Most eye-tracking studies focus on the eye-movements of a single person while she performs certain tasks. We, on the other hand, study pairs of people collaborating on problems, which implies considerations at the technical level, as eye-tracking devices are generally not designed to be used in such situations. Indeed, it is necessary to have a system able to measure the eye-movements of two people synchronously with high temporal precision. Moreover, these synchronous recordings must work in conjunction with a shared application that logs any changes or user actions with the same time reference as the gaze data. Last but not least, as we are studying collaboration, we also need to record speech from both participants again using a synchronous time base. These specificities prevent the use of the basic software provided by eye-tracker manufacturers and force the design of a new setup using low-level access to eye-tracking devices, which allows for precise control of the exact timing of gazes. In section 4.2, we present such a framework based on a centralized server able to perform time synchronization of the different data streams, gazes, speech, and actions at a high precision level. While it is mainly designed to be used with the specific eye-tracker we use, namely the

Tobii 1750, it provides insight for use in other contexts.

1.2.2 Dual gaze analysis

Analyzing eye-movements of two people collaborating also raises questions about how to characterize these dual eye-movements in order to extract meaningful information that could be related to collaboration. We review and explore different indicators that could capture some aspects of dual-gaze patterns. More specifically, we focus on gaze cross-recurrence analysis, a particular type of indicator originally used by Richardson and Dale (2005). This analysis aims to measure the coupling between streams of gaze data and has already been shown to be related to interaction processes between two people and particularly to mutual understanding. However, it has been used mainly in simple situations with simple static stimuli and it appears that its use in more realistic situations requires a lot of care in its interpretation in terms of extracting psychologically meaningful information. We discuss these issues in section 4.3 and we propose a method based on theoretical considerations to overcome problems that arise.

1.3 Gaze and collaboration

The various methodological issues presented above form the basis for the main goal of this thesis, which is to use eye-movement analysis to study socio-cognitive processes underlying collaboration. This is a very broad question and we focus more specifically on computer-supported collaboration (CSCL[3] or CSCW[4]) between two people in a remote situation. In chapter 5, we present studies on different types of tasks ranging from low-level sensory-motor coordination, such as a dual player Tetris game, to high-level cognitive activities, such as collaborative learning through the building of a concept map. Our main experiment, which is presented in chapter 6, is about pair-programming[5] and thus is very close to real-world activities performed on a daily basis in certain companies. Through the analysis of these different experiments, we highlight different phenomena that occur during collaboration and we also outline the design of specific tools that could take advantage of real-time gaze monitoring to enhance collaborative activities.

Concretely, we search for indicators of gaze patterns that are predictive of the nature and quality of collaboration. We classify these indicators according mainly to two criteria: whether they are single or dual indicators, that is whether they summarize gazes of one individual or of both individuals; and whether they are task-independent or not, that is whether their computation requires knowledge of the stimulus content or not. Of course, the best result we can expect is to find task-independent indicators that universally reflect collaboration independently of the task type. However it is not clear whether such indicators exist because the type and the

[3] CSCL stands for Computer-supported collaborative learning

[4] CSCL stands for Computer-supported collaborative work

[5] Pair programming is a software development technique in which two programmers work together at one workstation.

structure of the visual stimulus drive at least to some extent, the eye-movements that happen on them. For this reason, we will generally concentrate on task-dependent indicators which appear to be much more meaningful. Finally, we study gaze patterns and their relationship with collaborative processes at different levels of granularity. These levels range from macro-level, i.e. a whole interaction of several minutes, to micro-level, i.e. a single utterance, with intermediate levels represented by interaction episodes lasting between a few seconds to one minute.

1.3.1 Collaboration quality

At the macro-level, one main question of interest is about assessing the quality of interaction through dual-eye-movements. Is it possible to know whether a collaboration was successful or not by looking at the gazes of the collaborators? Assessing collaboration quality is a complex and open question for the simple reason that there is no clear definition of what good and bad collaboration is. Of course, there are several aspects that we generally consider to be related to a good collaboration, such as grounding or elaborated argumentation, but, in general, there are so many factors that play a role that we cannot really define good collaboration in an objective way. Moreover, the context of collaboration is also of a great importance. We do not expect the same kind of collaboration between a very good student and a bad student (in which case the good one might lead the interaction) as between two students of the same level (in which case we expect a more balanced interaction).

It is common, when speaking of interaction quality to distinguish between the process and the results. The process refers to the success of the interaction, i.e. whether the peers understood each other well or whether they were able to share the work in an efficient way. The result refers to the actual output of the collaboration, which is, for example, whether they solve the task well or not, or in the case of collaborative learning, whether they learned well or not. Of course, it is clear that these two aspects are interrelated and it is unlikely to a have a very good interaction process yielding a very bad result. However, other factors, such as collaborators' expertise, explain the quality of the result and thus, the relation is clearly not deterministic: it is possible to have a good interaction produce worst results than another bad interaction. Typically in the case of collaborative problem solving, two experts of the domain who collaborate badly can still have better results than two novices who collaborate well. However, the result is the only aspect that we can measure in an objective way. For this reason, at the macro-level, the time of a whole interaction, we mainly try to predict the result of the collaboration. We address the question of interaction process quality at lower levels by studying, for example, the comprehension of a single utterance or the quality of a short episode. In section 5.3, we present results about the prediction of the collaboration product quality from collaborators' eye-movements and in section 6.4.3, we briefly explore the possibility of detecting the collaboration process quality.

1.3.2 Expertise and group compositions

In computer-supported collaborative learning, it has been shown that the interaction could be enhanced if the partners are aware of their relative level of knowledge because it allows a faster and better grounding between them. Indeed, by being aware of their differences in knowledge, collaborators could, on the one hand, adapt their discourse to their peer and, on the other hand, better estimate the correctness of what was said by their partner Sangin (2009). This motivates the development of knowledge awareness tools which can provide such information to the collaborators. The problem with the building of awareness tools is that it requires some way to determine the respective knowledge of partners. This can be done through an a priori evaluation phase, but this greatly limits the use of such tools in real-life applications. By being able to automatically detect in real time the relative level of knowledge of the collaborators through the analysis of behavioral data, such as eye-movement data, it could be possible to automatically provide such information to the peers. In this respect, we explore, in sections 5.4 and 6.4.2, the possibility of detecting such a difference in knowledge from the eye-movements of the collaborators.

1.3.3 Interaction episodes

Collaboration can generally be decomposed into different types of episodes consisting of one or multiple utterances which represent a certain kind of interaction. These episode types are defined according to certain criteria about the kind of utterances composing them. It is possible to classify episodes according to the semantic content of the utterances. For example, we can distinguish between meta-cognitive episodes in which partners discuss about how they solve problems or about what they learn, and domain-content episodes in which they discuss about the content of the problem under consideration. Another way to define episodes is to consider the kind of exchange that is done. For example, there are argumentation episodes during which collaborators try to make their point by presenting their arguments or there are explanation episodes in which collaborators give detailed explanations to each other. These episodes are generally identified by first coding each utterance according to a coding scheme representing the criteria of interest and then by grouping these utterances according to their assigned code. Thus our goal is to find patterns in eye-movements which are representative of utterance codes. We present results of this kind in sections 5.5 and 6.4.2.

1.3.4 Explicit referencing at the utterance level

At the micro-level, we are especially interested in verbal references to visual objects, called explicit references, and how they are understood by the collaborators. Indeed, references to visual objects are an important aspect of collaboration. Contrary to co-located collaborative work where people can use other means of referencing, such as deixis or even the direction of gaze, in the case of remote computer-supported collaboration, explicit reference is a central concept, as it is often the only way to refer to the objects of interest. Thus, we are interested

in how the relationship between a speaker's gazes and a listener's gazes may actually reflect the comprehension of references to visual objects. Griffin and Bock (2000) have shown that a verbal reference to a visual element is generally preceded by gazes on the object of concern. Also, a similar effect has been found on the listener's side: a listener tends to look at the object of reference some time after hearing the reference (Allopenna et al., 1998). In section 5.6, we therefore explore whether these effects can effectively be used to make predictions on the comprehension of the references at the single reference level. We also extend these effects to text-based communication and show how they can be used to predict misunderstandings of utterances. Finally, we design an algorithm that uses this phenomenon to automatically discover relationships between words and visual objects through the monitoring of collaborator's gazes and their speech.

2 Background

Considering that we study the relationships between eye-movements and socio-cognitive processes underlying collaboration, it is necessary to have an overview of current knowledge in these fields. In the first section, we present a general introduction to basic concepts related to the study of collaboration and we show why collaboration is important for cognition in general. We also present practical applications of the study of collaboration. These aspects motivate the main topic of research of this thesis. The second section is devoted to general considerations about eye-movements and vision followed by a brief introduction about eye-tracking technology and methodology. This will introduce the tool and methodology, i.e. eye-tracking and eye-movements analyses, which are used in this work. Finally we present the main existing results about eye-movements in collaborative situations which form the basis of our study.

2.1 Collaboration

Study of collaboration is of a great importance for the understanding of cognition in general. Indeed, by the early nineties, a new current of thought emphasizing the importance of the context for cognition has emerged. This novel approach to cognition took different forms often related to different disciplines, such as situated cognition (Brown et al., 1989) in educational psychology, embodied cognition (Clark, 1998) in robotics and artificial intelligence, externalism (Noë, 2009) in philosophy or distributed cognition (Hutchins, 1995) in psychology. However they all revolve around the idea that in order to fully understand cognition, we should not consider it as a simple product of a single mind but rather as a complex system including the mind and its environment. These theories reject the simple computational view of the mind which tends to see cognition as based purely on internal representation. Instead, they postulate that representations are distributed between mind, body and environment. To simplify things, in this work, we group all these approaches under the term of distributed cognition.

2.1.1 Distributed cognition

One aspect of distributed cognition concerns physical artifacts that people use to enhance or support thinking. Tools such as pencil, whiteboards, calculators, meters are all artifacts over which cognition is distributed in the sense that they serve as aids that extends cognitive activity. For example, Hutchins (1995) has conducted an ethnographic study which shows how airplane pilots use their environment, i.e. the cockpit instrumentation, to offload their minds of useless information in order to have maximum resources to accomplish complex and dangerous tasks. He has shown how memorization and processing of airplane speed is not accomplished purely in the mind of the pilot. Rather, it is distributed among the pilot, the copilot, the two airspeed indicators and their bugs, the speed card booklet[1], the gross weight indicators and the speech exchange between the pilots and with the air guidance. Hutchins argues that it is not possible to give a correct account of how speed is processed during landing operations without considering the entire cockpit as a distributed cognitive system.

External representations such as drawings or texts are another kind of artifact that are used to support cognition. Zhang (1997) has shown that the way a tic-tac-toe game is represented drives the cognitive processes used to solve the game. Indeed, this game can have different isomorphic representations, such as the usual one based on a 3-by-3 matrix where players draw crosses and circles or, as a number game in which players draw numbers from 1 to 9 until they get 3 numbers that add up to 15. While these apparently different games have exactly the same abstract structure, performers act in a fully different way when they play to one or another version. They use cognitive operations that correspond well to what the specific game representation offers to them, thus taking advantage of the external representation to perform mental operations (e.g. spatial reasoning for tic-tac-toe vs numerical reasoning in the number game).

Actions performed in the environment can also be strategies that are used to support cognition. Kirsh and Maglio (1994) have demonstrated how people, in a Tetris game, use actions to avoid doing complex computations by themselves. In particular, they show that expert players do much more pieces rotations than what would be required to place the piece correctly. This allows them to avoid to do mental rotations by doing instead direct perceptual matching with each of the possible piece orientation. They also do additional, apparently useless, translations from the side of the board to avoid the need of a difficult and unreliable visual counting of horizontal position.

Last but not least, one of the most important aspects of our environment for cognition is certainly our social context. Through language, we can construct knowledge or concepts in group and this is perhaps in this way that cognition can reach its most developed state. Schwartz (1995) has shown that dyads were more able to induce an abstract mechanical rule than individuals. More precisely, he has shown that dyads succeeded more often than

[1] Speed card booklets are small numerical tables showing to the pilots important reference speeds necessary for landing operations.

what would be expected if we simply consider them as two individuals where only one who has to find the rule. Rather it appeared that dyads were able to induce the rule in cases in which both of the individuals taken alone would have failed. In another experiment, he has also shown that dyads were much more able to draw abstract representations of some problems than single individuals. Healey et al. (2000) did also an interesting experiment where dyads had to communicate about music solely by the mean of drawings. They showed that subjects tended to draw more abstract, and in this way more informative, representations of the music when they had more possibilities of interaction. More precisely, the amount of abstract drawings rose from 25% to 50% between a situation where they had no direct feedback between them and a situation which allowed them to give direct feedback to their partner. All these results tend to show that through social interaction people are able to build more abstract representations than they would do alone. This is of particular importance if we consider that abstraction capability is often considered as one of the key attribute of cognition (Hofstadter, 1995; Carpenter et al., 1990).

2.1.2 Language

Collaboration occurs mainly through the use of language and its role for cognition is no more questionable. The idea that language affects cognition dates back to the 1800s and it is often cited as the Sapir-Whorf hypothesis which, in its strong form, says that concepts and thoughts are fully determined by the language and the categories defined by it. Nowadays, we generally admit a weak version of this hypothesis according to which language has some influence on thoughts and concepts but that there is still a non-social component, presumably due to biological constraints, which shapes the basic thoughts and concepts of the mind.

Language is a complex phenomenon, which generally happens under the form of a dialogue between 2 or 3 persons. It appears that most theories which try to capture language as an individual process failed to account for the complexity of language in real life. Indeed, in many situations, dialogues are not understandable by an external rational agent. Even though there are precise grammar and semantic rules defining how language must be used, most of time dialogue does not follow such rules (Clark and Wilkes-Gibbs, 1986). Instead it consists in some exchange of many grammatically misconstrued sentences which do not mean anything if taken alone.

In order to account for such a phenomenon, Clark and Wilkes-Gibbs (1986) have developed the notion of grounding and common ground. If we analyze utterances during dialogue, it appears that people take advantage of what is known by their peers, and what their peers know about them to produce or to understand speech. Common ground refers to this set of knowledge that is shared by two people interacting and which is used to reach mutual understanding during verbal interactions. The concept of grounding is the process of building a common ground during an interaction. The theory of grounding attempts to explain language complexity by referring to intentional concepts: we use what we know that the other also knows to

determine how to say something in the best way. Taking a different stance, Pickering and Garrod (2004) proposes an alternative "mechanistic" theory which tries to explain dialogue from simple priming[2] mechanisms which lead to "language alignment". Alignment occurs because of mutual priming at various levels from phonetic to semantic through phonemic and grammatical level. Each discussant tends to reuse the tone, phonemes, grammatical structures or even concepts used by its partner. It results that they tend to become aligned through time and they end up with the same structures at all these various levels, including at the semantic level. This process would be largely unconscious except when misunderstandings occur that would trigger a conscious process to resolve it.

2.1.3 Computer supported collaboration

A more practical aspect related to the study of collaboration concerns the development of computer software, or more generally computer systems, supporting or enhancing collaborative activities. This domain of research which appeared in the 80s is subdivided into two main fields: Computer-Supported Collaborative Learning (CSCL) which focuses on the improvement of learning through collaborative activities on computing systems and Computer-Supported Collaborative Work (CSCW) which concerns more generally all collaborative problem solving activities. These fields are both closely related to distributed cognition theories as they emphasize the use of collaboration for improving learning or problem solving. Concretely, this consists in developing tools such as awareness tools (see for example Sangin 2009) which provide feedback to collaborators about actions or state of their peers or group mirrors (see for example Jermann and Dillenbourg 2008) which offer an "image" of the state of the collaboration. These kinds of tools aim at providing the collaborators with information that could help them to adapt to the collaboration. These tools are generally coupled with or embedded in an application that allows collaborative work through a shared workspace. Of course results about socio-cognitive processes of collaboration are of a great importance for the development of CSCL or CSCW tools. Conversely the study of the use of such tools can also inform us about the collaborative processes. We do not develop further these aspects because this thesis is neither about collaborative learning, nor about collaborative technologies. Instead, we aim at developing deeper understanding of collaborative processes and analysis methods that could, in the future, be used for the development of CSCL and CSCW tools.

2.2 Eye-tracking

In the present section, we offer a broad overview of the basic facts about human vision and eye-movements. We present also an introduction to eye-tracking technologies and the related methodology. This is a mandatory prerequisite in order to understand the developments that are done in this work.

[2] Priming is a psychological effect which makes that exposure to some stimulus influences responses to a later stimulus

2.2.1 Human visual system

The physiological and neurological mechanisms sustaining human vision are far beyond the scope of this work. However, it is interesting to have a broad overview of the basic aspects of the eye's functioning in order to understand well eye-movements analyses. Interested readers may refer for example to Duchowski (2007) for detailed explanations. Schematically, the eye is an aqueous sphere comprising a small aperture on one side and an internal photosensitive region on the opposite side (see figure2.1). This photosensitive region, called the retina, is responsible for gathering visual information and it is directly linked to the nervous system through the optic nerve. On the other side of the eye, the aperture is composed of different elements that constitute an advanced optical system. Without entering into details, this optical system has two main components. First, the lens, along with the ciliary muscle which modifies the lens curvature, controls the focal of the system in order to focus sharply on specific distance. Second, the pupil and iris plays the role of a diaphragm, allowing changes of the amount of light reaching the retina and of depth of focus.

We are not able to attend to everything in our visual field with great accuracy. Visual acuity is not uniformly distributed across the visual field because vision receptors are distributed across the retina in regions of varying densities. The center of the retina presents a high density area in terms of receptors called the fovea. The fovea provides the highest acuity of vision compared to the surrounding areas and covers approximately two degrees of the view field. The area outside the fovea is less accurate for perceiving objects and is referred to as the peripheral vision field. In order to see objects in detail, one must move the eyes in order to make the object of interest appear into the high-resolution area of the fovea. It is often assumed that the viewer's attention is also focused on objects within this specific area of the visual field.

2.2.2 Eye-movements

Before entering into the details of eye-tracking techniques and the major results in this field, it is worth to have a look at the general aspects of eye-movements. Eyes have two main dynamical modes, or eye-movements, which are fixations and saccades. Fixations, as indicated by its name, are still phases during which eyes make essentially no significant movement. Thus, a fixation is characterized by the angular position of the eyes and a duration, which is typically between 80 and 500 ms. Between two fixations, eyes make fast movements called saccades. These are very fast programmed movements with typical velocity reaching 200 °/s and consequently display short durations, typically between 10 and 80 ms. These two kinds of dynamical regimes, fixation and saccade, represent the main activity of the eyes, with around 80% of the time spent during fixations. There also exists a third kind of eye-movements, called smooth pursuits, which may happen in the case of object tracking. However, this type of movement is relatively more specialized and happens only in specific situations when some regularly moving objects have to be followed. In order to be exhaustive, it is also necessary

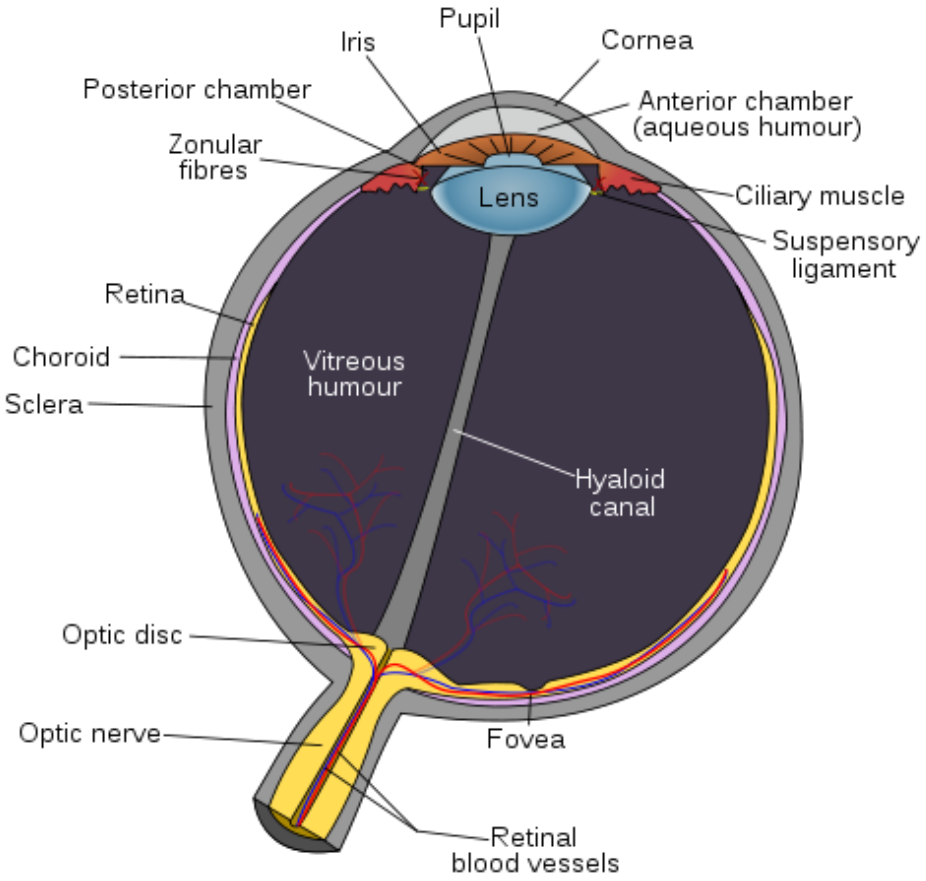


Figure 2.1: Schematic diagram of the eye. From *Wikipedia* by Rhcastilhos, 2007, retrieved 02 Sep. 2011, http://en.wikipedia.org/wiki/File:Schematic_diagram_of_the_human_eye_en.svg.

to cite atypical eye-movements, like vergence, which is simultaneous movements of both eyes to obtain or maintain binocular vision, as well as fixational eye-movements which are micro-movements occurring during fixations. These latter can be decomposed in different categories such as micro-saccades of high-amplitude and low frequency, and tremor of low amplitude and high frequency. One of their roles is to avoid stabilization of the image on the retina which would otherwise fade but it is still debatable whether such micro-movements have a real cognitive role to play. Interested readers may look at Duchowski (2007) for a detailed account of the different type of eye-movements.

2.2.3 Eye-tracking

The development of eye-tracking techniques has started more than one hundred years ago. Since then, a variety of methods have been developed to record eye-movements. Among the developed techniques, we can cite electrooculography (EOG) which consists in recording the electrical activity generated by the eye muscles with some electrodes, scleral search coil which uses electromagnetic induction in a metallic ring placed on a lens or even mechanical devices using a small balloon in contact with the eye and an arm and lever to transmit and amplify the movement of the eye. Other methods are based on a focused light which is reflected by the eye and then prints a photosensitive paper or is recorded by a camera. A complete review of these techniques is far beyond the scope of this work and can be found in Duchowski (2007). Globally the general idea is to develop a system which can measure eye-movements in the most accurate way while being the least intrusive possible for the subject. Indeed, intrusive systems, in addition of being quite disturbing for the subject, can alter eye-movements patterns by adding a physical constraint on the eye or by affecting the subject's cognitive processes.

In this thesis, we use a video-based screen-mounted eye-tracker, the Tobii 1750 (see Figure 2.2). This system appears as a normal computer screen with two infrared lights and a camera on the bottom. The infrared lights are reflected on the eyes and the camera records the eyes and the lights reflections. The video is analyzed by a dedicated computer software which extracts the pupil and the light reflections positions from the images. The vector formed by the pupil and the light reflection is associated to an eye positions through a calibration procedure in which actual eye positions are known. More precisely, the calibration consists in showing several points dispersed on the whole screen and in measuring the pupil and light reflex for each of these points, so that a mapping can be established between points on the screen and pupil-reflex vectors. Note however that this calibration can deteriorate over time (drift effect) if the conditions (subject position, lightning, eyes dryness) change and of course lead to increasing accuracy problems. These kinds of systems are nowadays widely used by psychology researchers because they don't require complex technologies on the one hand and they are completely unintrusive for the subject on the other hand. However, they suffer from accuracy and precision problems because of various difficulties related mostly with image analysis.

Such video-based eye-trackers have several important characteristics. Concerning timing aspects we have to consider sampling frequency, latency and time precision. Sampling frequency, in hertz, corresponds to how many gaze points are measured within one second. Latency, in milliseconds, is the time between the moment when the eye-position is recorded and the moment when the measure becomes available. Time precision, also in milliseconds, refers to the precision in the timestamps of the data. Concerning spatial aspects, we consider the spatial resolution and also two types of errors: accuracy and precision. Accuracy, generally expressed in visual angle degrees, indicates how well the measured eye location corresponds to the actual eye location and precision expresses how much multiple measures of a same eye location have the same value.

Figure 2.2: The eye-tracker used in this work, the Tobii 1750 (Tob, 2006).

The specific eye-tracker used in this study is the Tobii 1750 (Tob, 2006). It has a sampling frequency close to 50Hz with a time precision of +/- 3 ms and maximum latency of 35 ms. It has an accuracy of 0.5 degrees and a spatial resolution of 0.25 degrees while the precision is not reported. It can be used without resting the head as long as the head stays within a field of 21 x 16 x 20 cm (width x height x depth) centered at 60 cm from the screen. It could have a drift that can reach 2 degrees if great changes in light conditions occur. Note that all these values and information are the ones reported by the manufacturers but analyses from our part revealed that they are not always correct. Finally, note also that this eye-tracker does not provide measures in visual angles but only of the gaze positions measured in pixels on the screen.

Fixation identification

All eye-tracking systems generally provide the eye position either as Cartesian coordinates on a given surface, such as a screen, or as an angular position of the eye globe, at regular moment in time. These are called the raw data. However, as we have seen, eyes make specific movements such as fixations, saccades or smooth pursuits. When studying cognitive processes, it is much more meaningful to analyze directly those different types of movements than the raw eye locations. Indeed, fixations represent the most part of the time and moreover it is during fixation that we actually get visual information. For these reasons, we need to obtain a list of fixations from the raw gaze data measured by the eye-tracker. Saccades can be known by deduction as they appear between fixations but it is difficult to know the exact dynamic of these movements with low temporal resolution eye-trackers as the one used in this study.

Smooth pursuits occur only in specific situation with some dynamical stimuli and thus they are often ignored. For these reasons, we restrict our attention on the problem of identifying fixations from the raw gaze data, which is what is usually done by researchers in psychology.

Roughly, fixation identification consists in detecting episodes during which the gaze position varies very little and stays in a small perimeter opposed to moments of fast variations of position occurring in a constant direction. There is a variety of methods to perform this task automatically but none has been widely accepted nor is really fully satisfactory. Salvucci and Goldberg (2000) present a tentative taxonomy that identifies the most common techniques. Without entering into details, all these algorithms rely on testing whether the eye position remains in the same location for a sufficiently long period of time. This is accomplished either by estimating the instantaneous velocity of each gaze data or by measuring the dispersion of a set of adjacent gaze data. There are almost always at least two parameters associated to these algorithms: a spatial or velocity threshold used to determine whether gaze data are still close enough to constitute a fixation, and a temporal threshold which enforce a minimum duration for the fixation.

We may think that all these algorithms yield similar results and that their parameters may be easily chosen but it appears that this is actually a difficult problem. Karsh and Breitenbach (1983) have shown that small variations in the algorithms' settings are likely to produce huge variations in the resulting fixation patterns. While this result was mostly qualitative, other authors (Blignaut and Beelders, 2009; Blignaut, 2009; Manor and Gordon, 2003; Shic et al., 2008a) have quantified this problem. Blignaut and Beelders (2009) shows how variations of the spatial threshold affect the number of detected fixations, the dispersion within detected fixations and also the spatiotemporal pattern of the resulting fixations. Manor and Gordon (2003) also noticed that variations of the temporal parameter affect the result in a significant way. Last but not least, Shic et al. (2008a,b) have shown that variations of the spatial parameters may even inverse statistical results found in subsequent analyses on fixations duration.

The difficulty arises from the fact that the allowed amount of variation for a fixation can hardly be defined uniquely for any eye-tracker, any subject and any experimental situation. Indeed, there are two sources of variation of position during fixations. First, there are the fixational eye-movements which occur during fixations and which can vary between individuals as well as between conditions. Secondly and more importantly, there is the noise due to the eye-tracker, which varies a lot between eye-trackers and possibly also between conditions. As a result, raw gaze data during a fixation may actually have very different shapes depending on the eye-tracker as well as on the subject or condition.

Systematic errors post-correction

Eye-trackers usually require a calibration process before being able to record gaze positions in an accurate way. If this process is correctly accomplished, measured gaze locations should measure accurately actual gaze positions. One problem is that the correctness of the cali-

bration may deteriorate over time or when the subject moves during an experiment, thus introducing systematic errors in the measured gaze locations. Moreover, it appears that in some cases, even a correctly calibrated system may produce inaccurate data (Hornof and Halverson, 2002; Hyrskykari, 2006; Weigle and Banks, 2008). Interestingly these errors appear to be very systematic and it is possible that improvements of the corresponding eye-tracker software may solve such problems. However, in order to cope with the current situation, some authors have designed methods to correct automatically these problems afterwards. For example, Hyrskykari (2006) uses the fact that eye-movements make specific patterns during reading activities. Hence, he developed an algorithm which detects that a single line is read and when a new line starts to be read and he uses this information to make some correction in the measured eye locations. Hornof and Halverson (2002) use specific moments of their experiment (such as right before a target is being clicked) during which they can almost be sure of the subject's eye location. Thus, they measure during these moments the deviation between the measured and the actual gaze locations and they use this information to correct measured gazes during other moments.

2.2.4 Eye-movements control

Considering the importance of the eyes for our perception and considering that perception is at the core of cognition, it not surprising that eye-movements are related in some ways to cognitive processes. Since the invention of eye-tracking techniques, many researchers have done studies to relate eye-movements with cognitive activities. Yarbus (1967) did a simple study in which he presented a specific image to various subjects asking them to perform different tasks ranging from simple free viewing to making complex inferences, like estimating whether the people on the pictures were close relatives or identifying the ages of the peoples. The results of this study are quite striking and shows very clearly how the cognitive activity performed by a person looking at a picture influences her eye-movements (see Figure 2.3). From that, he concluded that the eye-movements were very little influenced by the content of the visual scene and much more by the cognitive processes engaged.

More recently, a study by Itti and Koch (2000) investigated a different claim, which is that eye-movements may be primarily driven by physical features of the visual scene. More concretely, they studied how the saliency, defined as a combination of orientation, intensity and color information, of an image could predict eye-movements made onto the image. They built a saliency map model which could predict which parts of an image are the most salient and thus have the greater chances to be looked at first. Their model appeared to be in good qualitative agreement with human psychophysical data. Subsequent studies (Itti and Koch, 2001) have refined the definition of saliency to make it match more closely with real data. Moreover, researchers have tried to develop models combining saliency information with high-level information corresponding to top-down influences (Navalpakkam and Itti, 2005).

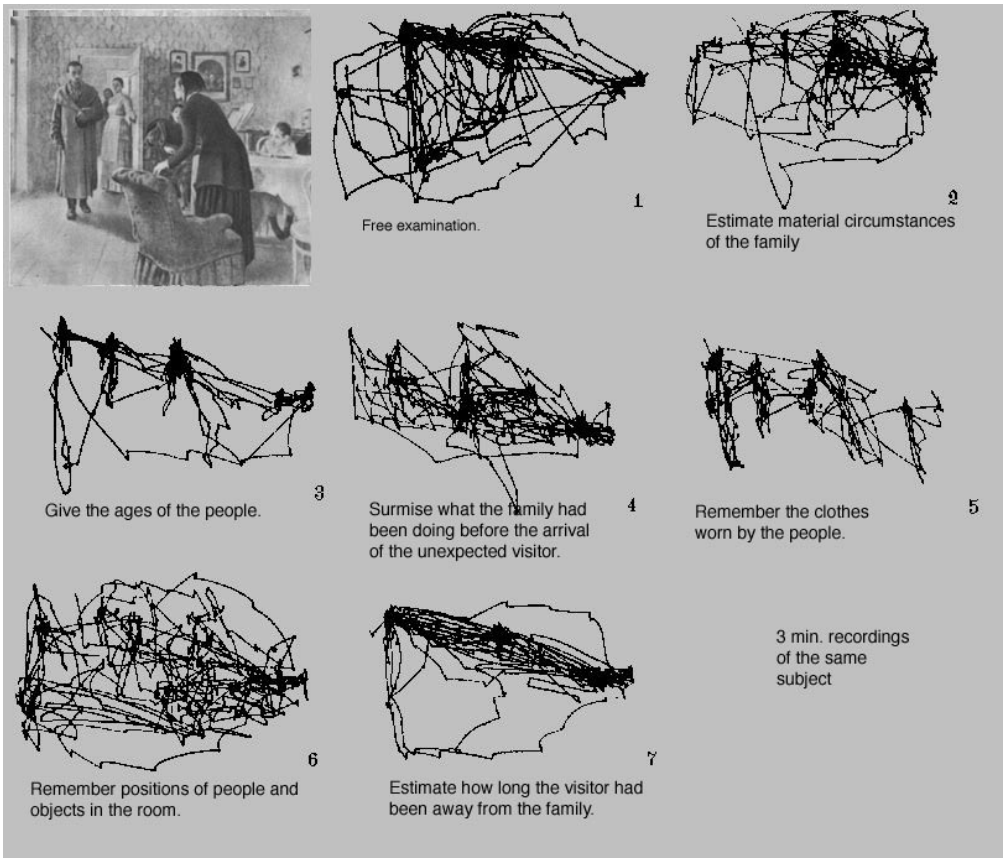


Figure 2.3: Image from Yarbus (1967) which shows the difference of eye-movement patterns for a single subject performing different tasks on the picture.

In the end, it appears that the usual opposition in cognitive sciences between bottom-up and top-downs processes is also present in this domain. On the one hand, it is clear that at some point the high-level cognitive activity, i.e. what we want to do with our visual content, influences in a significant way how our eyes move. On the other hand, it appears also that in some situations, such as the first moments of exposure to a new stimulus, low-level neurological processes may actually drive the eyes. A good explanation to reconcile these two aspects is that, when presented with a novel stimulus, eyes are mainly driven by low-level aspects of the visual scene, such as saliency. As the time goes and the person becomes able to recognize and understand the stimulus in a higher-level cognitive context, eyes start to be controlled by these high-level cognitive processes. Such an explanation is consistent with hybrid model of cognitive architectures, such as the parallel terraced scan (Hofstadter, 1995), in which perception and cognition begins by being mainly stimulus-driven and becomes more and more influenced by high-level concepts, through the building up of "representational hypotheses"

2.2.5 Eye-movements and collaboration

Several previous studies show that eye-movements may be related to collaborative activities. Indeed, it appears that gaze is largely influenced by speech which is at the heart of collaboration. Gaze appears to be used to monitor the environment while speaking or listening. More specifically, while speaking, there exists an *eye-voice span* which is a time delay between gazing at some specific object and uttering the name of that same object and conversely, while listening there is a *voice-eye span* which is the time between hearing the name of an object and the first gaze on that object.

For example, in a simple picture description task (Meyer et al., 1998), subjects started to gaze at the object to be named 700 ms before starting their utterance. Also they started to look at the second object to be named 300 ms before their utterance. In a more realistic situation, it has been shown in a simple sentence formulation experiment that speakers will to tend to look at the things they are referring to just before pronouncing the corresponding words, in average 900 ms before the word onset (Griffin and Bock, 2000). In this same study, Griffin and Bock showed that this effect was not significantly affected by variations of causal structure (active or passive) of the formulated sentence. On the opposite the order of mention was the main factor explaining what was fixated when. These results suggest that the visual scene is used as a kind of cognitive support during sentence formulation. Other studies (Griffin, 2001; Griffin and Oppenheimer, 2006) show that the durations of the fixations on the referents is affected by the difficulty of retrieving the corresponding word and also it is increased when speaker are forced to use an inaccurate label. This latter result tends to show that these gazes on referents preceding speech are not simple aids to find the word but rather a cognitive support for sentence construction. Finally, Zelinsky and Murphy (2000) have shown that there is a correlation between the time spent gazing at an object and the spoken duration for naming that object. This happens even when they don't have to name the objects but only to remember them. This suggests that linguistic information may be used even for pure memory tasks and hence this could affect eye-movement behavior.

In a similar way, Allopenna et al. (1998) have shown that in a simple task in which people have to find referred object among four objects, they gaze at the referent object some time after hearing the corresponding noun. More precisely, the probability of fixating a referent becomes larger than the probability of fixating any other object 500 ms after the onset of the word and becomes close to one 800 ms after the onset of the word. This is of course a very simple situation not comparable to dialogue but this gives a first insight on the time required for gaze to follow speech. We can however expect longer lags in more realistic, more complex situations in which the number of objects is larger, as this could require some time to find the object of interest.

Richardson and Dale (2005) studied the relationship between gazes of a speaker and of a listener of a monologue. Using a measure called cross-recurrence analysis[3], they showed that

[3] Cross-recurrence is a general measure that quantify similarity between two dynamical systems

there exists a coupling between speaker's gazes and listener's gazes. More specifically, the listener tends to look at the same objects as the speaker with a delay of 2 seconds. In addition, the level of this coupling is related with the level of understanding of the listener. The more the listener comprehends what she hears the more likely she will look at the same objects than the speaker with a lag of 2 seconds. In another study (Richardson et al., 2007), they showed that this effect was still present in a dialogue situation but with a maximum coupling occurring at a lag of 0 second. They hypothesize that this was due to the fact that they both speak and thus both lead eye-movements turn-by-turn, resulting in maximum recurrence at a lag of 0 s. Finally, they also showed that the general level of recurrence for lags lower than 3 seconds was influenced by the initial presence of a common ground between the subjects. These results are coherent with the effects described above found in simple situation. Indeed, if the speaker looks at visual referents before pronouncing their names and the listener looks at these objects after hearing their noun, they will look at the same objects with a delay corresponding to the sum of these lags. The sum of the lags found in simple setting is however smaller than 2 s. but it is likely that this difference comes from a difference of complexity.

2.2.6 Gaze-awareness tools

On a more practical side, dual eye-tracking technologies have promising applications with the development of computer tools that would analyze eye-movements of remote collaborators to provide them with information about their partner's gazes. We designate such tools with the broad term of Gaze Awareness Tools. We can distinguish two different ways of taking advantage of dual eye-movements to help collaborators. On the one hand we can simply provide collaborators with a trace of their partner's gazes, thus allowing them to replace gaze monitoring that may occur in face to face situations. This actually provides them with more information than they would have in face to face situations as they could have the exact point of gaze of their partner at any time but on the opposite, this could be overwhelming for them as certainly not all gazes are meaningful. On the other hand, a second way consists in finding reliable patterns in collaborators' eye-movements that could be predictive of some aspects of collaboration. Hence it could be possible to use the prediction of such a system to provide help to each collaborator or to adapt their shared interface (e.g. predict misunderstanding to encourage repair acts).

Several studies have been performed to explore the usefulness of transferring gaze information between collaborators. Some authors (Ishii and Kobayashi, 1992; Monk and Gale, 2002) have developed systems allowing full gaze awareness and eye-contact between collaborators. They showed a positive impact of such system for problem solving task in one case (Ishii and Kobayashi, 1992) and a reduction of speech quantity and ambiguity in the other case (Monk and Gale, 2002). Other researchers have done experiments with computer systems that simply project the gaze of the collaborators, generally as simple dots, on the workspace of their peer (Velichkovsky and M., 1995; Brennan et al., 2008). They seem to show a slight improvement of performance over similar condition without gaze sharing. However these

results are not clear and more research is required to really assess whether gaze transfer is beneficial and especially, under which form.

Eye-tracking methodology Part I

3 Eye-tracking methodology

Eye-trackers provide raw gaze data, i.e. the gaze location at regular time interval. In order to make meaningful analyses of this data and to relate eye-movements patterns to psychological processes, several very important preprocessing steps are required (see figure 3.1). Software provided by eye-tracker manufacturer offers some functions to perform part of this processing but as we will see, if we restrict ourselves to the simple processing done by this software, analyses have great chances to be biased and noisy.

This chapter presents developments that we have accomplished in eye-tracking methodology to perform this preprocessing in the best possible way. The structure of the chapter follows the different steps presented on figure 3.1. The first part is devoted to the problem of fixation identification. We present usual algorithms used to perform this task and we show problems encountered with these algorithms. We then present a new adaptive algorithm that we developed to overcome these problems. The second part concerns the correction of systematic fixation location errors. We show that the original data may suffer from huge systematic errors and that this can greatly alter subsequent analyses. We briefly present existing solutions found in the literature for correcting such problems and then show an automatic method that corrects these errors in many situations. Finally we tackle the problem of hit detection, i.e. the association of fixation to stimulus object, and we show why the usual simple solution may be biased. We offer a more elaborated, probabilistic solution to this question that limits the issues exposed. We end up with a discussion about the computation of gaze features, higher-level indicators that aims to summarize eye-movements patterns.

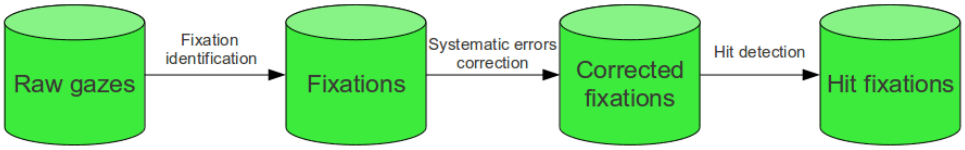


Figure 3.1: Eye gaze data preprocessing prior to experiment analysis.

3.1 Raw gaze data

The eye-tracker used in this study, namely the Tobii 1750, provides eye positions data for each eye in pixels on the screen every 20 ms. More specifically, each data point has the following information:

- Timestamp [ms]
- Right eye location on the screen [pixels]
- Right eye location on the camera [0..1]
- Right pupil size [mm]
- Right eye distance [mm]
- Right eye validity
- Left eye location on the screen [pixels]
- Left eye location on the camera [0..1]
- Left pupil size [mm]
- Left eye distance [mm]
- Left eye validity

Validity codes gives information about the quality of the data. By combining validity codes of both eyes, we get a general quality level which goes from undetected eyes (missing value) to good detection for both eyes with two intermediate levels which are respectively good detection for a single eye and correct detection with ambiguity on the eye. Generally, we consider only data that are well detected, and possibly also those with a single eye detected, and take the others as missing values. Those missing values have to be handled carefully during data processing.

Eye locations on the camera are coordinates between 0 and 1 indicating the location of the eye on the eye-tracker camera. This information is practically useless, except for checking that the subject is not out of the field of view, or too close from the border of the field of view. The eye distance information is an estimation of the distance between the eye and the screen.

Eye locations are given in pixels on the screen which has a resolution of 1280 pixels by 1024 pixels. Hence we do not have access to the angular positions of the eyes. It is possible to compute an approximation of the angular movement by using the distance information but this is not reliable because first, the head distance information is not reliable as an absolute value but only as a relative value (Tob, 2006) and secondly we don't know the relative position of the eye, in the plane parallel to the screen, relatively to the center of screen. To get an idea, one visual degree corresponds roughly to 1 cm (around 40 pixels) at a head distance of 50 cm.

3.2 Fixation identification

Most eye-gaze analyses are based on a segmentation of the gaze data into the different kinds of eye-movements. Eye-trackers measure the gaze location at a given sampling rate, yielding a time series of raw gaze positions. Then, it is necessary to identify the different kinds of eye-movements from this time-series of gaze locations. More specifically, we are generally interested in differentiating between fixations and saccades, the smooth pursuits being present only when faced with continuously moving objects. There is no widely accepted method to accomplish this procedure. Indeed, various fixation identification algorithms, also called fixation filters, have been proposed in the literature but there is no consensus on what would be the correct or the best solution. This is an important problem in eye-tracking research as it makes comparing the results from different studies very difficult.

Many algorithms used for this task, informally called velocity-based algorithms, rely on the same idea: they detect high increase of gaze velocity. Indeed, the main difference between a saccade and a fixation (and potentially a smooth pursuit) is the velocity of the eyes: it is very low during a fixation and high during a saccade. Alternatively, some algorithms, called dispersion-based algorithms, rely on the spatial dispersion of consecutive gaze locations which is eventually similar to the velocity (as higher raw gaze velocity is likely to produce more dispersed gazes). This latter method is sometimes considered as a better solution for low frequency eye-trackers as it is not possible to get a good estimate of instantaneous velocity with those systems. Finally, it is also possible to use the raw gaze acceleration as an indicator, the saccades producing a high acceleration peak followed by a high deceleration peak. However, this concerns mainly high-frequency eye-trackers as many data are required to estimate correctly the acceleration.

Salvucci and Goldberg (2000) proposed a non-exhaustive taxonomy of these algorithms and Duchowski (2007, pp. 137-153) also offers a good overview of some of these algorithms. One possible classification is based on the complexity of the mathematical model used to predict the type of eye-movement. On the one hand, we have *threshold-based* algorithms which classify gaze locations into saccade or fixation based on a threshold comparison. This threshold may concern the current velocity (e.g., Duchowski 2007, pp. 141-143; Salvucci and Goldberg 2000, algo. I-VT), the dispersion of consecutive gazes (e.g. Salvucci and Goldberg 2000, algo. I-DT; Manor and Gordon 2003; Blignaut and Beelders 2009; Karsh and Breitenbach 1983) or the acceleration (e.g., Duchowski 2007, pp. 141-143).

On the other hand, we have *model-based* algorithms which use more complex models of eye movements to differentiate between fixations and saccades. There is at least two different such algorithms: Hidden Markov Model (HMM)-based and Kalman filter-based. HMM-based models use a manually designed 2-states HMM with one state corresponding to fixation and one state corresponding to saccade. Each state has different emission probabilities of gaze velocity (Salvucci and Anderson, 2001). Kalman filter-based algorithms model the fixational eye-movements dynamic using a Kalman filter and then use the innovation (the error between

prediction and observation) of the filter to detect saccades (e.g., Sauter et al. 1991; Rewari and Poon 1993). Note that this class of algorithms is more complex to setup as they have more parameters to set compared to the threshold-based. Their advantages is that they are likely to produce better results as they could potentially accommodate better for subtleties present in real data (e.g. local variation of noise level).

In addition to this great diversity of fixation identification algorithms, there is also the problem of setting the parameters of a chosen algorithm. Karsh and Breitenbach (1983) recognized more than 20 years ago the importance of the fixation identification algorithm definition. They showed that the fixations scanpaths may vary greatly simply by changing the algorithm parameters and they emphasized "...how ambiguous such measures [N.B. statistics about fixations] may be if the various criteria [N.B. the thresholds] are left unreported." (Karsh and Breitenbach, 1983, p. 63). Moreover, it has been shown that these parameters may have very important impact on the subsequent analyses. Indeed, in addition to changing completely the statistics about fixations (fixation durations distribution, number of fixations, etc...), the choice of different parameters may also invert the difference between two conditions (Shic et al., 2008a,b). Finally, it has also been shown that these parameters may be subject-dependent, although the origin of this dependence is not well-defined (Blignaut and Beelders, 2009). Despite of those various issues, few works have been done in this field. From all these considerations, we think that it is an important foundational problem for eye-movements research and it requires more attention from eye-movements researchers.

With the simplest algorithms, namely *threshold-based* algorithms, the parameters space is generally limited to dimension 2. There is a spatial threshold which defines the maximum velocity or the maximum spatial dispersion for a fixation and a temporal threshold which defines the minimum duration of a fixation. The effect of the temporal threshold is quite predictable as it will simply prevent the detection of too short fixations (Manor and Gordon, 2003). However, the spatial threshold may change the identified fixations drastically and in a non-linear way (Blignaut, 2009).

In this section, we investigate these issues related to fixation identification. First, we explore in detail the effect of this spatial threshold of fixation identification process using a velocity-based algorithm similar to the one used in the default software. We show how it affects simple metrics such as the number of identified fixations, the mean duration of fixations or the dispersion of gazes inside fixations. We also show that this threshold affects each subject differently and we explore the reasons of these inter-individuals differences.

Secondly, we define a *fixation set quality score* which aims to measure the quality of fixation identification automatically and objectively. Based on this score, we perform a numerical optimization of the velocity threshold and we explore how this optimal threshold varies across subjects.

Finally, based on these findings about the variations of optimal thresholds between subjects, we design an adaptive fixation identification algorithm which takes into account variations

of noise in the raw data. This algorithm has only subject-independent parameters which are optimized using the *fixation set quality score*.

The results presented in this section are based on the analysis of the gaze data of 637 subjects recorded in four different experimental settings: a Tetris game, a Concept-Map building task (Sangin et al., 2008), a visual induction task (Nüssli et al., 2009) and a map annotation task (Cherubini et al., 2008). These experiments are detailed in chapter 5. All these experiments were run in collaborative settings with two subjects performing the task together. However, for the present study, each subject is considered separately as we were only interested in the individual gaze data. All experiments lasted between 20 and 50 minutes.

3.2.1 Inter-individual differences

By analyzing fixations produced by the default filter, we were surprised to find that some subjects had almost no fixation at all. These differences between subjects were not related to the gaze data quality, as they were found for subjects with similar levels of quality. After a careful analysis of the raw gaze data, it appeared that these special subjects greatly differed from others in their raw gaze data. More specifically, their *intra-fixation dispersion*, i.e. the variation of gaze locations during fixations, was much higher than for other subjects. In other terms, the maximum distance between two gaze points during a fixation could reach up to 100 pixels for certain subjects while it was less than 30 pixels for others (see figure 3.2). This variability clearly has a great impact on the identification of the fixation because it affects directly the measured raw gaze dispersion and velocity and thus, it changes the threshold value required for a good identification.

This was confirmed by conducting an analysis on the effect of the identification threshold on the number of fixations. We ran a velocity-based fixation identification algorithm for all thresholds between 20 pixels and 120 pixels with a step of 10 pixels on 18 sample subjects. As we can see on figure 3.3, when we compare the number of fixations with the velocity threshold, every subject has the same pattern of variation. For low threshold values, almost no fixation is detected because the *intra-fixation dispersion* is too high to pass the threshold. As the threshold increase, the number of fixation increase up to a maximum. Then, for higher threshold, it starts to decrease again because, some short saccades become detected as being *intra-fixation dispersion* and thus, multiple consecutive fixations get merged in one fixation. The interesting point is that, while the pattern is the same for every subject, it is slightly shifted for some of them. For these subjects, the maximum number of identified fixations is found for much higher thresholds. As an example, if we took the default value used by the eye-tracker analysis software (indicated by the vertical dotted line), we would end up with two subjects having almost no fixations while they could have 7000 (2000 for the second) fixations detected by using a different threshold value. Also, others subjects would clearly have suboptimal number of identified fixations (2000 instead of 4000)

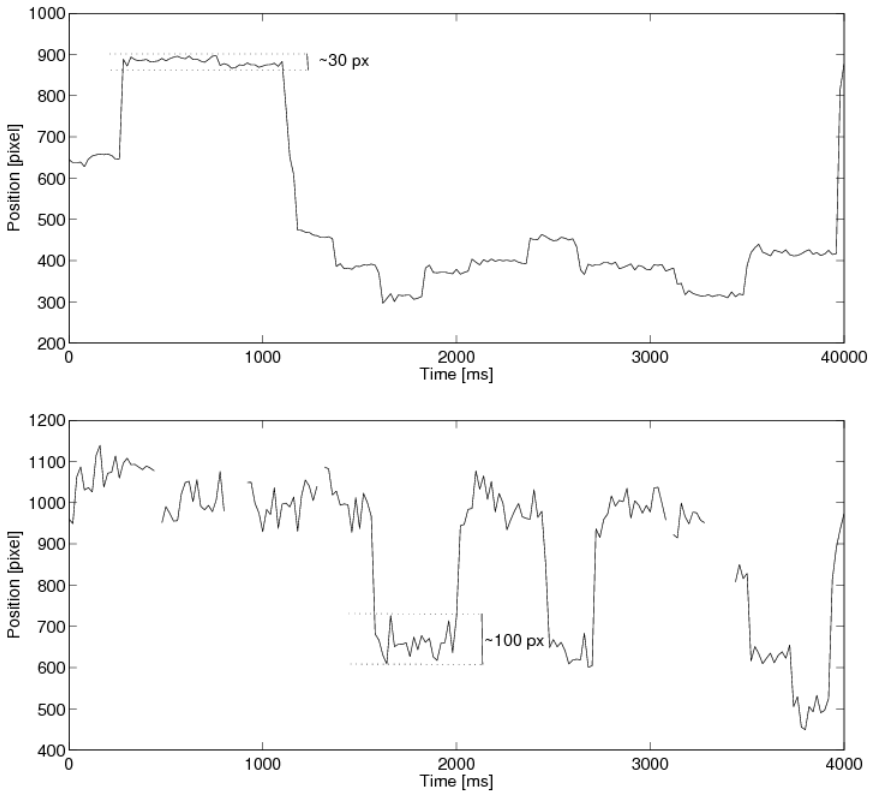


Figure 3.2: X-coordinate of the raw gaze data for a low-dispersed subject(up) and a highly-dispersed one(down). The scale range of the vertical axis is the same for both graphics.

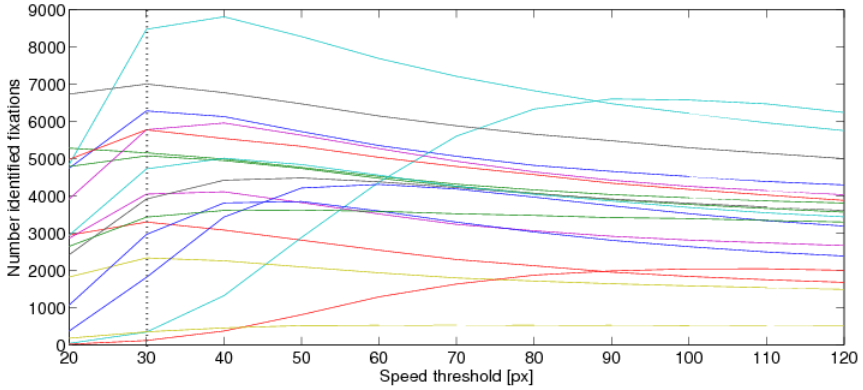


Figure 3.3: Number of identified fixations according to the velocity threshold used for identification. Each line is a different subject. The vertical dotted line indicates the default threshold used by the Tobii Clearview software.

3.2.2 Origins of differences

We investigated various hypotheses which could explain these differences in the *intra-fixation dispersion*. This dispersion is due to the fixational eye-movements on the one hand and to the eye-tracker noise on the other hand. Indeed, both are causes of gaze position variations during fixations. Hence, variations of this *intra-fixation dispersion* between subjects have two possible origins: physiological differences of fixational eye-movements or experimental artefacts that increase recording noise. Differences in fixational eye-movements may be due to intrinsic differences that may exists between subjects or to differences in the physiological state that lead to variations of fixational eye-movements. Differences of recording noise may be due to any factors that may affects the image processing done by the eye-tracker, which includes physiological differences and uncontrolled experimental factors, such as lighting or head distance. Unfortunately, the only aspects we could really investigate in our data were some of the uncontrolled experimental factors. Indeed, we had very few information on the physiological conditions of the subjects and apart from some qualitative case analyses, we could not explore these hypotheses in detail.

Subject's vision problems

One hypothesis was that vision problems or vision correction could explain these differences. This could either be due to modified eye dynamics because of the problem or the correction, or this could be due to recording artefacts because of the correction. We had information about vision problems only for one experimental setting which corresponds to 18 subjects but this small subset contained a good variety of different levels of *intra-fixation dispersion*. We compared the average dispersion between people with glasses or lenses and those without any problem but there was no difference. It was possible to find at both extremes a case with glasses or lenses. So, this hypothesis is rejected.

Lighting conditions

One possible explanation that we could not explore is the difference in lighting condition. Indeed, Martinez-Conde et al. (2004) reported that "...microsaccades tend to become larger in the dark..." and thus it could at least partly explain a higher *intra-fixation dispersion*. Also, according to Tob (2006, pp. 14-15), "Accuracy of the Tobii Eye Tracker varies depending on conditions (such as lighting and quality of calibration) and individuals" and "...a deterioration of a calibration [may happen]... mainly due to changes in characteristics of the eyes that are caused by change of pupil size (for example because of changes in surrounding light conditions...". These are some clues that these differences in the observed *intra-fixation dispersion* may be related to variations in lighting conditions. Thus, while we were not able to assess whether this was a source of problems in our first experiments, we controlled this factor in the most recent experiments.

Subjects' fixational eye-movements

It is also possible that the observed differences are partly due to subjects differences. More precisely, some intrinsic physiological factors may cause some subjects to have bigger fixational movements than others. Blignaut and Beelders (2009) cited among other explanations the age-related macular degeneration as a potential cause of increase in fixation stability (results reported by Timberlake et al. 1986). However, normal aging seems to not influence the fixational eye-movements. Kosnik et al. (1986) found no difference in the fixation stability between old and young subjects but they brought out the existence of great inter-subject differences in fixation stability. Particularly, they reported a young subject having much higher values of dispersion than others and this effect was consistent across two different experimental sessions (Kosnik et al., 1986, p. 1722). Moreover, they reported that subjects could be trained 'to control their eye-movements by providing real-time visual feedback' and decrease their fixational movements by half. This could suggest that fixational eye-movements do not have a stable pattern, even for a same subject.

In any case, differences in the amplitude of fixational eye-movements may explain only partly the observed difference of *intra-fixation dispersion*. Indeed, the differences reported by the various studies cited above are within a range of around 0.3 degrees which corresponds to differences around 10 to 15 pixels on a screen at 60 cm. The differences that we observed have a much greater range with differences of up to 70 pixels.

Head distance

Finally, we have investigated in greater details a specific hypothesis which is the existence of a relationship between head distance and average gaze velocity. Indeed, considering that the eye-tracker records gaze locations in pixels on the screen, the actual visual angle corresponding to a given pixel distance may vary depending on the head-screen distance. So, if we assume that the gaze velocity is constant in terms of visual angles, then it would be reflected as a non-constant velocity in pixels if the head-screen distance varies. We call this phenomenon *unit conversion effect*.

Tobii eye-trackers provide an estimate of the head distance for each gaze points, although that according to the Tobii Programming Interfaces manual (Tob, p. 137), "The values [head distances] given are best used as a set of relative values, not absolute". Hence we have used these values to test whether it affected the *intra-fixation dispersion*. We performed a meta-analysis on the gaze data of the 637 subjects. We computed the instant gaze velocity of each gaze point and tagged them as belonging to a fixation or to a saccade. As explained above, usual fixation detection algorithms may be hindered by variations of gaze velocity and thus, could bias analyses performed on the resulting fixations. So, in order to limit this problem, we used the adaptive algorithm that we have developed (see following section 3.2.4) and which uses an adaptive threshold to be independent of the variation of *intra-fixation dispersion*. Finally, in order to smoothen and to remove outliers from the data, we averaged the gaze velocity for

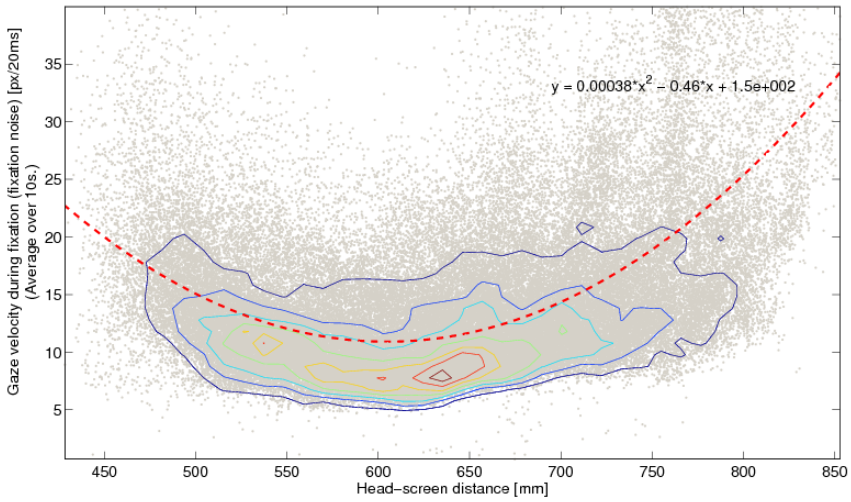


Figure 3.4: Fixation gaze velocity plotted against the head distance. Each data point represents the mean of these values over 10 seconds. As the number of points is very big (around 90000), the density contours are plotted on top of the points. The dotted line shows a quadratic best fit corresponding to the equation written on the graph.

fixations and saccades over segments of 10 seconds. This appeared to be sufficiently long to flood outliers in the mean and short enough to avoid large head movements during that period. The resulting variables are referred hereafter as the fixation gaze velocity (or fixation velocity) which is an indicator of the *intra-fixation dispersion* and saccade gaze velocity respectively.

We plotted these averaged velocities during fixations against the corresponding average head-screen distance value on figure 3.4. We plotted also the density contours of the points as they are too numerous to be distinguished. There is a non-linear, U-shaped, relationship between these variables. More specifically, it seems that a quadratic function fits fairly well with the data.

Another analysis supports this hypothesis of a quadratic relationship. We computed linear regressions between the head-distance and the fixation velocity for each subject separately. On figure 3.5, the slopes of these regressions have been plotted against the average head distance of the corresponding subject. This graph clearly shows the existence of a linear relationship, which is also confirmed by the high correlation of 0.84. Indeed, we have negative regression slopes for subjects who are close to the screen and positive slopes for subject far from the screen. This is entirely consistent with a quadratic relationship as suggested by the first result. We can indeed consider each slope to be an approximation of the tangent of the quadratic curve at the subject's average head distance. Of course, this is only a rough approximation as it supposes that the subjects did not move their head too much so that they stay on a small part of the parabola which can be locally approximated by a linear function.

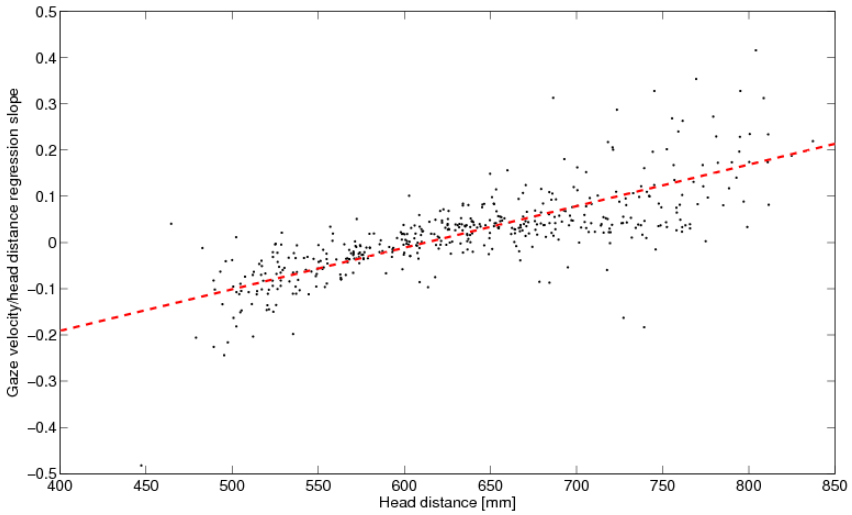


Figure 3.5: Slope of individual regressions plotted against average subject head distance. The y-coordinate of each point represents the slope of the regression between head distance and gaze velocity for a particular subject while the x-coordinate is the average head distance of that subject during the experiment.

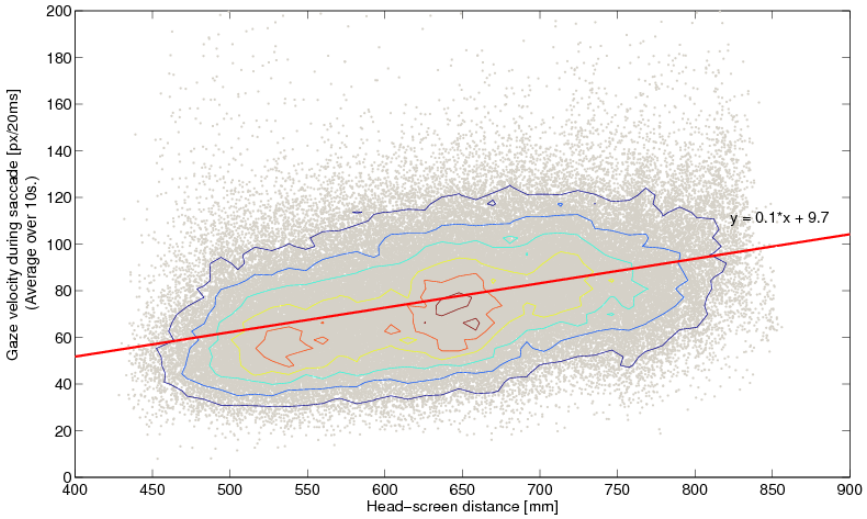


Figure 3.6: Saccade gaze velocity (averaged over 10 seconds) plotted against the head-screen distance. The dotted line shows a linear fit corresponding to the equation written on the graph.

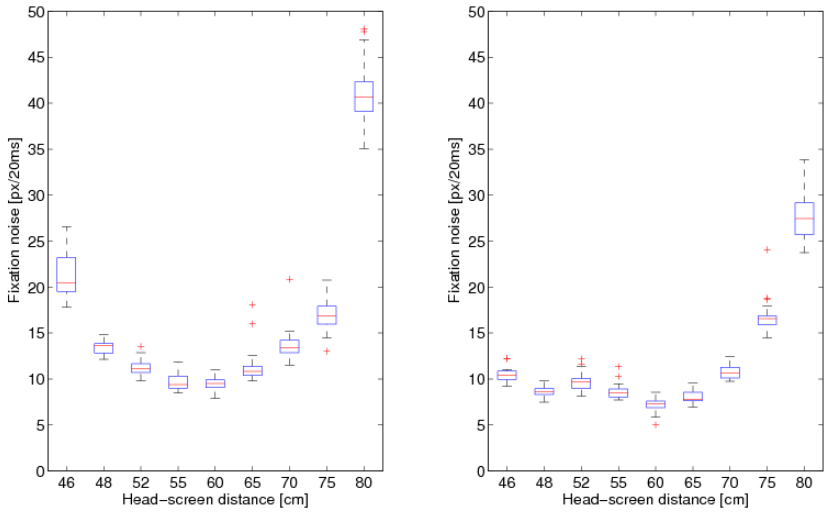


Figure 3.7: Boxplot of gaze velocity during fixations according to measured head distance for the two subjects of the controlled experiment. Left is M, right is P.

We also plotted on figure 3.6 the gaze velocity during saccades. Contrary to the fixation gaze velocity, the relationship looks much more linear in a direction consistent with the unit conversion effect, which is interesting as we would not have expected in different effect on the raw gaze velocity during saccades than during fixations.

Finally, in order to confirm these results, we ran a small controlled experiment on 2 subjects, M and P, in which we controlled precisely, using a head-rest, the distance between the head and the screen. We showed to the subjects a simple stimulus consisting of dots appearing at random locations. We repeated this task for different head distance ranging from 46 cm to 80 cm, which covered the range of detection of the eye-tracker. Figure 3.7 shows the fixation gaze velocity of the two subjects side-by-side. One of subjects, M, has a well-defined U-shaped relation while in the case of the second subject, the effect is less clear for short head distance. However, it is mainly one measurement set, at a head distance of 48 cm, which is inconsistent compared to the rest. Nevertheless, it is still pretty clear that the global pattern is U-shaped. Note that we did not fit any function because it appeared that the actual pattern is not really a perfect quadratic function. Instead, we can observe that the increase of the left branch (for head distances lower than 60 cm.) is smaller compared to the right branch.

Figure 3.8 shows the gaze velocity during saccades for the two subjects. Subject P has a more-or-less linear trend, which could be consistent with the meta-analysis. However, the other subject, M, does not show a clear linear trend. Instead, it could well also be a U-shaped relationship.

A limitation of this exploration concerns the fixation identification algorithm used. Indeed, the threshold of fixation algorithms should take into account the amplitude of the fixation

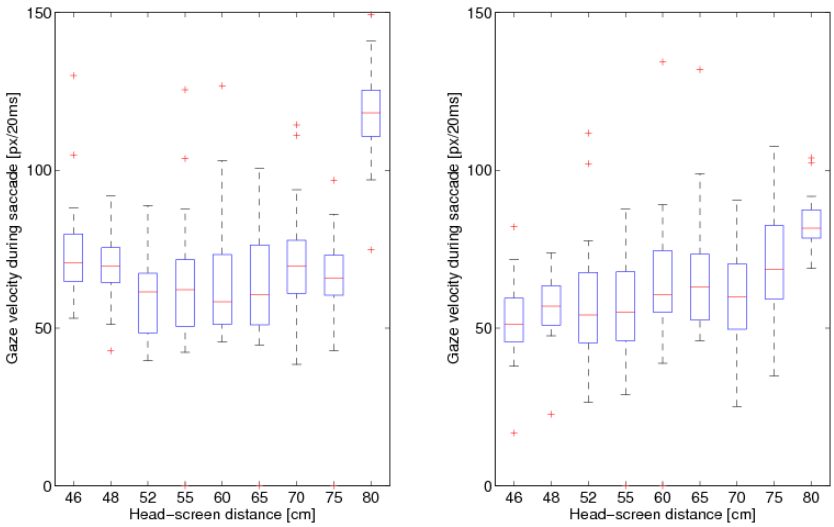


Figure 3.8: Boxplot of gaze velocity during saccades according to measured head distance for the two subjects of the controlled experiment. Left is M, right is P.

gaze velocity. And this is precisely the variation of fixation gaze velocity that we are measuring. However, we have overcome, at least partly, this problem thanks to an algorithm which adapts the threshold continuously to the local level of dispersion. Also, similar analyses done on the general gaze velocity, i.e. without considering separately the gaze velocity during fixations or during saccades, showed results very similar to those found for fixations. Thus, if a result should be questioned, it is more the linear effect on the saccade gaze velocity rather than the U-shaped effect of the fixations.

An unanswered question is why differences of head-distance affect non-linearly fixation gaze velocity, hence *intra-fixation dispersion*. The linear effect found for gaze velocity during saccades could be explained by a *unit conversion effect*, i.e. a given visual angle velocity yields increasing pixels velocity when the head distance increases. However, the quadratic effect found for gaze velocity during fixations could not be explained by this *unit conversion effect*. Actually, it is possible that the observed non-linear effect of head distance on *intra-fixation dispersion* is the combination of a *unit conversion effect* plus another effect of unknown origin. In any case, this does not explain why effects are different for fixations than for saccades.

Finally, we do not know if this effect is specific to the eye-tracker we have used, namely the Tobii 1750[1]. The generalization to other eye-tracking devices depends on the origin of the phenomenon. If this is related to an intrinsic property of video-based eye-trackers, it may well apply to other devices using a similar technology. In any case, this affects our data and thus we had to take it into account in order to perform clean analyses.

[1] Note that our experiments have been conducted on two different devices of the same model. This makes highly improbable the claim that it is due to problem specific to a particular device.

3.2.3 Empirical analysis of fixation identification threshold

Having recognized the existence of important problems concerning fixation identification caused by variations of *intra-fixation dispersion*, we have conducted extensive analyses on the threshold used for this process. Our goal was to find a way to assess the quality of the identification in order to find the best threshold that has to be used for each subject.

Possible cases of fixation misidentification

In order to get an idea of how to characterize the quality of fixation identification results, we made some qualitative analyses of raw gaze data and possible misidentification that could happen. We could identify two different cases of misidentification. The first case happens when the spatial threshold is too low compared to the *intra-fixation dispersion*. This results in the detection of *false saccades* during what are actually fixations. Thus, fixations are shortened, or split in multiple small fixations or even not detected at all (see figure 3.9, up).

The second case of bad identification appears when the spatial threshold is too high compared to *intra-fixation dispersion*. The result is that some saccades are wrongly identified as being *intra-fixation dispersion* and thus, multiple close fixations are grouped as one unique long *false fixation* (see figure 3.9, down).

Fixation identification quality metrics

From the qualitative analyses of fixations misidentification's, we developed a *fixations set quality score* which aims to assess the quality of a set of identified fixations given the corresponding raw gaze signal. The rationale is to be able to compare quantitatively the results produced by different algorithms or different parameters. This allows us to perform automatic comparison of algorithms and thus it becomes possible to perform parameter optimization. The *fixations set quality score* is based on two assumptions. First, we want to have a maximum total fixation time, to minimize problems of detection of *false saccades*. Secondly, the dispersion of the gazes inside a fixation should be minimal, to minimize detection of *false fixation*. Thus, we designed a fixation quality score based on two criteria: 1) the *intra-fixation dispersion*, defined as the spatial standard deviation of all the gaze points composing the fixation, must be minimized and 2) the total *fixational coverage*, defined as being the ratio between the total time spend in fixation and the total experiment time, must be maximized. The resulting score is simply the ratio of the fixational coverage over the *intra-fixation dispersion*.

It is important to note that this score produces good results as long as we impose a lower limit on the duration of a fixation. Otherwise, it would be possible to achieve a very high score by detecting many very short fixations containing only one or two raw gaze points. Moreover, another constraint that makes this score meaningful is the fact that between two consecutive detected fixations, there is always a saccade of 20 ms in-between. Hence, for a given time period, two identified fixations always cover less fixation time than a single one. Note also that

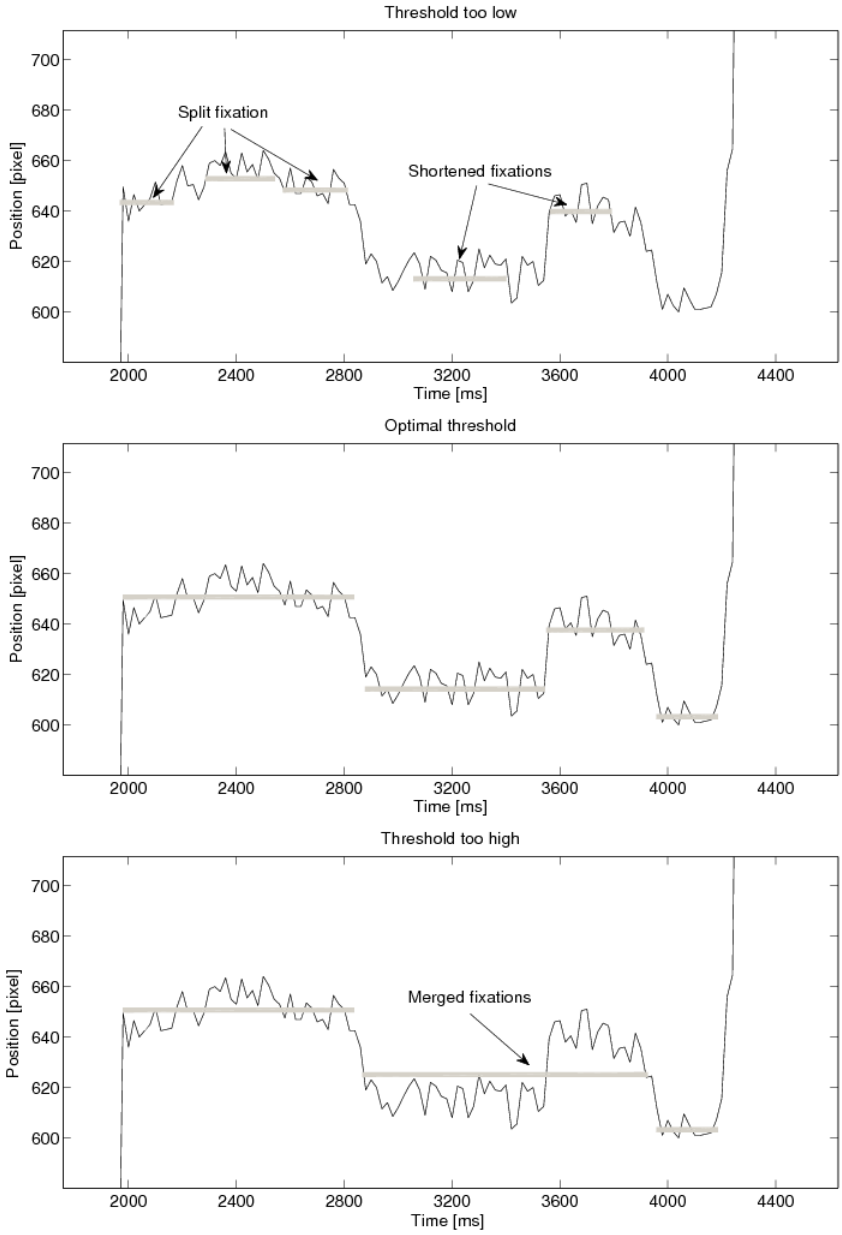


Figure 3.9: Examples of identified fixations (represented by the gray areas) with three different threshold values. Below optimum, fixations are split, shortened or not detected (top). Above optimum, fixations are merged (bottom).

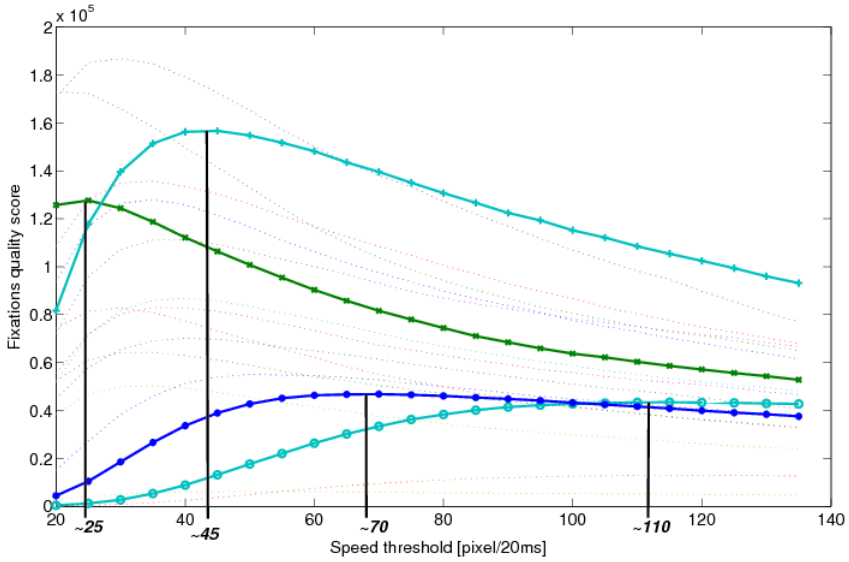


Figure 3.10: Fixations quality score compared to speed threshold for 18 subjects. The fixations scores (vertical axis) of different subjects are not comparable. The score peaks (corresponding to the optimal threshold) of four subjects have been highlighted.

this score is not intended to be compared across subjects or trials but only on a within-trial basis in order to compare the effect of various algorithms on the resulting fixations of a given trial.

The idea behind such a metric is that it allows us to define an optimal fixation-saccade segmentation based only on mathematical criteria and disregarding the fact that we are dealing with gaze data. Such an approach has also the advantage that the definition of the fixations is independent of the unit (pixel or visual angle) and of the intrinsic *intra-fixation dispersion*.

Systematic threshold analysis

We computed the *fixations set quality score* for each subject by varying the threshold values. The result for the same 18 subjects that were analyzed in section 3.2.1 can be seen on figure 3.10. The resulting curves follow a pattern similar to the number of fixations (see figure 3.3), exhibiting each a well-defined optimum. Thus, it is possible to find for each subject its optimal threshold value according to our *fixations set quality score*.

We ran a systematic optimization of thresholds on our whole population of 637 subjects. On figure 3.11, we see how the resulting optimal thresholds are distributed. The distribution is clearly not normal with the great majority of subjects (more than 80%) being between 20 and 40 pixels. However, only 20% of the subjects have an optimal threshold equal (with a tolerance of 5 pixels) to the default threshold used by Tobii software (30 pixels). Moreover, there are

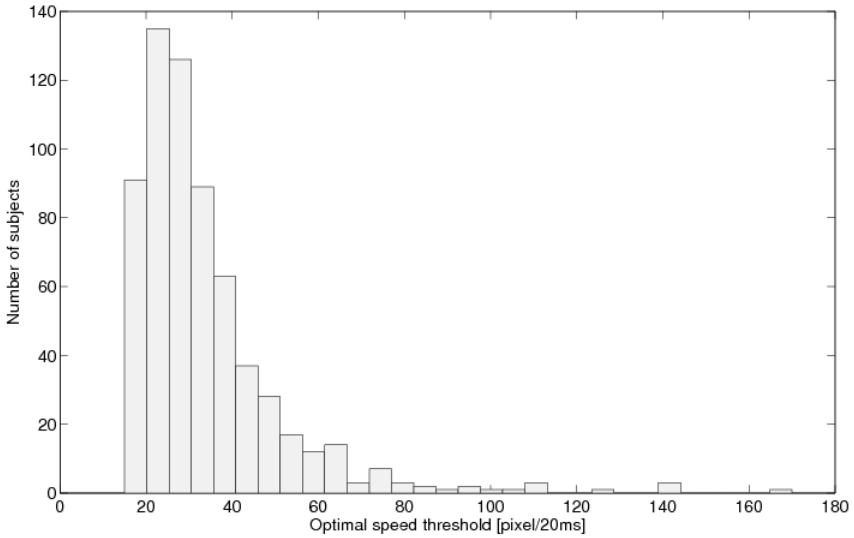


Figure 3.11: Distribution of the optimal thresholds for the 640 subjects.

about 20% of the subjects which have thresholds bigger than 40 pixels and 10% of the subjects have a threshold equal or bigger to 60 pixels.

Relationship between optimal threshold and average gaze distance

We investigated the relationship between these optimal thresholds found empirically and the average raw gaze velocity computed over all gaze data. Indeed, if the intrinsic *intra-fixation dispersion* is bigger for subjects who have higher threshold, it should be reflected in the average gaze velocity which should be also higher. The results are consistent with our expectations and we found a very good correlation ($\rho = 0.84$) between the average gaze velocity and the optimal threshold (see figure 3.12). This correlation was even better if we removed outlier gaze locations, i.e. points that are tagged as good data by the eye-tracking device but which do not have consistent positions (more than 150 pixels outside of the screen). By using a simple linear regression, it is possible to get a formula to compute the optimal threshold of a subject given its average raw gaze velocity. The resulting formula is:

$$T = 1.3 * V - 8.4 \tag{3.1}$$

where T denotes the optimal threshold and V the average raw gaze velocity.

Fixation quality improvement

Finally, we have analyzed the increase of *fixations set quality score* when using the optimal subject-dependent threshold compared to using a fixed default threshold for all subject. We

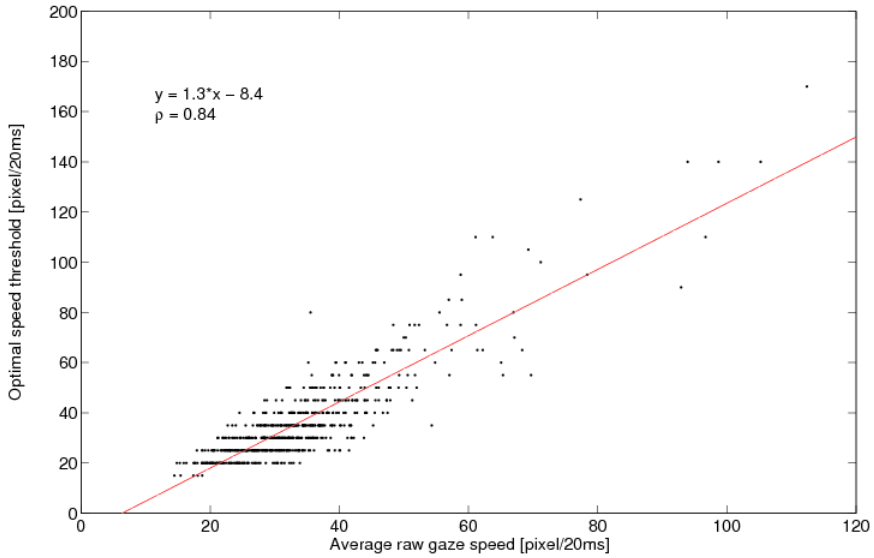


Figure 3.12: Relationship between average raw gaze speed (equivalent to the average distance between consecutive POR) and the optimal speed threshold.

have also compared this optimal threshold with the threshold computed using formula 3.1. Table 3.1 shows the average improvement in the fixation quality score for all 637 subjects. We see that the *fixations set quality score* improves by 50% in average if we use the optimal velocity threshold instead of the default one. Moreover, we see that even with a simple estimation of the threshold based on formula 3.1, we already get a huge improvement of 44%. These results show that the use of a fixed threshold identical for all subjects is very detrimental for the quality of the identified fixations.

Table 3.1: Improvement of *fixations set quality score* between default threshold, estimated threshold from raw gaze velocity and formula 3.1 and optimal threshold found through optimization.

	Improvement
default -> optimal	50.23%
default -> computed	44.17%
computed -> optimal	4.07%

The existence of a linear relationship between the average gaze velocity and the optimal threshold already offers a potential solution to our problem, namely choosing the best threshold for each subject. Indeed, it is possible to first compute the average gaze velocity of a subject and then, to use formula 3.1 to compute its optimal threshold. Such an approach leads to an average fixations quality only 4% smaller than the optimal threshold (compared to 50% worst for a fixed threshold)

Moreover, we have also used an approximation of the average gaze velocity computed only over the first one or two minute of recording. If we use such an approximation to compute the optimal threshold with formula 3.1, the identified fixations will have a score 8% lower than the optimal fixations (10% for the approximation over 1 minute) versus 4% when the average speed is computed over the whole experiment. So, this means that such a technique could actually be used in almost real-time situations.

3.2.4 Adaptive algorithm design

The solution offered in the previous section, i.e. to estimate subject's optimal threshold from its raw gaze velocity, is not fully satisfactory. First, it does not reach the optimal threshold in all cases. Secondly, it will not handle any potential change of *intra-fixation dispersion*, hence of optimal threshold, over time, which may happen, for example when the subjects move their head (see section 3.2.2). Finally, it has some constraint to be used in real-time as we need one or two minutes for the estimation of the threshold. These considerations lead us to design an adaptive algorithm which could estimate the threshold continuously.

We can find some examples in the literature of adaptive fixations identification algorithms. Duchowski (2007, pp. 150-152) describes briefly two such examples of adaptive filters. More recently, Nyström and Holmqvist (2010) have proposed a more elaborated adaptive fixation filter. This is an iterative algorithm which reestimates the threshold level at each pass using the fixations identified in the previous pass. It repeats this process until it converges to some stable value. While it is a very good algorithm producing nice results, it still has the limitation of not being usable in real-time application.

Algorithm design

The main idea of our adaptive algorithm is to estimate the optimal threshold for fixation identification by using the last few seconds of gaze. The general behavior of the algorithm is similar to the classical velocity-based algorithm (Salvucci and Goldberg, 2000, algo. I-VT). The difference is the addition of an adaptive threshold computation part. This part collects the velocity of the previously encountered gaze samples and uses these values to compute the current threshold (see striped part on figure 3.14). The mean of these collected values is used to compute the current threshold using a linear function. Basically, it consists of computing a gaze velocity moving average and then of applying a linear transformation on this average. Thus, while we had one parameter, the threshold, in the original velocity-based algorithm, we now have instead three parameters: the number of collected gaze samples (**WS** for window size) and the two parameters of the linear transformation used for the computation of the threshold (**THS** and **THI** for threshold slope, respectively threshold intercept)

A simple implementation of this algorithm yields only poor results, with fixation scores even lower than by using a fixed threshold. It appeared that by using this method, the moving

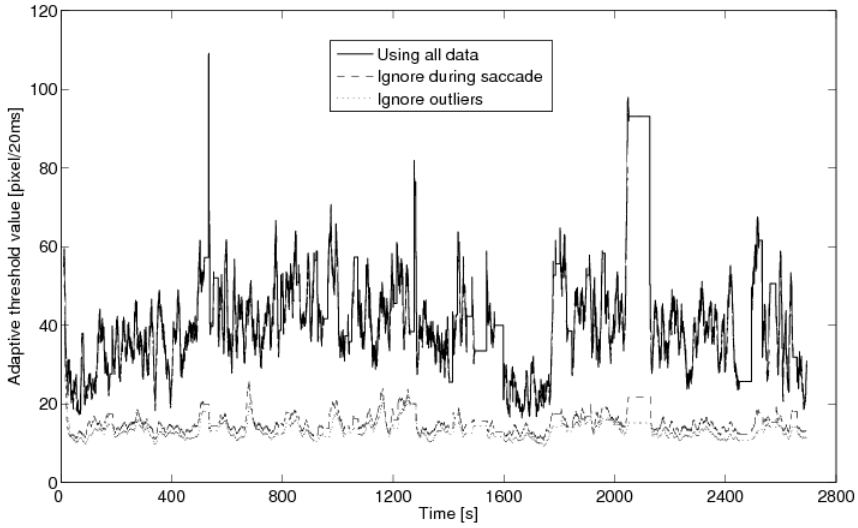


Figure 3.13: Evolution of the moving average value by ignoring or not some of the data point

average and thus the threshold value, had great variations during the time-course of the gaze data (see figure 3.13, continuous curve). These variations were greater than the expected threshold variations within a subject, ranging for example between 20 and 60 pixels. The reason was mainly the presence of long saccades which yields several consecutive high velocity points. The moving average was greatly affected when such point were encountered and thus became very high for a certain time. Thus, small saccades were not detected as such anymore if they followed a big saccade.

In order to avoid this problem, it is necessary not to take into account the velocity during saccade into the computation of the mean. This is a real difficulty because we are faced with a recursive problem: we need the moving average to know where the saccades are and we need to know the saccades to compute the moving average correctly.

The solution which has been retained was to use another criterion for the decision of including or not the current velocity value in the moving average. More precisely, we use a statistical outlier test which consists in comparing the current value to the current mean plus a certain number of standard deviations of the values currently in the window (see figure 3.13, dotted lines). The results using this method were quite good. However, this introduced a new parameter, **WSF** for window standard deviation factor, which corresponds to the factor which multiplies the standard deviation to determine the outliers limit.

The final algorithm is presented schematically on figure 3.14. We have explicitly separated the adaptive threshold computation part from the fixation identification part, this latter being similar to the original velocity-based algorithm. The algorithm has 5 parameters:

- **WS**: Window-size, the maximum number of values collected in the moving average window
- **WSF**: Window standard deviation factor, the factor which multiplies the standard deviation of the collected values to decide if a new value is an outlier or not
- **THS**: Threshold slope, the coefficient of the linear transformation applied to compute the velocity threshold from the moving average mean
- **THI**: Threshold intercept, the intercept of the same linear transformation
- **MFD**: Minimum fixation duration, we used a value of 100 ms in this study. Note that in our case, the effect of this parameter is predictable as it simply filters out short fixations

The values of these parameters have been optimized so that the average fixation score improvement over all subjects is maximized (see following paragraph).

Moreover, the algorithm uses the following variables:

- *vg*: Gaze velocity, velocity of the current gaze sample (distance between current and previous point)
- *window values*: Set of the last **WS** non-outlier gaze velocity values
- *wt*: Window threshold, threshold to decide if the current gaze velocity(*vg*) is an outlier or not(and thus if it can be included in *window values*)
- *mv*: Mean value, current mean of *window values*
- *sv*: Standard deviation value, current standard deviation of *window values*
- *th*: Threshold, current value of the adaptive threshold
- *current fixation*: Buffer containing the gaze samples of the currently identified fixation
- *fixations list*: List of already identified fixation

Finally, the two following formulas define how *th* and *wt* are computed from other variables:

$$th = \mathbf{THF} * m + \mathbf{THO} \tag{3.2}$$

$$wt = m + \mathbf{WSF} * sv \tag{3.3}$$

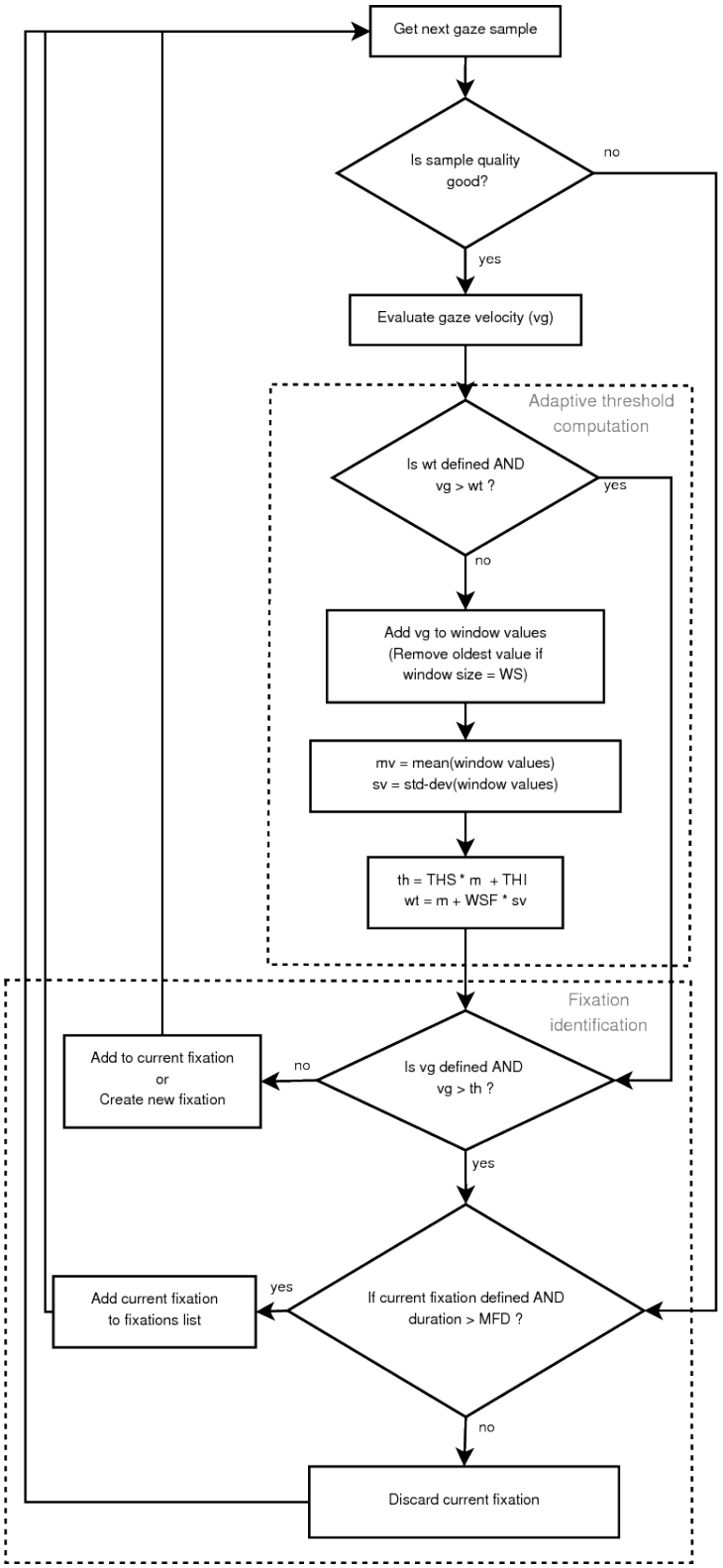


Figure 3.14: Flowchart of the adaptive speed threshold algorithm. Upper-case words in formulas represent parameters (constant values) and lower-case words are variables.

Parameters optimization

We have conducted an optimization process in order to set the various parameters related to the adaptive threshold computation. This was accomplished by a systematic exploration of the parameters space using the *fixations set quality score* as the fitness function. However, as the computation time for the fixation detection of 600 subjects is relatively long and considering that we had to explore a four-dimensional parameters space[2], we used only a subset of our subjects to test a specific set of parameters values.

Also, because of the large number of parameters, it is not possible to visualize directly the fitness landscape in order to exhibit the existence of local and/or global optima, which makes the optimization more difficult. Fortunately, preliminary coarse-grained analyses of the parameters revealed that two of them, namely the window size (**WS**) and the threshold intercept (**THI**), have only very slight effects on the fitness score compared to the two other parameters. Thus, we explored systematically only the threshold slope (**THS**) and the window standard deviation factor (**WSF**).

It is also important to note that the actual fitness score used was not the fixation quality score per se as this value is not comparable across subjects. The actual score of an individual subject was the ratio between the fixation quality over the optimal fixation quality found using the systematic threshold analysis of section 3.2.3. These ratios were then averaged over all subjects in order to get a unique score for a specific parameters set. Such a ratio reflects the improvement (or the loss) in percent compared to the simple algorithm with an optimal fixed threshold.

We did not find a well-defined fitness optimum. Rather, we found that we could have similar results for different pairs of (**WSF**, **THS**). For optimal parameter values, the ratio of the mean fixation score compared to using a fixed threshold filter with an optimal threshold is 99.3%. Hence, the performances are similar to using the optimal threshold. Compared to the other proposed solution which consists in estimating the threshold from the average gaze speed beforehand, the average performance are still better. Moreover, the adaptive algorithm does not require any special procedure to be initialized, while an a-priori estimation of the average gaze speed requires to have at least one or two minutes of gazes recording before application[3] Finally, the adaptive algorithm has the great advantage to be able to take into account potential variations of the gaze speed which would occur during the recording of a specific subject[4]

[2] Parameter **MFD** (Minimum Fixation Duration) was not optimized as it is not related to the adaptive threshold computation. This would anyway require to have a different fitness score as the current score could be affected by this parameter in a way that would not reflect a variation in the actual fixation quality.

[3] This is however not completely true in the sense that the very first fixations identified by the adaptive algorithms have great chances to be badly identified as the estimation of the threshold is based only on very few gazes values.

[4] This could happen for example if the head distance changes during the recording.

3.2.5 Discussion

We have shown the existence of important differences between subjects in the fixation identification process. These differences are due to a subject-dependent *intra-fixation dispersion*, i.e. lesser or greater variations in the gaze dispersion during a fixation. This affects the fixation identification as this process relies on the relative stability of gaze locations during a fixation. It appears also that these differences are at least partially explained by variations of the subjects' head distance. Indeed, we have shown that if the head distance is bigger or smaller than the optimal head distance, the *intra-fixation dispersion* increases.

From an analysis of bad fixation identification cases, we have defined a score that measures the quality of the fixation identification process. Thanks to this score, we have found the optimal velocity threshold of each subject. We have shown that, although most of the subjects have a threshold value comparable to the default value, between 20 and 40 pixels, other subjects have higher values, up to 120 pixels.

We have also shown that this optimal threshold of a particular subject could directly be estimated from the average raw gaze velocity. The use of the optimal threshold (or the estimated one) leads to high improvement, 50% in average, of *fixations set quality score* compared to the use of default settings. Hence, we think that it is an important improvement which is required for reliable analyses

We have designed and implemented a new velocity-based fixation identification algorithm with an adaptive threshold which could overcome the problems due to individual differences. We have then optimized the parameters of this new algorithm by using the increase of fixation score compared to the score of the optimal threshold as the fitness function. Although this algorithm has more parameters than the original velocity-based algorithm, it has the advantage that the variation of the parameters affects all subjects in the same way, thus allowing us to define an optimal subject-independent value for these parameters. Moreover, this algorithm, or more specifically the design approach followed for its development, is a step towards a potential device-independent algorithm as its fixations definition (through the fixations set quality score) does not rely on a specific gaze unit (pixel or visual angle). Of course, this is theoretical and it needs to be tested against visual angle data to see whether the results are the same. Specifically, it would be interesting to check that the optimum is reached for the same parameters values. If we could have a unique algorithm that works for any eye-tracker with the same parameters, that would clearly a great step for eye-movements research.

These analyses rely in a great measure on our fixations quality metric. This metric tends to be high when the fixations cover a great amount of time and when they have a low fixation noise. It is unclear if it is actually possible to define optimal fixations by using such a metric. However, we think that if it is not possible to define an operational fixation definition at the physiological or cognitive level. Such a metric has at least the advantage of defining the fixations in a purely mathematical way and thus it allows researchers to have the same operational definition of fixations. It is clear that our metric is not perfect as it relies on the fact that very short

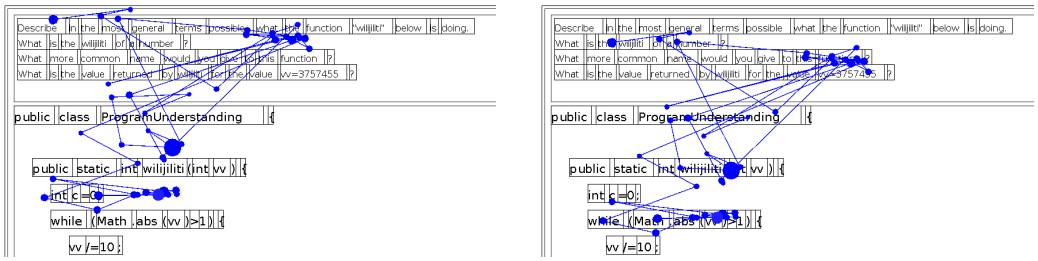
fixations are rejected. The other solution offered to researchers is, as proposed by Karsh and Breitenbach (1983) or Salvucci and Goldberg (2000), to report precisely which algorithm has been used along with the exact values of the parameters used. Although these practices are already a great step, it would be even better if we could all have the same definition of fixation. Another argument for the existence of an optimal threshold is that we can clearly see cases of fixations misidentification, as shown on figure 3.9. For example, it seems wrong to consider fixations of the top graph or of the bottom graph. The middle graph depicts quite clearly a "better" fixations identification.

3.3 Systematic accuracy errors correction

Eye-tracker accuracy errors represent the deviation between the measured location and the actual gaze location. It appears that there could be systematic trends in these errors and this could be a real problem as it could bias subsequent analyses. Consider for example the fixations presented on figure 3.15. It appears clearly that they are shifted and that the actual fixations look much more like figure 3.15b. Indeed, before the correction we have many gazes which are on blank parts, which seems not very realistic. If we analyze directly the uncorrected version (figure 3.15), we would have the impression that some parts, such as the mathematical formula on the bottom, are never looked at while if we take the corrected version, these parts are clearly looked at.

In order to overcome problems of systematic accuracy errors, different solutions have been proposed. A general common solution consists in having one or several additional pseudo-calibration phases before, during and/or after the actual stimulus presentation by using a specific stimulus similar to the usual calibration stimulus. The data on this stimulus can be used to estimate a gaze offset a posteriori. For example, Frank et al. (2011) included three instance of a 11 s. calibration stimulus in their experiment. They used the data recorded during these calibration stimuli to detect and reject a posteriori subjects who had really big errors of accuracy. Moreover, they also used this data to make a regression in the screen space, in order to find the best translation and/or scaling so that the gaze data match the pseudo-calibration stimulus at best (Frank and Vul, 2010). Similarly, Kammerer (2009) asked her subjects to fixate successively nine equally-distributed dots on the screen before doing the actual task. She used the gaze data corresponding to this specific stimulus to compute a horizontal and a vertical offset for each of the dots. The gaze data corresponding to the actual stimulus was then corrected using the offsets measured in this way. Specifically, each new gaze data was corrected according to a weighted mean of the offsets of the four closest recalibration dots.

Hornof and Halverson (2002) have developed a more sophisticated approach by using what they refer to as the *implicit required fixation locations*. These are specific locations at specific moment in the experiment for which it is almost certain that the subject looked at. In their particular case, a search task, the searched target, just before the subject click on it, is



(a) Before (b) After

Figure 3.15: Example of recorded gaze data with systematic errors (left) and after correction of these errors (right)

considered as such a location if the subject succeeded in the trial. They computed for each of these implicit required locations the vector between the corresponding fixations and the location. They use this value in real-time in order to trigger a new calibration whenever this value exceed a certain threshold and they showed that this allows to decrease the amplitude of the deviation for the subsequent trials, indicating a usefulness of the triggered recalibration. Moreover, they also analyzed the direction of the deviation between the implicit locations and the fixations. They show that this deviation actually has some systematic trend in it which varies from subject to subject and also sometimes from screen regions to screen regions. By correcting this systematic error, they managed to decrease the average over 16 subjects of the deviation from 41 pixels to 12.6 pixels. In a more recent study (Zhang and Hornof, 2011), they developed a new method which avoids the requirement of *implicit required fixation locations*. Instead they compute for each fixation the deviation vector to the closest AOI and plot all these deviations on a 2 dimensional graph. They use a mean shift algorithm to find the center of a cluster in this resulting graph and show that this center corresponds to the actual systematic error trend. This method works even though in many cases the closest AOI is not the actual object looked at because in those cases, the deviation has no systematic trend while in the cases for which the closest AOI is the actual object looked at, there is a systematic trend corresponding to the systematic errors. Hence, the systematic deviation vectors form a cluster while bad deviation vectors are uniformly dispersed.

Hyrskykari (2006) used a more complex algorithm to compensate for such systematic errors while analyzing gaze data during reading. This algorithm is based on the fact that we read a text line by line from left to right. Roughly it compensates for vertical errors while a line is read by considering that the same line is read until there is a long horizontal backward movement. It also detects the beginning of a new line being read and uses these events to adjust the mapping by assuming that the next line is read. Finally, as this algorithm is used in a real-time gaze-contingent application, it also has a manual correction mechanism allowing the user to inform the system if the wrong line is identified as being read. This algorithm has the advantage of being able to correct errors which are not systematic over the whole screen but it has the drawbacks of being usable only in reading situation.

In this study, we propose a novel method to identify systematic gaze deviations, called gaze offset hereafter, by using only natural eye-movements and knowledge about the viewed stimulus. Similarly to Hornof and Halverson (2002), we use the idea of *implicit required fixations locations* but in a more general way. Indeed, in many situations, like for most computer software-based stimuli, it is possible identify blank regions where the user should never, or very rarely, look at. By using such an assumption, it is then possible to find an optimal gaze offset which minimize the quantity of gaze in blank regions. While this is originally based on a common-sense assumption, we shall actually see that the results seem to indicate that this assumption is good. We propose two such offset estimation procedures: an optimal brute-force method which requires large amounts of computing power but which computes the most optimal offset (considering our assumptions) and a near-optimal analytical procedure which approximates the optimum by requiring less computing power, thus allowing it to be ran in real-time.

3.3.1 Automatic offset estimation

Our correction procedures use only natural eye-movements without requiring the presentation of any specific calibration-like stimulus. They are primarily designed to work on stimuli which do not cover completely the field of view but which, on the contrary, have blank parts, such as in many computer software's. More specifically, these algorithms rely on the following assumptions:

- People look at AOIs most of the time
- There exists a systematic gaze offset specific to a trial or subject[5]
- This offset is constant across space (the offset is the same on every place on the monitor)
- This offset is time-independent (It does not vary across time)

The first assumption is reasonable by common sense: when we are performing a specific task in a computer software, we have to look where the information is, that is in the AOI and not in the blank parts of the application. However, it could still be questioned in the sense that there may be some moments where the subject does not fixate into AOI. A possible example would be when the subject is doing purely cognitive activity, like mental computations. More generally, we can imagine that there is a kind of continuum between purely visual tasks, for which we can expect to have gazes only in AOIs and highly cognitive tasks, for which most of gazes does not fall inside AOI. So, in general, we could say that this assumption is valid as long as the task is visual enough, which is the case for us, and we expect it to be the case for most eye-tracking studies as otherwise it wouldn't worth to study eye-movements. Finally, let's note

[5] In our case, each subject did only one recording trial. Hence, both definitions are the same and hereafter, we refer to subject specific offsets.

that we will gain more insights about the validity of the assumption by looking at the results. Indeed, if this is true, we should find a well-defined optimal offset.

The three next assumptions are more hypotheses which have to be explored. Indeed, we don't have so much information to tell whether a possible systematic gaze offset may vary across the screen or across the time. Actually, there seems to be evidence that those are not true: the offset may change during a trial and it may be different on top of the screen than in the bottom of the screen. However, by considering these to be true facilitates greatly the offset estimation procedure. Thus, it has been decided to take them as granted and then, to explore their implications by running offset estimation only on part of the data. More specifically, the procedure has been applied separately to the first and second halves (in term of time) of the trials and the resulting offsets were then compared to find out if some differences exist. The same has been done for spatial variations by splitting the data into several sets depending on where the fixations were located.

Brute-force technique

Given the previous assumptions and by adding the requirement that we have an exhaustive description of all present AOIs, we can easily design a procedure which will estimate the most likely offset such that the amount of gaze into the AOIs is maximized. The idea is to define a fitness function which assigns a fixation likelihood to each point of the screen. This score should be high for points inside AOIs and should decrease as the distance to an AOI increases. The goal is to find a constant gaze offset which maximizes the average fitness over all fixations. The simplest way to achieve this is by using a brute-force approach which consists in testing every possible offset in a given range by computing for each one the resulting average fitness. This produces a landscape on a two-dimensional space (the two dimensions of the offset) in which we search for a maximum. Moreover, the shape of this landscape may inform us on the property of the offset. For example, if the resulting landscape is flat, this may mean that there is no optimal offset. On the opposite, if it reveals a well-defined peak, it means that there is a systematic offset which maximize the number of fixations inside.

The most important step is to define the fitness function G. The simplest form of such a function is a step function which assigns 1 to all points inside AOIs and 0 to all points outside AOIs, possibly with a tolerance distance to consider points close to an AOI as being inside. A more elaborated fitness function consists in defining a function $H(d)$ which computes a score value between 0 and 1 given the closest distance to an AOI. This must be a decreasing function having a value of 1 for a distance of 0 and which must tend to 0 at infinity. Different choices are possible for this function. For example, we can use a truncated linear function which decreases linearly from 1 to 0 up to a maximum distance and then remains at 0, or we can use a centered Gaussian function normalized to have its maximum at 1. Our choice was to use a logistic-like function. The rationale is that we want to allow gaze points to be a little bit outside AOI up to a certain distance corresponding roughly to the eye-tracker maximum non-systematic accuracy error and we want almost no gaze beyond this distance. This is well

modeled by a logistic function as it keeps a value close to one up to a certain distance and then decrease abruptly. More specifically, this function has the following formula:

$$H_{logis}(x) = \frac{1}{1.0 + e^{-b+xs}}$$

Where b is a location parameter and s a scale parameter. It is possible to set these parameters manually by choosing the length of the upper plateau, D_{high}, before the value drop down under 0.9 and the point, D_{low} at which the value reach 0.1. The parameters, b and s, can be computed from these two values using the following formula:

$$b = -\frac{D_{high} + D_{low}}{D_{high} - D_{low}} \ln(9)$$

$$s = \frac{-2\ln(9)}{D_{high} - D_{low}}$$

Finally, the fitness function can also be corrected to take into account the number of fixations that have been tested. Typically such a correction exaggerates the differences between low and high scores as the number of tested gazes increases. This is used to indicate that we have more confidence in high values and less confidence in low values, when the number of tested gazes increases. A good function that fulfills this goal if the fitness score is between 0 and 1 is a power function that raises the original fitness value to a power which depends linearly on the number of tested gazes such as:

$$G_{cor} = G^{c+Np}$$

Where N is the number of tested fixations and c and p are two parameters that have been set respectively to 0.4 and 0.00125.

The second step consists in choosing a range, O, of possible offsets to test, by keeping in mind that the more offset values we test the longer is the computation. We decided to test all offsets between -100 pixels and +100 pixels, both in vertical and horizontal dimension, with an increment of 1 pixel. This results in 40'401 ($201 * 201$) possible offsets to be tested.

Algorithm Let:

- $G(l)$ be the fixation fitness function taking a location l and returning the fixation fitness
- O a two-dimensional range of possible offset values
- F the set of all fixations
- R a result matrix of the same size than O indexed by an offset vector
- o^* the optimal offset

The algorithm proceeds as follow:

for all $(o_x, o_y) \in O$ **do**
  for all $f \in F$ **do**
    $R(o_x, o_y) = R(o_x, o_y) + G(f_x + o_x, f_y + o_y)$
  end for
  $R(o_x, o_y) = R(o_x, o_y)/size(F)$
end for
$(o_x^*, o_y^*) = argmax(R)$

We can also normalize the resulting fitness matrix, R, so that the values can be viewed as probabilities. It is then possible to get an indication of the offset quality by computing the variance of the offset in that way. The results of this method are presented in section 3.3.2.

Analytical technique

The brute-force method described above is rather computationally intensive, both in terms of memory and of computation. The number of operations to be executed may become very large, especially if we want to explore a large number of possible offsets. More specifically, the computing time and memory grow quadratically with the number of tested offsets and linearly with the number of fixations. This makes the use of this technique hard to apply in real-time situations, especially considering that eye-tracking already requires a fair amount of computational power.

We can indeed see this problem as a probability problem by looking at the fitness function as a fixation probability function. Then the goal becomes to find the offset which maximizes the probability of having the set of recorded fixations given the stimulus. By modeling the stimulus as a probability density function (PDF) with the offsets as parts of its location parameter, we can find the offset maximizing the likelihood of the fixations through a maximum likelihood estimation. Hence, in this approach, the likelihood of the fixations is taken to represent the offset fitness. Note that we consider the offset as being applied to the stimulus objects and not to the fixations themselves. However this does not change anything except the sign: a positive offset applied to the fixations corresponds to a negative offset applied to the stimulus. The difficulty is to find a good way to model the stimulus as a PDF. For stimuli composed of simple geometrical objects, a simple approach consists in decomposing them into such objects and to find a PDF describing each of these simple objects. The complete stimulus can then be modeled as a mixture model consisting of the PDFs of the different objects. We will restrict our attention to two types of objects, namely ellipses and rectangles. Ellipses may be modeled using a 2D Gaussian distribution while rectangles may be modeled using a

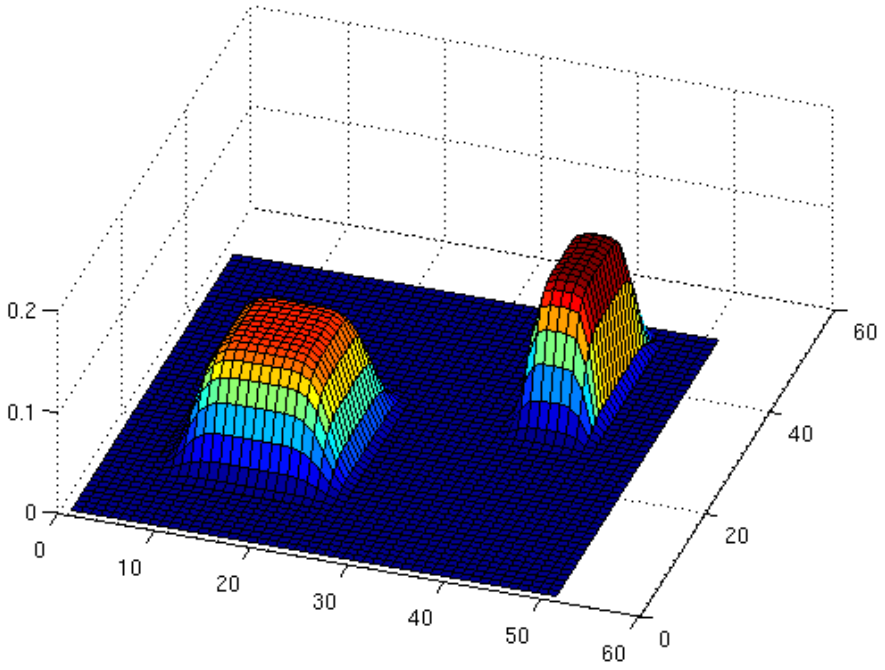


Figure 3.16: Two examples of 2D generalized Gaussian distributions with different location and scale parameters.

generalized Gaussian distribution of the form[6]:

$$G(x,y) = \frac{p^2}{4\rho_x\rho_y\Gamma\left(\frac{1}{p}\right)^2}\, e^{-\left[\left(\frac{x-\mu_x}{\rho_x}\right)^p+\left(\frac{y-\mu_y}{\rho_y}\right)^p\right]} \tag{3.4}$$

In addition to the usual location and scale parameters, μ and ρ, this distribution has a shape parameter, p, which describes the "rectangularity" of the curve. If $p = 2$ we have the simple Gaussian distribution and if $p > 2$ we obtain a "rectangular" Gaussian. As we can see on figure 3.16, this kind of PDF is a good candidate to model rectangular objects. Of course, both the Gaussian distribution for circles and the generalized Gaussian distribution for rectangles are only approximations of the actual objects as they do not have sharp borders.

Hence, we will describe rectangle objects with such a generalized Gaussian for which the mean will be equal to the center of the rectangles plus the offset value, the scale parameters ρ_x, ρ_y will be set to some factors, w_x, w_y, of the rectangle width and height and the shape parameters will be set to fixed value producing an adequate rectangular shape.

[6]Note that we consider only the simplest case in which rectangles are parallels to the axes.

Thus, for an offset (o_x, o_y), we define the PDF of a rectangle r^j having center coordinates (c_x^j, c_y^j) and size (s_x^j, s_y^j) as:

$$G_o^j(x,y) = \frac{p^2}{4w_x s_x^j w_y s_y^j \Gamma\left(\frac{1}{p}\right)^2} e^{-\left[\left(\frac{x-c_x^j+o_x}{w_x s_x^j}\right)^p + \left(\frac{y-c_y^j+o_y}{w_y s_y^j}\right)^p\right]} \tag{3.5}$$

This function has 3 parameters that need to be set: p the shape factor and w_x, w_y the multiplicative constants to obtain the scale parameters from the rectangle size. These parameters have first been estimated from qualitative tests and then have been optimized using the results from the brute force method as a baseline.

One problem which arises with this solution is with fixations that falls completely outside of the stimulus object and thus have probabilities equal to 0 because of numerical precision. Hence if such a fixation happens, the total joint probabilities of all fixations will also become 0 thus preventing any parameters optimization. A simple correction to this problem consists in adding a uniform component to the mixture. This component represents a baseline probability of looking outside of the stimulus. It this then possible either to determine its weight or to estimate it along with the offsets parameters.

We now describe the whole mixture model composed with the functions described above. Here we will consider a stimulus consisting only of rectangles but the extension to model with rectangles and ellipses is quite straightforward as simple Gaussian's are a special case of generalized Gaussian's. Let's consider a stimulus consisting of N_r rectangles. To simplify things, we can view the model as a hierarchical mixture. We have a first mixture of N_r generalized Gaussian's with weights $\alpha_0^0 ... \alpha_{N_r}^0$ and a mixture of 2 components, a uniform component with weight β and second component which is the mixture of generalized Gaussian's. Hence, the weight of the generalized Gaussian's in the global mixture are simply $\alpha_i^0(1-\beta)$.

The weights $\alpha_0^0 ... \alpha_{N_r}^0$ need to be chosen so that the probability of a fixation to be perfectly in the corresponding AOI, that is on the mode of the associated generalized Gaussian, is the same for all AOI. In other words, bigger AOI need to have bigger weights because they cover a greater surface and thus have bigger chance to be looked at. Mathematically, their weight is simply the proportion of their surface compared to the total surface covered by all AOIs. Hence, the weight for component j is:

$$\alpha_j^0 = \frac{s_x^j s_y^j}{\sum_{k=1}^{N_r} s_x^k s_y^k} \tag{3.6}$$

Thus, we have a mixture model for which the weights as well as the scale parameters of the components are known. The unknown parameter is a global parameter, corresponding to

the gaze offset we are searching for, which affects the location parameters of all components. Thus, we can find the maximum likelihood estimate for this parameter, which corresponds to search for the offset for which the joint probabilities of our fixations on the PDFs defined by the AOIs are maximum.

First, the likelihood of the offsets given the fixations $f^1, f^2 ... f^{N_f}$ is:

$$L\left(o_x, o_y | f^1, f^2 ... f^{N_f}\right) = \prod_{i=1}^{N_f} \left[\left(1-\beta\right) \left(\sum_{j=1}^{N_r} \mathbf{1}_{y_i=j} \alpha_j^0 G_j^o \left(f_x^i, f_y^i\right) \right) + \beta \mathbf{1}_{y_i=N_r+1} U \right] \tag{3.7}$$

Here, the y_i are the membership values which are the indexes of the mixture component from which the fixation f_i has been generated from. These variables are unknown which prevents a direct maximization of the likelihood. We can however use the Expectation-Maximization (EM) algorithm to solve this problem. This is a general iterative algorithm which estimates the maximum likelihood of some parameters when some variables are latent, i.e. unknown or hidden. EM requires an initial guess of the parameters of interest and improves this guess until it converges to a maximum. The persistent problem is that we have no guarantee that it will end up on the global maximum. This can be partially solved by running the algorithm from different initial values and compare the maxima found.

The algorithm alternates between expectation steps, during which the expected value of the log likelihood is computed given the current value of the parameters, and maximization steps, during which we search for the next parameters values so that the likelihood is maximized. The expectation step consists in computing partial membership value a_i^j given our current estimate of the parameter. To simplify further the equations, let's consider the following function which represents the first level of the mixture, namely the mixture of the N_r generalized Gaussian's for a current value of the offset o^t:

$$H^{o^t}\left(x, y\right) = \sum_{j=1}^{N_r} \alpha_j^0 G_j^{o^t}\left(x, y\right) \tag{3.8}$$

Then, the partial member values, $a_{i,j}$, are computed using the following formula:

$$a_{i,j}^{o^t} = \Pr\left(y_i = j \mid o^t\right) \begin{cases} \frac{\alpha_j G_j^{o^t}\left(f_x^i, f_y^i\right)}{(1-\beta) H^{o^t}\left(f_x^i, f_y^i\right) + \beta U} & \text{if } j <= N_r \\ \frac{\beta U}{(1-\beta) H^{o^t}\left(f_x^i, f_y^i\right) + \beta U} & \text{otherwise} \end{cases} \tag{3.9}$$

From equations 3.5, 3.7 and these partial membership values, we can compute the expectation of the log likelihood, given our current estimate o^t:

$$\begin{aligned} Q(o \mid o^t) = &\mathbb{E}(\log L) = \sum_{i=1}^{N_f} \left(\sum_{j=1}^{N_r} a_{i,j}^{o^t} \Bigg[\log\left(\alpha_j^0 (1-\beta)\right) + 2\log(p) - \right. \\ &\log\left(4 w_x s_x^j w_y s_y^j\right) - 2\log\left(\Gamma\left(\frac{1}{p}\right)\right) - \left(\frac{\mu_x^j - c_x^j + o_x}{w_x s_x^j}\right)^p - \left(\frac{\mu_y^j - c_y^j + o_y}{w_y s_y^j}\right)^p \Bigg] \Bigg) + \\ &a_{i,N_r+1}^{o^t} \left(\log(U) + \log(\beta)\right) \end{aligned} \tag{3.10}$$

The maximization step consists in computing the next value of the parameter, o^{t+1} which maximizes this function:

$$o^{t+1} = \underset{o}{\operatorname{argmax}} \left(Q\left(o \mid o^t\right)\right) \tag{3.11}$$

Hence, we have to compute the roots of the partial derivatives of Q with respect to our parameters o_x, o_y, which gives us the following equations:

$$\begin{aligned} \frac{\partial Q(o|o^t)}{\partial o_x} = 0 \leftrightarrow \sum_{i=1}^{N_f} \sum_{j=1}^{N_r} a_{i,j}^{o^t} \frac{p}{w_x s_x^j} \left(\frac{\mu_x^j - c_x^j + o_x}{w_x s_x^j}\right)^{p-1} = 0 \\ \frac{\partial Q(o|o^t)}{\partial o_y} = 0 \leftrightarrow \sum_{i=1}^{N_f} \sum_{j=1}^{N_r} a_{i,j}^{o^t} \frac{p}{w_y s_y^j} \left(\frac{\mu_y^j - c_y^j + o_y}{w_y s_y^j}\right)^{p-1} = 0 \end{aligned} \tag{3.12}$$

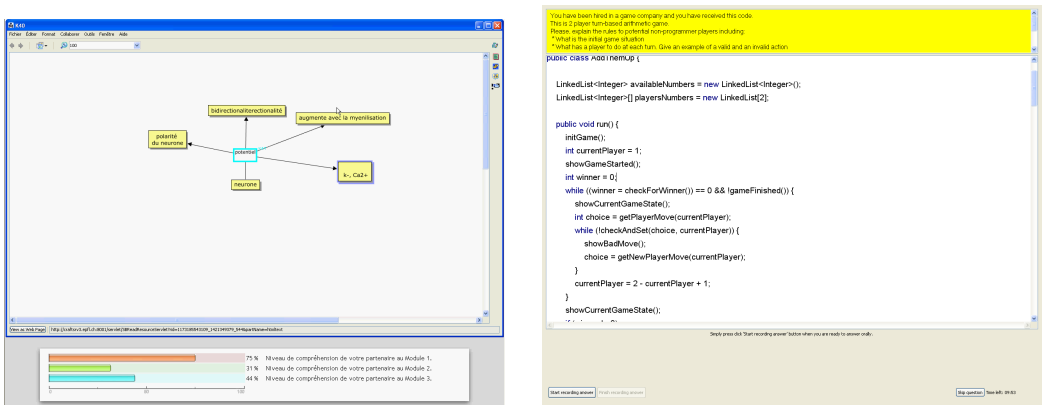
Considering that these equations are not solvable analytically, we have to use numerical methods, such as Newton's method, to find its root. Then, we start again the expectation step until our estimate converges. Alternatively, we can also directly apply a gradient descent on Q. Finally, we can also consider the weight of the uniform component, β, as a parameter. In this case, we can also compute its next value with the following direct formula:

$$\frac{\partial Q\left(o \mid o^t\right)}{\partial \beta} = 0 \leftrightarrow \beta = \frac{\sum_{i=1}^{N_f} a_{i,N_r+1}^{o^t}}{\left(\sum_{i=1}^{N_f} a_{i,N_r+1}^{o^t}\right) + \left(\sum_{i=1}^{N_f} \sum_{j=1}^{N_r} a_{i,j}^{o^t}\right)} \tag{3.13}$$

In the next section, we compare the results of this analytical method with the brute-force method, which is considered as a baseline.

3.3.2 Correction results

Assessing the quality of the estimated offsets is not easy. Indeed, we have no baseline to which we can compare the results. A solution would be to have experts who manually code to which AOI each fixation belongs to in order to compare with the results found using the best offset found. However it is very questionable whether an expert can really do such a coding, especially when the offset is big. In any case, as we will see with the results from the brute-force method, the correction is good because the fitness score of the best offset is generally much higher than for the null offset, which corresponds to no correction. This indicates that the corrected fixations are more on the objects of interest than the uncorrected ones. For some experiments, this improvement is dramatic and it is clear that the corrected fixations reflect much more the actual fixations.



(a) Concept-map building

(b) Program understanding

Figure 3.17: Screenshots of the two experiments that were used for the analysis of systematic errors correction method.

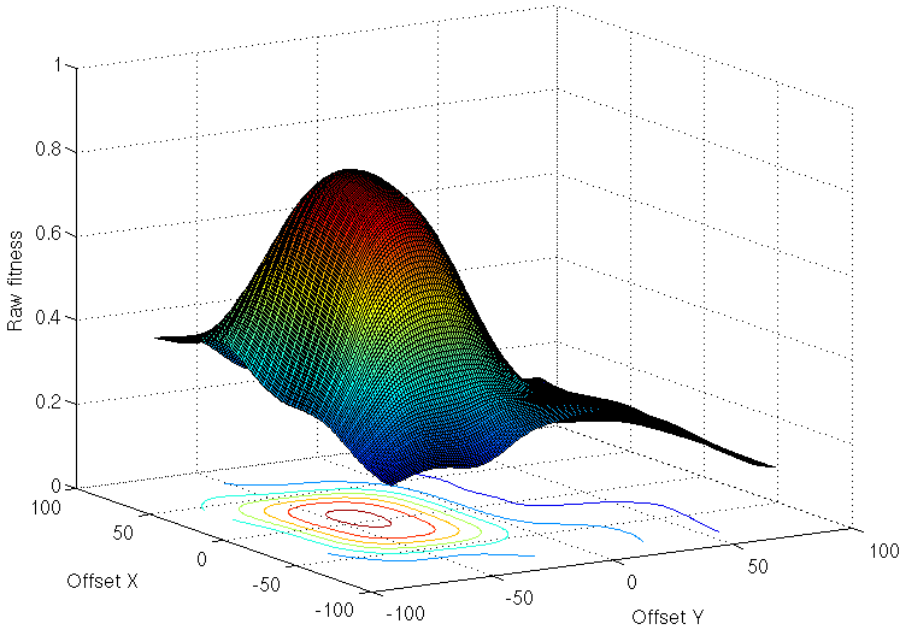
Both the analytical and brute-force methods have been ran on 220 subjects from two different experiments. The first experiment (50 subjects), called KAT experiment, is a collaborative concept-map building tasks (see section 5.2.2) in which the stimulus consisted of a set of labeled boxes with labeled links between them (see figure 3.17a). The second experiment (170 subjects), called GREPP experiment (see chapter 6), was a program understanding task for which the stimulus consisted essentially of JAVA code and English text (see figure 3.17b). In both cases, the stimulus was dynamic as the subjects could scroll the screen and even create objects in the concept-map task. The first experiment lasted 20 minutes and the second one lasted around 50 minutes. It is important to note that in the second experiment, GREPP, subjects were using a head-rest which prevented them to move their head too much. In the KAT experiment, they were free to move as they want.

Brute-force technique

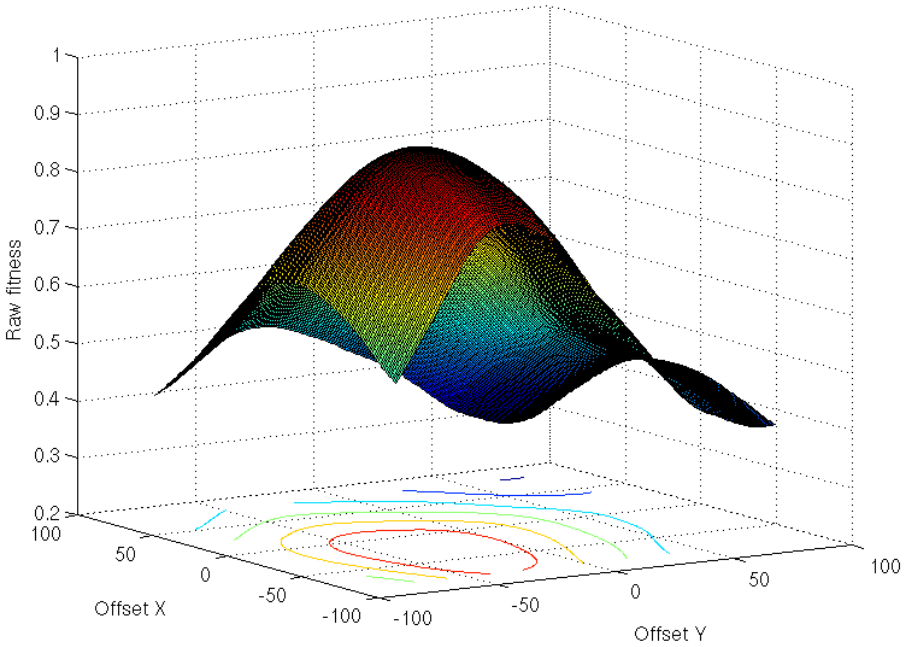
In order to assess the quality of the correction, we can look at the resulting fitness values in the offset space. The existence of a fitness peak in this landscape shows that there exists an optimal offset for which the fixations fall into AOIs with a higher probability. Also the fact that this peak is not necessarily located on the origin of the offset space, which corresponds to a null offset, indicates that there is a systematic trend in the errors. Indeed, if the errors were not systematic and centered around zero the peak would be centered on zero. On figure 3.18a we can see the average fitness of all fixations for one subject in the concept-map experiment. We see clearly that there is a well-defined peak which is not located on the origin. More specifically, the fitness is highest for a horizontal offset of 8 pixels and a vertical offset of -40 pixels and has a value of 0.82 compared to 0.33 at the origin. Considering that in the first approximation the fitness value is 1 for fixations inside AOIs and 0 for fixations outside AOIs, we can interpret these values as the percentage of fixations inside AOIs. Hence, it is a huge improvement between the raw fixations and the corrected fixations with 50% more fixations inside AOIs. The resulting fitness for the subject of the GREPP experiment (see figure 3.18b) is less explicit but we still have a good peak with a maximum located at -28 pixels for the horizontal offset and -30 pixels for the vertical offset which results in a fitness improvement from 0.77 to 0.90.

In order to quantify the general improvements that this correction brings about, we looked at the distribution of fitness improvements over all subjects in both experiments (see figure 3.19). We see that in many cases (121 subjects), the improvement is quite small, less than 10% more fixations inside AOIs. However, there are still 45% subjects which have more than 10% more fixations inside AOIs and among these, 13% have more than 20% more fixations inside AOIs. Moreover, it should be noted that we consider only the fact that some fixations fall completely outside any AOI if there are not corrected but we do not consider fixations that are assigned to the wrong AOI. Indeed, it is likely that the actual improvement is actually bigger because the correction can also have a positive effect on fixations which are already inside AOIs before the correction. It is especially the case in the GREPP experiment as the AOIs (lines of code) are dense and close to each other. However, this is much more difficult to assess as we don't have any baseline to compare with.

In order to assess whether the method works well we can compare it with the results found when it is applied on random gazes. Hence we have run the same correction on the same data by simply replacing each gaze location by a random position uniformly distributed on the whole screen area. In this way we keep the same stimulus, hence the same fitness function, and we can check whether the increase of the maximum fitness value is not caused by the shape of the stimulus. If we look at the resulting fitness in the offset space for the two same subjects than on figure 3.18, we can see that, when using random fixation locations, there is no clear fitness peak (see figure 3.20). Also, on figure 3.21, we can see how the offset variance, computed by considering the fitness values as probabilities, changes for real and random data when we consider more and more fixations. A first observation is that in both experiments the



(a) Subject from KAT experiment



(b) Subject from GREPP experiment

Figure 3.18: Resulting fitness values in the offset space for one subject of the KAT experiment and one subject of the GREPP experiment

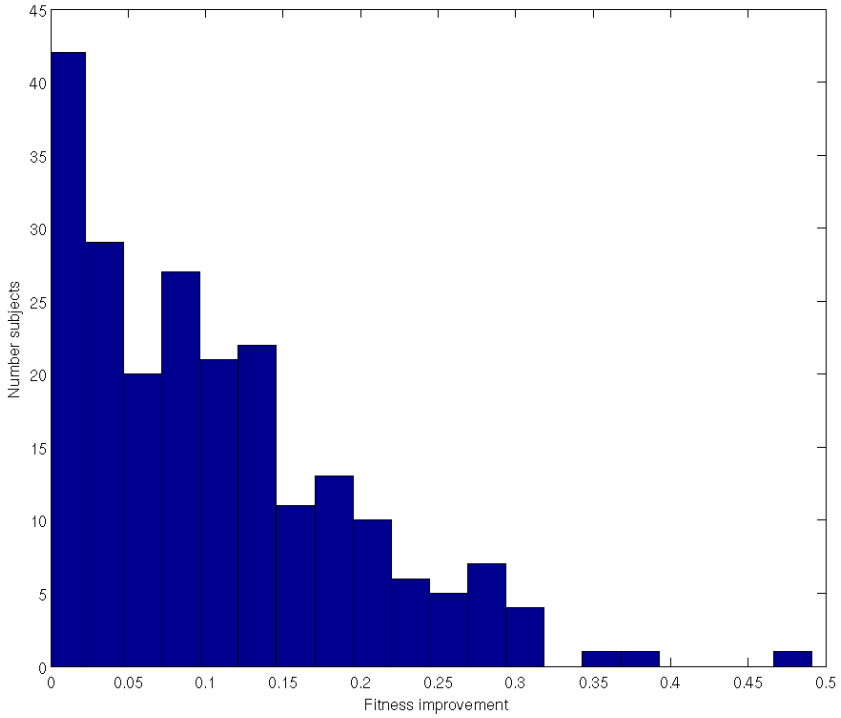
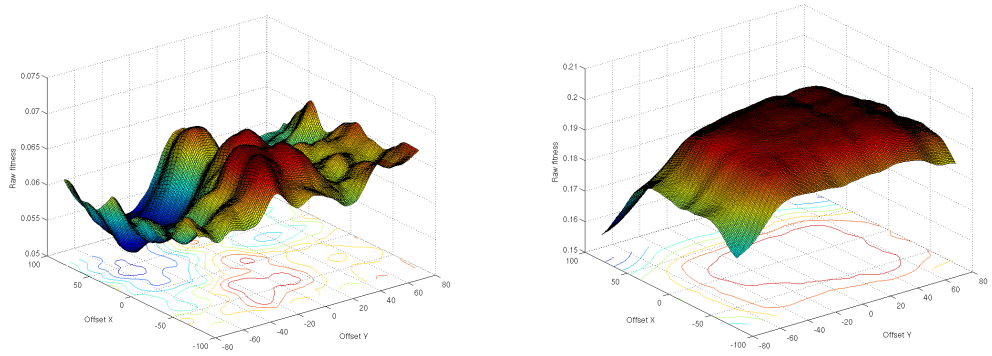


Figure 3.19: Histograms of the fitness difference between uncorrected and corrected fixations for all subjects in both experiments

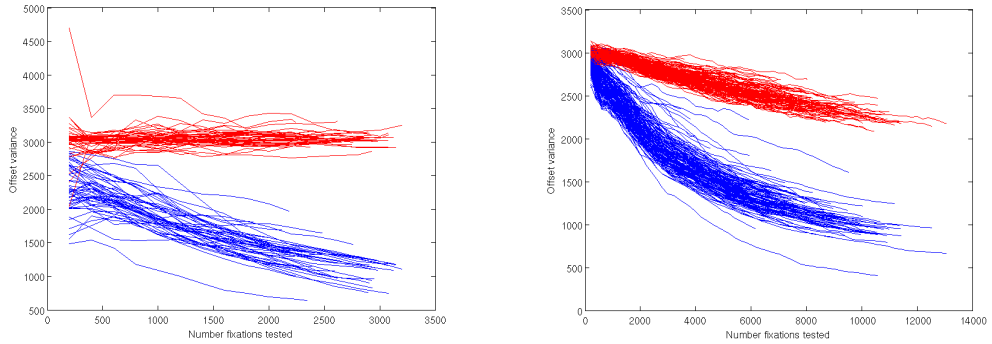
offset variance with real data (blue curves) decreases as we test more fixations which means that we become more and more confident in the offset. This is another evidence which shows that the offset found is correct. Also, the variance becomes lower than with random data (red curves) as soon as we consider enough fixations. However, we can note that the required number of fixations for the real cases to have a lesser variance than the random cases is much higher in the case of the program understanding task (note the difference of the horizontal scale). The second observation is that in the case of the program understanding task, the variance decreases also with random fixations, although much less than with real data. The best explanation for this is a bias in the stimulus structure. Indeed, there is both a horizontal bias, because there is always some texts on the left part of the screen but not always on the right part, and a vertical bias, because the bottom of the screen contains relatively few AOIs compared to the upper part which contains the text. Hence, uniformly distributed random gazes will tend to be slightly shifted towards the upper-left corner of the screen. This is also confirmed by the shape of the resulting fitness values for the example shown on figure 3.20b.

Finally, it is worth looking at the distribution of all the offsets found among all our tested subjects. The histograms of the offset values are shown independently for each experiment and for the two components on figure 3.22. Interestingly, we can see that there is a trend in



(a) KAT experiment (b) GREPP experiment

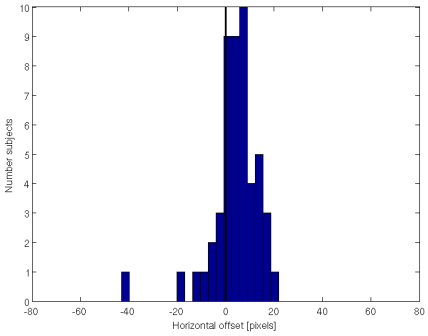
Figure 3.20: Resulting fitness values, when using random fixation locations, in the offset space for one subject of the KAT experiment and one subject of the GREPP experiment.



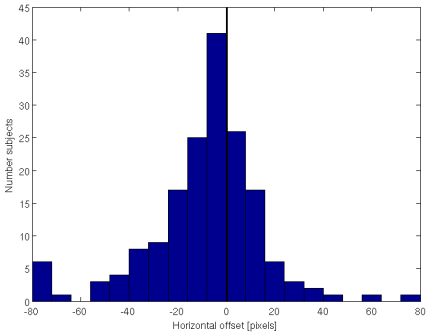
(a) KAT experiment (b) GREPP experiment

Figure 3.21: Evolution of the offset variance with the number of tested fixations. Blue curves are real data and red curve are random fixations applied on the same stimulus.

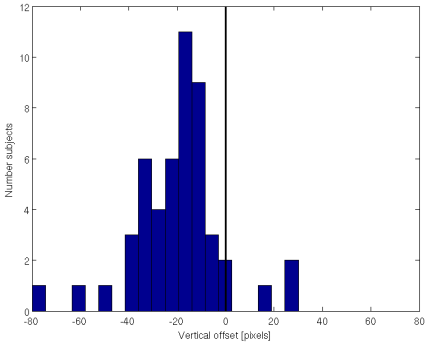
the distribution of the vertical offset (figures 3.22d and 3.22c) which is very often lower than 0 with a mode around -20 pixels. This means that in general data recorded with this eye-tracker are shifted upward compared to real eye-locations. In comparison, there is no such trend in the horizontal offsets (figures 3.22b and 3.22a) which have distributions centered on zero. Secondly, we can also note that the vertical offset, in absolute value, is also generally greater than the horizontal one and it can be as large as 80 pixels, which represents almost 10% of the screen size. The fact that we have a great part of the subjects who have 20 pixels or more of vertical offset may have dramatic consequences if it is not corrected. Indeed, if we consider for example the GREPP experiment which consists of text, the distance between the centers of 2 lines is 30 pixels. Hence, it is clear that for many subjects, there is a great chance that the fixations will be associated to the wrong line, which could then change completely the result of subsequent analyses about how programming code is read.



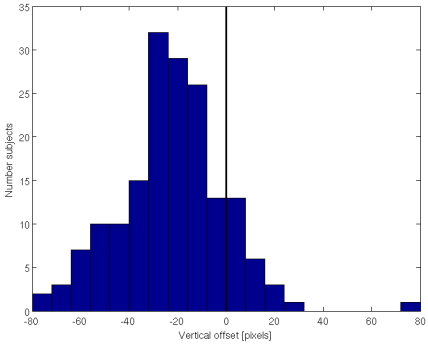
(a) Horizontal offset in KAT experiment



(b) Horizontal offset in GREPP experiment



(c) Vertical offset in KAT experiment

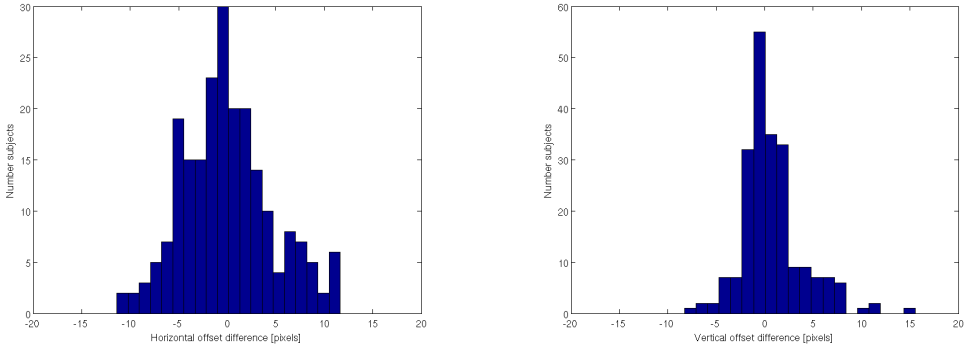


(d) Vertical offset in GREPP experiment

Figure 3.22: Histograms of the offsets found for all subjects of both experiments.

Analytical technique

To assess whether the analytical method works well, we ran it on our whole set of 220 subjects and we compared the resulting offset found with the one found with the brute-force method. For almost every subjects (217 subjects), the differences in the offset values lie between -20 pixels and + 20 pixels. On figure 3.23, you can see the distribution of these differences. For the vertical offset (figure 3.23b), almost all subjects have differences lower than 10 pixels and the distribution is centered on 0 and quite sharp. Hence, for most of the subjects, the analytical method found the same result as the brute-force approach with maximum 5 pixels of difference in one way or the other. Note that it doesn't necessarily mean that the analytical method is wrong by 5 pixels. Indeed, it could well be the case that the results of the analytical methods are more correct but we cannot say something about that as we don't have any baseline to compare to. The distribution of the differences for the horizontal offset is a bit more diffuse than for the vertical offset, with more subjects having differences. However, the differences are still quite low. Hence, we can say that the analytical method is in good agreement with the brute-force method and thus, it could be used to estimate offset with less computation.



(a) Horizontal offset difference

(b) Vertical offset difference

Figure 3.23: Histograms of the differences between the brute-force and the analytical method for the horizontal and vertical offsets.

3.3.3 Discussion

The results of the brute-force systematic error correction indicate that in many cases, there is a systematic accuracy error in the gaze data recorded by the eye-tracker we used, the Tobii 1750. While we cannot really assess directly the quality of the correction because we have no baseline to compare our data with, we still have several clues that indicate that we are really improving the accuracy of the data. First, the assumption that most fixations must fall on AOIs is well motivated by the fact that, in order to perform the task correctly, which is what almost all subjects did, it is necessary to look at those objects. Secondly, even if we relax this assumption and assume that some subjects looked a lot outside of the AOIs, in the blank regions, then we should not find any fitness peak in the data. Indeed, in such a case, the fixations would be randomly distributed around the AOIs and thus no peak would appears as each offset would move some of the fixations into AOIs while leaving the majority of fixations outside. This is confirmed by applying the method on randomly distributed fixations which shows that in such a situation, no peak appear and the confidence we have in the offset measure, i.e. the offset variance remains high.

However, this brute-force correction method suffers from its computational complexity. It requires a lot of computational power and memory and these requirements increase if we want to cover a bigger offset space or with a finer grain. The analytical solution overcomes these limitations and is able to estimate offsets faster and using less memory. By using the results of the brute-force method, we have been able to set the unknown parameters required for this analytical method and in its final form, it yields results very close to those of the brute-force method for most of the subjects. This analytical method has also the great advantage that it can be easily extended to more complex offset models in which the offset is not constant but depends on space and/or the time. Indeed, with such models, there would be more than 2 parameters which would make the brute-force method difficult and expensive to apply. On the opposite, the analytical solution could still be applied by simply estimating more parameters.

The existence of these offsets in the recorded gaze data may be a great problem if they are not corrected. Indeed, as we have seen, more than half of the subjects are affected by such problem in a relatively important way with offsets reaching 80 pixels. This can affect in a significant way subsequent analyses by changing the objects to which fixations are associated. An open question that we have not been able to test is whether these systematic errors are related to bad calibrations. However, previous works Hornof and Halverson (2002) tend to show that it is not the case, or at least not completely. Finally, it seems also that these errors are not due to head movements as in one of the two experiments, subjects were using a head-rest and still have important offsets.

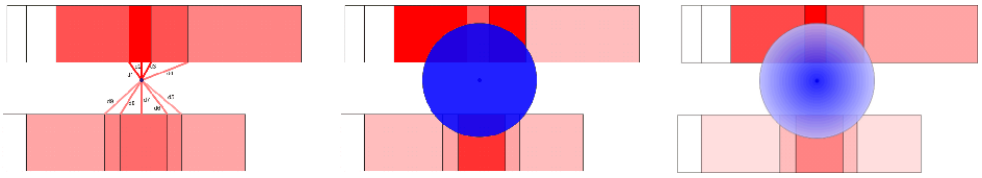
3.4 Hit detection

Hit detection is the process of associating each fixation to an object of the stimulus. Indeed, to perform meaningful analyses of gaze patterns in relation with a stimulus, it is necessary to link fixations with the stimulus. The usual solution to this problem consists in associating each fixation with the AOI containing it or with the closest AOI if the fixation is not contained within an AOI. This is a straightforward and intuitive solution. However, there are several potential drawbacks with this method. First, the decision to associate a fixation to a given AOI or to another close AOI is abrupt and may change completely for a difference of 1 pixel which makes sometimes the decision arbitrary. Secondly, it doesn't take into account the eye-tracker non-systematic accuracy errors which causes that the closest AOI is not necessarily the correct one. Finally, it associates each fixation to a single object which may be incorrect as it is clearly possible to see several objects with a single fixation.

Actually, these various biases are modulated by the size and the arrangement of the AOIs under consideration. Indeed, if the AOIs are big and distant from each other, there are very few chances that a fixation falls within the limit of equidistance between 2 areas. Similarly, it is quite unlikely that 2 areas are perceived within one fixation. On the opposite, if there are many small and close AOIs, then all the problems described above are likely to affect the results found using a simple solution as described above.

3.4.1 Probabilistic hits model

We propose another solution which aims to solve these issues in complex situations with many objects by assigning probabilities instead of making sharp associations. More specifically, this consists in attributing for each fixation a probability of looking at each object depending on the relative position of the fixation and the object. In other words, we associate to each fixation a Probability Mass Function (PMF) over the domain of all AOIs. Of course, generally, only few AOIs, that are close enough to the fixation, will have a non-null probability. To do so, we need to define how these probabilities are computed. The general idea is to assign probability values proportionally to the closeness of the corresponding AOI to the fixation. Then, the question is which closeness measure we should use. We propose three different possibilities to



(a) Distance-based (b) Area-based (c) Gaussian-based

Figure 3.24: Different methods to compute hit probabilities of a fixation

compute the closeness of a fixation to an AOI. Figure 3.24 summarizes these different methods for a schematic situation with several AOIs of various sizes and an ambiguous fixation.

Distance-based

A simple method (figure 3.24a) consists in taking the shortest distance between the fixation and the AOI. The drawbacks of such a method is first that it depends on the granularity of the AOI segmentation and secondly that the shortest distance does not necessarily reflect correctly the closeness of a gaze to the AOI. Indeed, if we have a fixation close to the corners of a large AOI and also of a small AOI (such as the two top-left hit areas on figure 3.24a), they will be considered as being at the same distance, which seems intuitively wrong. Indeed, most parts of the large area are quite far from the fixation while this is not the case for the small area. Hence, it seems incorrect to consider them as being at the same distance. A possible correction would be to use the distance to the center of the AOI but then, the problem is that it is really meaningful only if we have circle-like AOI because otherwise, it doesn't always reflect an actual distance to the AOI (consider for example flat rectangular AOIs).

Area-based

A more elaborated method (figure 3.24b) is to use the area of the intersection between a circle centered on the fixation and the AOI. This is a better closeness measure because it measure which amount of the AOI is at a certain distance from the fixation. However, it should be noted that this value is affected by the size of the AOI in the sense that the maximum closeness value that an AOI could have is equal to its area. It is thus possible to normalize it with the size of the AOI which corresponds to consider which portion of the AOI is covered by the fixation area.

Volume-based

The area-based solution consists in computing the area or the portion of the AOI that is close enough to the fixation. A more refined solution is to compute instead the average distance of the AOI to the fixation. An easy way to do this is to consider a volume, such as a cone, centered on the fixation and to compute the volume over the AOI. This corresponds to assign lower values to the parts of the AOI which are further from the fixation. We can even consider a 2

dimensional Gaussian curve as the volume shape (figure 3.24c). This is a way to model the unsystematic accuracy errors of the eye-tracker by assuming that these errors are normally distributed and centered on 0. Finally, note that similarly to the area-based solution, the resulting value is bounded by the size of the area and thus, it is also possible to normalize it with the AOI surface.

Probabilities assignation

Once we have chosen a measure of closeness, we can use it to assign probabilities to each AOI surrounding a fixation. It can be done in different manners and the way we do it depends on the interpretation of the probabilities we want to make. Actually, there are two ways to interpret the resulting probabilities, either in terms of probability that the real fixation was on the AOI considering that the eye-tracker have some unsystematic errors, or in terms of the amount of visual attention that is attributed to the AOI if we consider the fact we can look at more than one object with one fixation. The choices we have to do about the closeness measure and the way we assign probabilities are not the same whether we want to interpret the result in one way or the other. In the first case, i.e. to model eye-tracker errors, the Gaussian volume closeness measure is clearly the most adapted but for the second case, to model visual span, it depends on whether we think that an object closer to the fixation center is more "seen" than an object further from it. If we assume that it is the case, then a cone or Gaussian volume-based is a good measure but if we assume that all objects within a certain field are all seen as much, then we have to use the area-based measure. The normalization of the values with the AOI size makes sense only if we consider the interpretation of the values as an amount of visual attention. It may then be done to reflect the fact that smaller AOIs need less visual attention than bigger ones. Alternatively, it can be considered later on in the analyses by interpreting the assigned valued in accordance with the AOI size.

Finally, the actual values used as probabilities may be either the closeness measure per se or it can be the closeness measure normalized so that the values of all AOIs for one fixation sum to 1. In the case of an interpretation as modeling the eye-tracker error, this normalization has the effect of considering that a fixation is necessarily on an AOI and has no chance to be outside any AOI. One alternative is to do this normalization by first checking that the total values exceeds a certain threshold and otherwise to assign 0 for all AOIs. Thereby, we consider only cases where the fixation covers enough AOIs area. In the case of modeling the visual span, the meaning of such a post-normalization is more subtle and would reflect that each fixation always brings the same total amount of visual attention versus a situation where the total amount brought by one fixation is smaller when the fixation is further from the AOIs.

3.4.2 Analyses with probabilistic hits

The use of probabilistic hits influences the way subsequent analyses are done. Indeed, instead of simply considering sequences of fixated AOIs, we now have to consider sequences of PMF

over the AOIs. However, it is possible to take this problem back to the simple problem of AOIs sequences. Indeed, from these probabilistic sequences, we can compute all the possible AOIs sequences along with their corresponding probability. From that, we can then compute the expected value of any indicator computed on a single sequence by taking the average of the indicators over all sequences weighted by the probability of the sequences. In practice, this is difficult to accomplish because the number of possible sequences grows exponentially with the length of the sequences. So, instead of computing the exact expected value, we can estimate it by sampling the possible sequences. This corresponds to draw a sufficiently large number of sequences, let's say 1000, by choosing each elements of the sequence randomly according to the PMFs defined for each fixation. Then, we simply compute the average of the indicator over the sampled sequences. Note also that we can also do simpler computation by considering only the most probable sequence of AOI. This would then be similar to using a simple single-object hit detection, except that we use a better closeness measure than the usual closest distance measure.

Finally, it is also possible to use directly the probabilistic values for the computation of some indicators. This latter solution is more adapted when we consider that these probabilities reflect the amount of visual attention. This consists in considering that these values represent the density of gazes over the AOIs and we can use these values as such to compute specific indicators. A simple example is to aggregate the values over several fixations, generally by weighting them with the fixation duration, to get a meaningful gaze density over some period of time. From such a gaze density, we can compute for example dispersion values.

3.4.3 Model chosen

We used probabilistic hits to analyze data from our main experiment (see chapter 6). We chose to use a volume-based model to compute closeness between fixations and AOIs. More specifically, we used a 2 dimensional isotropic Gaussian to model the fixation volume. The standard deviation of this Gaussian was set to 20 pixels. This value was chosen in accordance with the accuracy error claimed by the eye-tracker manufacturer. Indeed, the eye-tracker has an accuracy of ± 5 deg which corresponds to about 1 cm (around 30 to 40 pixels) for a head distance of 50 cm. Hence, 95% of the fixation probability is distributed in the range of the eye-tracker error (40 pixels). The closeness values were normalized so that the probabilities related to a specific fixation sum to one.

3.4.4 Discussion

We have seen that the usual solution for hit detection is not adapted when the AOIs are numerous, small and close to each other because it is biased, unrealistic and doesn't consider the presence of accuracy errors. Thus, we propose a new probabilistic model for hit detection which consists in assigning a PMF over all AOIs to each fixation for which the values are proportional to the closeness of the AOI to the fixation. This PMF may reflect two different

aspects. First, it may be used to model eye-tracker accuracy errors by providing the probability that the actual fixation is on an AOI. Secondly, it may indicate how the visual attention provided by a fixation is distributed over several objects, to model the fact that the visual span is not limited to a single object. The choices on how to compute exactly these probabilistic hits vary depending on which aspects we want to model. The drawbacks of the model presented here is that it is difficult to combine these two aspects together and still be able to interpret the values. A solution would consist in modeling separately these two aspects by having a two-step procedure. Indeed, we could first assign probabilities modeling the eye-tracker accuracy errors and then, a second step modeling the visual span. However, it is not clear how we should proceed exactly and more work would be required.

In the present work, we decided to use probabilistic hits to model the eye-tracker errors because in this case, the meaning of the probabilities is well defined. Also, subsequent analyses can easily be done by sampling the resulting PMFs sequences as explained above.

3.5 Discussion

Eye-trackers are affected mainly by two types of errors: precision and accuracy. Precision, which refers to the amount of variations in the measure of a same gaze location, is likely to affect the fixation identification process. Indeed, it should be taken into account when choosing the threshold used to detect stillness. With the eye-tracker used in this study, the Tobii1750, it appeared that the precision level may vary greatly between trials. A part of this variation is due to the variation of head-screen distance which tends to increase the noise when the head is closer or further from the optimal distance. As we have seen, these variations of precision are likely to affect in a significant way how the fixations are detected. If they are not taken into account, it is likely that we end up with subjects having too much or too few, too long or too short fixations. Moreover, if we consider previous works (Karsh and Breitenbach, 1983; Shic et al., 2008a; Blignaut and Beelders, 2009), we see that this can have a great importance for subsequent analyses. Hence, in order to deal with such noise-variable data, it is necessary to use an adaptive algorithm, such as the one proposed by Nyström and Holmqvist (2010), that will estimate this noise level and adapt the detection process accordingly. We also offer such an algorithm which is based on the idea that saccades occurs rarely and are outliers in terms of gaze velocity. Our solution has the advantage of running in a single pass which makes it usable for real-time application. Moreover, it also estimates a threshold that can potentially vary over time, which is useful if we consider the fact that head movements can affect the noise level. The originality of our approach resides in the fact that these analyses, and in particular the optimization of the adaptive algorithm parameters, have been performed through the use of a metric to measure the quality of the fixation identification process. Although this metric could certainly be improved to better reflect a good identification process, we think that this is a very promising approach to these issues as it allows us to rely entirely on objective criteria.

The second type of problem, accuracy errors, refers to the fact that the measured gaze location

may be shifted from the actual gaze location. Such errors are likely to affect analyses that relate fixations with stimulus objects. If such errors are unsystematic, i.e. centered and randomly distributed, this will simply increase the noise in those gaze-stimulus relationships. But, if there are some systematic trends in these accuracy errors, this could bias completely those relationships by assigning systematically the incorrect AOI to each fixation. Qualitative analyses of our data reveal that, in many cases, recorded gaze locations are affected by such systematic accuracy errors and often in an important way. Hence, it is necessary to correct in some ways these systematic errors, for example by using a specific calibration-like stimulus dedicated to the measure of these errors as it is done by Frank et al. (2011). In our case, we used an approach similar to Hornof and Halverson (2002). We developed a method to automatically correct such errors based solely on natural eye-movements over the main stimulus, by considering the AOIs as representing zones of high fixation probability compared to blank regions of the stimulus. The results show unambiguously the effectiveness and the usefulness of this automatic correction. Finally, we developed an analytical method to make this correction in an efficient way, so that it can be used in real-time applications. More importantly, this latter method can also be used to estimate non-constant offsets that could possibly vary across the screen and possibly across the time. However, while there is a lot of evidence that actual offsets have such dependence on time and especially on space, the exact form of the relation is difficult to know and thus more work is required on this aspect.

Finally, if we consider that systematic errors have been corrected, the last problem to cope with are the unsystematic accuracy errors which causes the recorded fixations to be shifted by some amount in a random direction from their actual location. In order to deal with this issue, we propose a novel solution to detect the correspondence between fixations and stimulus objects. Instead of applying a sharp decision which associates uniquely each fixation to a single object, we use a probabilistic approach which assigns to each fixation a probability distribution over all the objects of the stimulus. This method considers that the accuracy errors are centered and normally distributed. With such an assumption, it is not difficult to compute the probability that a fixation is on a given AOI given its recorded location. It consists in computing the integral, bounded by the AOI borders, of a 2D Gaussian centered on the fixation location. It is then possible to compute expected value of gaze indicators by sampling the resulting probabilistic sequences. Finally, this probabilistic framework for hit detection can also be used to model instead the fact that the visual span of one fixation is not limited to a single object but can spread across several objects. However, in this latter situation, the choices on how the probabilities must be computed and how they must be interpreted are difficult to do and often more subjective.

Figure 3.25 summarizes these different pre-processing steps and how they aims to cope with the various kinds of problems that affect eye-tracking data. Simple processing offered by eye-tracker software generally does not take into account these errors. We think that it is an important issue because, as we have shown, this can greatly affect eye-tracking data at various levels. Hence, if we aim to get robust and generalizable results out of eye-movements data, we absolutely need to apply such kind of corrections. The present work first points out

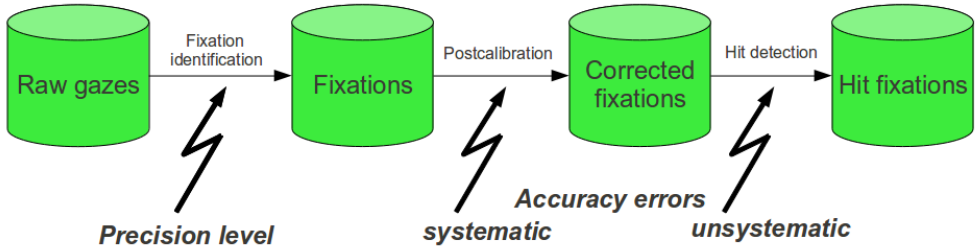


Figure 3.25: Eye gaze data preprocessing and how they aim to cope with the different types of eye-tracker errors.

and quantify these problems and secondly brings some tentative solutions. Although, these solutions are not perfect, they already make a step towards better eye-movements analyses.

4 Dual eye-tracking methodology

This chapter concerns the methodology specific to the use of two eye-trackers synchronously to study the interaction between two persons. We refer to such a methodology as dual eye-tracking. We first briefly discuss the motivations for the use of two synchronized eye-trackers and what are the challenges that come into play. We continue with the presentation of the technical issues related to this technique and of the solutions that were adopted. Finally, we present, cross-recurrence analysis, a specific type of analysis that can be conducted with dual eye-tracking data. We explain how it is performed and we show existing results that have been found using such analyses. Finally, we present problems that may arise with cross-recurrence analysis in real collaborative situations and we offer tentative solutions for these issues.

4.1 Introduction

Dual eye-tracking consists in using two eye-trackers working synchronously to study the eye-movements of two interacting persons. This novel technique is of a great interest for the study of collaboration and interaction in general. Indeed, it allows researchers to get an insight about the cognitive processes engaged during social interaction. We need to specify that in this study we focus on people working on computers remotely through a shared application. Hence, we do not study face-to-face interaction. This means that we are interested in analyzing eye-movements of collaborators on their object of interest, and not the potential mutual gaze, i.e. gazing at the other's gaze, that is routine in face to face situations. Moreover, our goal is to study collaboration in ecologically valid situations, i.e. in tasks that are the closest possible to real tasks, which contrast with simple, very controlled task.

The motivation for using dual eye-tracking methodology is twofold. On the theoretical side, it provides a novel way to get insights about the socio-cognitive processes of collaboration. Having access to eye-movements of collaborators, generally with other behavioral data such as speech and actions, provides an opportunity to model what is going on in the mind of collaborators. On a more applied aspect, having two eye-tracker connected through a shared application could allow the development of gaze-aware applications. Such applications could

use the eye-movements of collaborators to make inferences about the ongoing collaborative activities and provide meaningful information to the collaborators.

However, the use of such a methodology raises several issues. The first challenge is technical. Indeed, our goal is to get two synchronous streams of gaze data which represent the mutual eye-movements of two collaborators working on some shared stimulus. This is technically challenging because eye-trackers, or at least the ones used in this study, are not designed to be used in such situation and thus, do not offer simple mechanisms to achieve temporal synchronization between them. We experimented different implementations to fulfill this goal and this is the topic of the first section. The other challenge is the analysis of dual-gaze data. Indeed, we need a good way to characterize the patterns between the eye-movements of two people working on the same stimulus. One very promising type of analysis is cross-recurrence analysis introduced, in the context of gaze studies, by Richardson and Dale (2005). However, as we will show, several issues occur when using such an analysis in the context of real collaboration. The second section is thus dedicated to the presentation of cross-recurrence analysis in the context of real collaborative problem-solving activities.

4.2 Technical development

The most important technical issue regarding dual eye-tracking studies is the time synchronization of the various streams of data logged during the experiments. More specifically, the difficulty is the synchronization of the gaze data of the two eye-trackers together as well as the other data, generally speech and actions data. These logs are often produced by three different sources. The two streams of gaze come from the two eye-trackers' operating software while speech and actions are recorded by the shared application used by the subjects to collaborate. The Tobii 1750 eye-trackers used in these studies do not offer any sort of mechanisms to be used in a dual eye-tracking setup. Therefore, we experimented with different technical setups to use them synchronously. A second issue that may arise in dual eye-tracking studies is the space synchronization. Indeed, depending on the type of shared application that is used, it may be possible that the collaborators do not always have the exact same view of the shared workspace. For example, the workspace may consist of an area bigger than the screen size and the collaborators could possibly move their view in this workspace independently from each other. In such a situation, it may be necessary to synchronize the gaze positions so that they become comparable. This is accomplished by converting gaze positions on screen into gaze positions in the workspace coordinates reference system.

In this section, we first briefly describe the different possibilities offered to operate the eye-tracker. We then present the various technical setups we implemented with an explanation of how time synchronization is accomplished in these various setups.

4.2.1 Eye-tracker operation modes

In order to be used, Tobii 1750 eye-trackers require a proprietary software, called TET server, to be installed on the computer to which they are connected. This software acts as a driver for the eye-tracker, offering eye-tracking service to some client application that will be responsible for collecting gaze data. Thus, the actual eye-tracking process is managed by another software that must connect itself to this TET server in order to use the eye-tracker. Different options are offered for this client application.

The easiest way to operate this kind of eye-trackers is to use the software provided by the manufacturer, ClearView. This software allows to define stimuli of different sort, such as slide shows, web pages or movie, and to record gaze data synchronized with the defined stimulus. For example with a slide-show stimulus, the software can provide the times of slide changes synchronized with the gaze time so that it is possible to know exactly which gazes were made on a given slide. It also allows us to use custom stimuli, i.e. to access any application during the gaze recording, in which case there is no synchronization between the stimulus and the gaze data. However, in all cases, the software also logs every input events, i.e. keyboard and mouse clicks, disregarding the type of stimulus used. Moreover, it also captures a video of the screen, as well as of the sound input, during the whole recording. These additional logs (input events, screen and sound capture) are all synchronized with the gaze data. In particular, the video and sound recordings start exactly at the time of the gaze recordings. As we will see in the next section, these logs are opportunities to synchronize external data with the gaze data. Unfortunately, it appeared that these additional logs were actually not so well synchronized than what is claimed. More specifically, the video was not really perfectly synchronized and this resulted in some cases in differences of several seconds over an experiment of 20 minutes, which is huge considering the time-scale of gaze data.

The eye-trackers can also be operated in two different ways through the use of programming APIs provided by the manufacturer. A first API, the high-level API, allows us to control the Clearview software from an external application. In particular, the recording can be started automatically. The second API, the low-level API, allows us to control entirely the eye-tracking process from an external application. In that case, the external application is in charge of everything such as the logging and the calibration process which makes this solution reserved to computer specialists. However, it gives a total control on the eye-tracking process and thus makes possible the combination of eye-tracking with any other application. This is particular useful to build gaze contingent applications. It also allows to access the timer used to timestamp the gaze data which in turn allows an external application to make synchronous logging of other data. This latter functionality will be very useful for our situation. Both APIs are provided under the form of a dynamic library (DLL) for the Microsoft Windows operating system and can be used from another computer through a TCP/IP connection.

4.2.2 Technical setup

We implemented and tested four different setups of different complexity. The first two solutions use two computers and rely on the manufacturer software, Clearview, to operate the eye-trackers and on a third-party shared application to support the collaboration. While they are simpler to implement at first hand, the synchronization of data can be hard to accomplish and is generally not very accurate. The two other setups use a third computer which runs a custom shared application and which is in charge of the logging of all data. While it is more complex to implement, it offers much greater flexibility in the use and logging of the gaze data and can reach high level of synchronization accuracy.

Setup 1: Two computers and manual eye-tracker operation

This solution, which is presented schematically on figure 4.1, is certainly the most straightforward way to use two eye-trackers in parallel. It consists simply in running manually each eye-tracker using the ClearView software on both computers by using a custom stimulus setting. Then, the third-party shared application, which constitutes the stimulus, is run on each computer. Such a setup will produce three unsynchronized sets of data which have to be synchronized afterwards. First, we have the two sets of data, i.e. gaze data, input events, and screen recording, recorded by the two eye-trackers through the ClearView software. Secondly, we have the logs of actions and stimulus changes produced by the shared application. We assume that the shared application has some internal synchronization mechanism which makes it produce a unique log with a unique time base that contains events happening on both computers. This latter log provides the only way to relate the time of our different datasets.

The general idea for synchronizing these streams of data is to find correspondences between events logged in the shared application logs with input events logged by the eye-trackers on each computer. In this way, it is possible to establish a correspondence between the timestamps of each eye-tracker's logs and the timestamps of the shared application logs. Thus, all events are taken back to the time base of the shared application. The crucial point is that the shared application logs some events which correspond directly to input events for one of the two subjects, so that we can find in the log of one subject's eye-tracker the input events corresponding to the shared application event. For example, in one experiment we conducted (see section 5.2.2), the shared application was a tool to build diagrams consisting of labeled boxes and links. In this case, the application logged the creation of new boxes with the identifier of the subject who created it. Creations were generally accomplished by making a double-click which allowed us to match the double-click events appearing in the subject's log with the box creation events of the application. However, we required several instances of such cases because there were other double-click events that did not correspond to box creations. Thus, the solution consisted in having several such events (actually, we also matched other types of events) and in finding the best time offset so that they all match with their corresponding input events on the subject's log. This step is accomplished for both subjects separately and we ended up with two time offsets, one for the difference between one

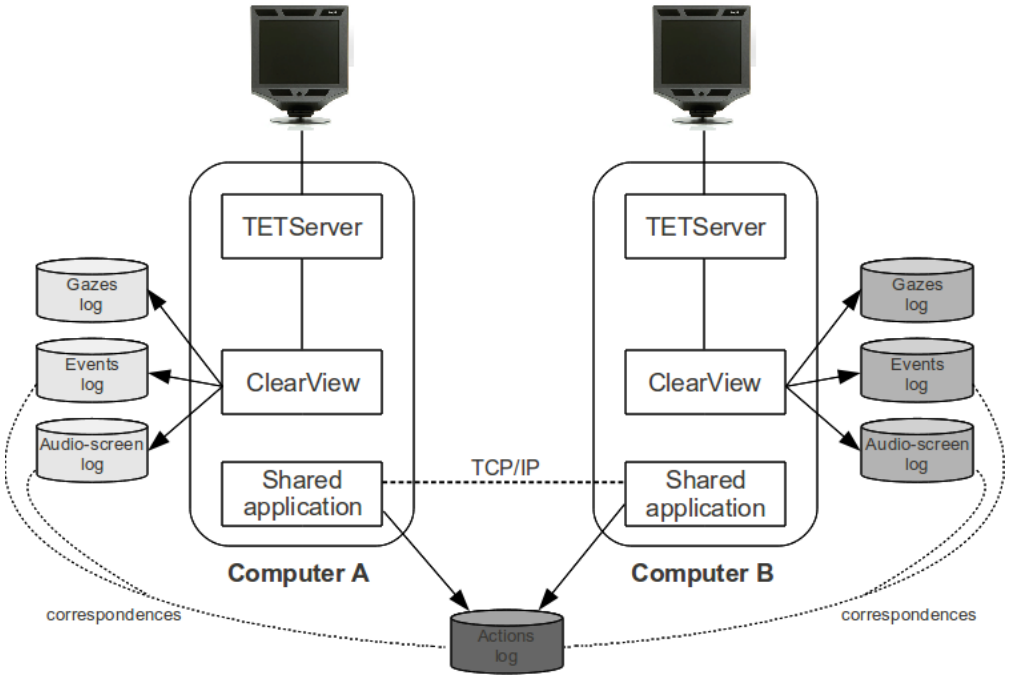


Figure 4.1: Dual eye-tracking technical setup 1. Differences of gray levels indicates that logs do not have the same time basis

subject and the shared application and one for the difference between the second subject and the shared application. With this in hand, the times of all data streams could all be taken back to a single time base.

There is another possibility of synchronization which is relatively more difficult to perform but that may save the situation when the shared application does not offer the required action log. The approach consists in detecting shared events that happen on both computers at the same time in the screen recordings of both subjects. For example, in the example described above, when a box creation occurs, it appears on both screens at the same time. Thus, it could have been possible to detect the times of appearance of boxes in the screen recordings of each subject. Thus, we could find the time offset between the eye-tracker logs of each subject. Note that while this may seem tricky because it requires computer vision, this may be often relatively easy because as we are dealing with very regular images and the comparison can be accomplished by doing basic image matching. We actually used such a technique to perform a spatial synchronization of the gaze data coordinates with the shared application coordinates. Indeed, in this case, the workspace in which subjects could create boxes and links was bigger than a single screen and they had the possibility to scroll in this space. However, the application did not log scroll events which made that we had no information of what was the current view of the workspace for one subject at a given time. Thereby, we used the video to detect changes of viewport for each subject.

Finally, there is a third solution to find time correspondence between the two eye-tracker

data which can be used even if we do not have simultaneous events . This technique requires however that the subjects are able to speak to each other and that either they are in the same room so that both eye-trackers record both voices, or that a third application records both voices. In such a situation, we could use the audio recordings included in the screen recordings of the eye-trackers and find the best time offset between them so that they get aligned. Indeed, if both voices are present on both recordings, even if each recording has a different relative amplitude between the two voices, it is possible to align the audio streams to make them match at best. Similarly, if we have a third recording with both voices in it, it could be match separately with the audio streams of each eye-tracker because they have portions of sound in common. In the example described above, we had such a third audio stream that came from an audio conferencing software used for communication between the two collaborators. We used this technique to synchronize this third audio stream with the eye-tracker's time so that we ended up with our external audio streams to be synchronized with all other data.

The quality of synchronization of this method depends greatly on the logging accuracy of the third party application. Indeed, it requires that the logging of shared actions is done in an accurate way. Qualitative analyses also revealed that a time compression or expansion may exists between the two computers which means that the time passes faster on one computer. This requires to make a more subtle correction than a simple offset such as to compute an offset and a compression factor.

Setup 2: Two computers with automatic eye-tracker operation

In order to avoid the need of these post-synchronization steps, we implemented a slightly different solution which relies on the use of the high-level eye-tracker programming API. The general setup is similar to the previous solution with two separated computers, each one running the eye-tracker software separately. The only difference is that we use the programming API to launch the recording process in ClearView on both computers at the same time. This requires the development of a small application that could run on any of the two computers and that is responsible for sending a triggering message to both computers at the same time. The rationale is that if both eye-tracker softwares are started at the same time, the beginning of recordings of both eye-trackers could then be considered as representing the same time and the offset between the two eye-tracker times could be directly computed from these values. This setup is shown on figure 4.2.

It remains that it is still necessary to synchronize the shared application logs with the two eye-tracker logs. Different cases are possible. First, the shared application can be closely linked with the application responsible of triggering the eye-tracker recording process. This is in principle reserved to situations in which we use a custom, or open-source, shared application so that it can easily integrate the module responsible for starting the eye-tracker and can register the time at which the eye-tracking process is started. Second, if we use a third-party closed-source application, we have to find a way to inform the application about the starting time of the eye-tracking process.

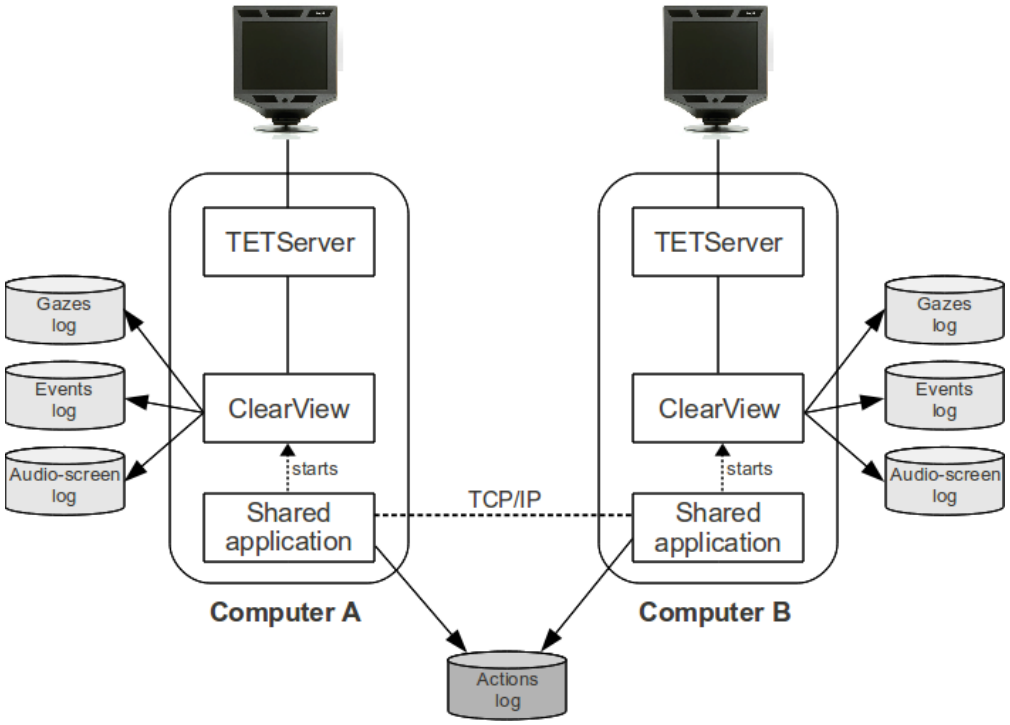


Figure 4.2: Dual eye-tracking technical setup 2. Differences of gray levels indicates that logs do not have the same time basis

This setup is slightly better than the previous one because it doesn't require any post-processing for synchronizing the various data streams. However, there are potential drawbacks with this technique. First, it requires some programming of application which has to start the eye-tracking process. While this is not a huge task, it still requires some competencies and care to be performed correctly. This is however not a real drawback compared to the previous method as the first step requires also programming development for the post-synchronization part. Second, this method does not work with any shared application because, as we explained above, it is necessary to be able to make the application aware of the precise eye-tracking starting time. Third and more importantly, this method relies on the assumption that the triggering of the recording process is fast and accurate and that will happen at the same speed on both computers. As this works through the ClearView software, there is a great chance that it is not very precise and that some variable time lags may occur between the triggering message and the actual start of recording. Finally, this solution doesn't take into account the potential presence of a time compression or expansion between the two computers.

Setup 3: Three computers with synchronous data logging

Considering the difficulty of the previous methods as well as their imprecision in the resulting synchronization, we developed another technique which is based on the low-level eye-tracker API. This solution consists in writing an application that is entirely responsible of the eye-

tracking process for both computers. Of course, this also implies, as in the previous solution, to use a custom, or at least open-source, shared application that allows for all data to be logged in the same way.

As shown on figure 4.3, this setup uses three computers, one for each eye-tracker and one third server computer which runs the shared application and which is in charge of all logging. The two client computers run only lightweight applications which send user actions to the server and receive changes to apply from it. The server application is the core of the system. It must integrate calls to the eye-tracker library to start the eye-tracking process and to collect the gaze data. In short, the eye-tracker API provides a mechanism which consists in defining a callback function which is called for each new gaze data. The parameters of the callback function contain all the raw gaze information as they are described in section 3.1. In order to be able to communicate to both eye-trackers from the same computer, we duplicated the DLL of the eye-tracker API and we called them separately within the server application. The synchronization of the data was done by putting new time-stamps to the gaze data as they were collected by the application.

The problem of this technique is that it relies on the re-time-stamping of the data by the server. This may cause problems for two reasons. First, the server does not necessarily have an accurate timer and secondly, it introduces a latency (due to transmission) between the moment the data is collected and the moment it is time-stamped by the server.

Setup 4: Three computers with unique high-precision timer

We have finally been able to correct the issues of the previous setup by using a special feature of the eye-tracker API. Indeed, the API provides a way to choose how the data are time-stamped. More specifically, it allows one to use the timer of another computer. By default, it uses a timer from the computer running the TET server (in our case, the two client computers). However, it is possible to tell the API to use the timer of the computer which makes the calls to start the eye-tracker (in our case, the server). Hence, it is possible to instruct both eye-tracker to use the timer of the server computer which makes them using the same timer and hence timestamping the data with the same time base. Note that such a use of this functionality, i.e. to have two eye-trackers using the same timer, is not planned by the manufacturer and they explicitly told us that they could not guarantee that it will work (personal communication with Tobii). However, we made several tests which show that this actually works as expected. Indeed, even when the eye-trackers are started at different times, both gaze data streams have comparable timestamps. The general technical setup is similar to setup 3 (see figure 4.3).

This method also has the great advantage that the timer used by the eye-trackers to timestamp the gaze data can also be accessed through the eye-tracker API. Hence, it is possible to use it to time-stamp all other data produced by the shared application. The only data that required special artifact is the audio data. Indeed, this latter is not logged but simply recorded in an audio file. Thus, it is not possible to timestamp it as the time goes. Our solution consisted

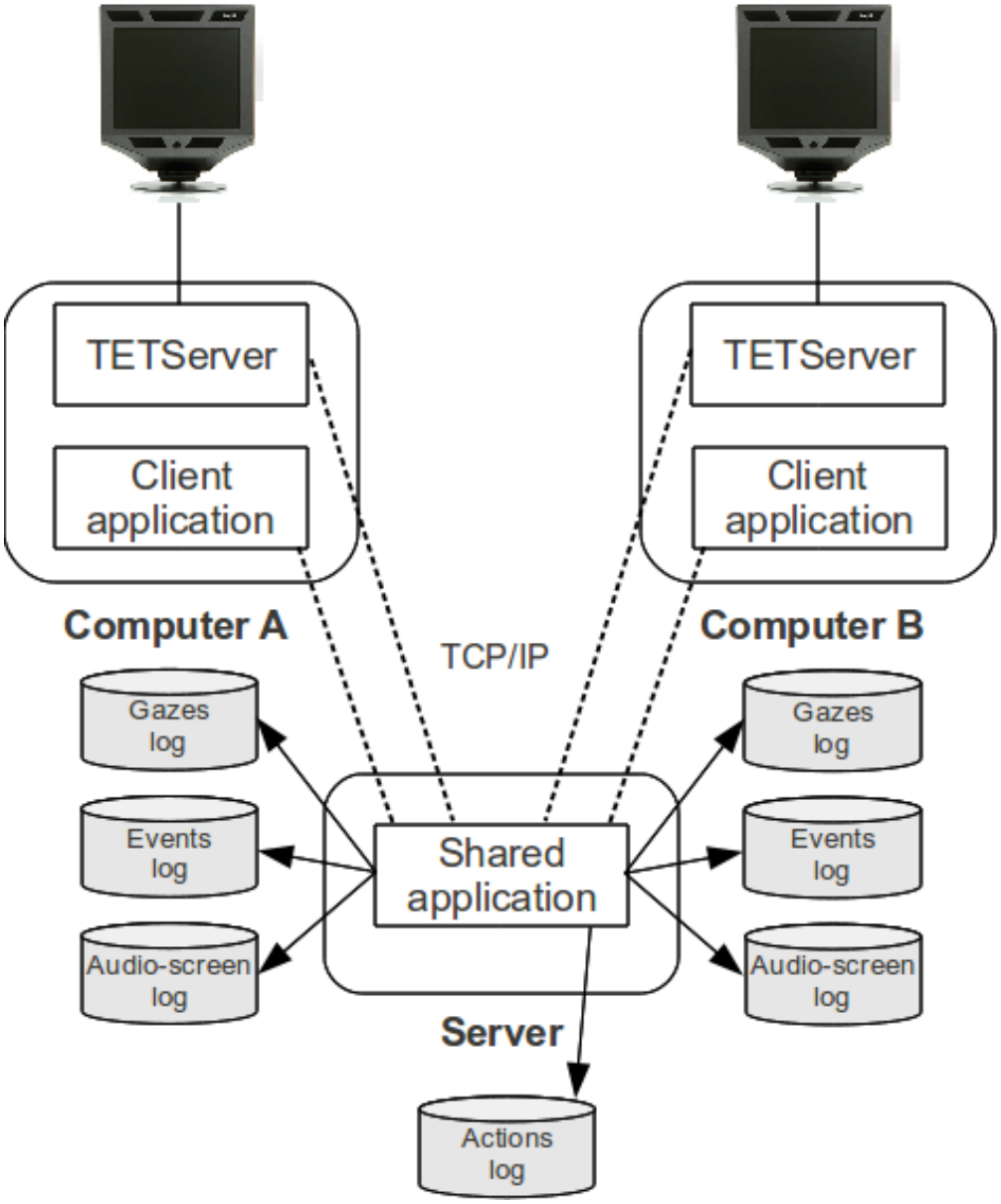


Figure 4.3: Dual eye-tracking technical setups 3 and 4

in using some functions which allow us to know the current frame number of the audio file currently recorded. Thanks to such a function, we periodically logged correspondences between the current time provided by the eye-tracker API and the current recorded frame of the audio file. It was then very easy to compute the offset and scaling factor between these two values and the result revealed to be very precise with a near-perfect linear relationship between the two streams.

This solution appeared clearly as being the best we can achieve with this hardware. It reaches perfect accuracy, i.e. the gaze data of both eye-trackers are timestamped with the same time

base even if they are started at different moments. The only aspect which needs to be handled with care is the logging of the subjects' action data. Indeed, as it is the server computer which is in charge of the logging, all actions done on a client computer must first be sent to the server before being logged. This could introduce a little lag in the actions log because of the duration of message transmission. Finally, this method also relies on the mechanism used by the eye-tracker server to use the timer of a third computer and we don't know how well it is accomplished. This latter aspect is the only potential source of synchronization errors.

4.2.3 Discussion

Dual eye-tracking studies imply several technical issues mainly related to the synchronization of the various data streams that are recorded. In particular, the main challenge is the accurate synchronization of the gaze data produced by two separate Tobii 1750 eye-trackers. We have presented different possible technical setups allowing us to achieve this goal with variable accuracy level. The final setup we proposed achieves this goal in a very effective way. It allows us to record all data, i.e. both subject's gazes, subject's actions, stimulus changes and audio data in a synchronous manner through the use of a single timer.

Of course, these considerations are very dependent on the type of eye-trackers and the mechanisms they offer for their operations. Hence, the solutions presented here concern mainly the Tobii 1750. However, they also give insights on how to solve these issues in more general situations.

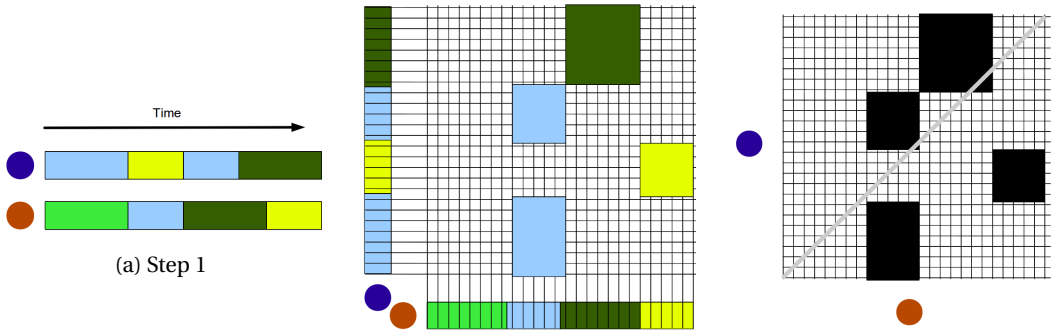
4.3 Cross-recurrence and dual-gaze analyses

In order to characterize quantitatively dual eye-movement patterns, we need some indicator that measures the relationship between the eye-movements done by the two subjects. We focus particularly on gaze cross-recurrence, which is a special case of a very general type of analysis to study recurrence in dynamical systems (Eckmann et al., 1987). In the context of gaze analysis, eye-movements can be considered as a dynamical system in which the gaze position or the looked AOI represents the state. Thereby, cross-recurrence analysis can be used to measure how much and when two subjects look at the same thing. It was first used in this context by Richardson and Dale (2005) who analyzed the coupling between gazes of a speaker and a listener of a monologue. They used such an analysis to show that there was a coupling between the speaker's gazes and the listener's gazes and that this coupling was related to some aspects of the interaction, such as the level of comprehension or the common ground between the subjects.

4.3.1 Cross-recurrence analysis

The principle of cross-recurrence is to build a binary matrix, called a cross-recurrence plot (CR plot in short), that displays similarities between two temporal sequences of states of some dynamical systems. Such a matrix has the time of one dynamical system as first dimension and the time of the second as second dimension. Hence, each point of the matrix corresponds to a specific time for the first system and a specific time for the second. The value in a given cell of the matrix indicates whether the states of the two systems for their respective times are recurrent or not, i.e. whether they are similar or not. As state values are generally continuous, similarity is usually assessed by testing whether the difference of state values is lower than a threshold. The construction of such a matrix for systems having discrete state values is described schematically on figure 4.4. The resulting CR plot (figure 4.4c, right image) can be visually interpreted as follows. First, the amount of black parts (recurrent states) gives an overall idea of how similar in general the states visited by the two sequences are, independently of the time. Secondly, the position of the black zones in the matrix gives some information about the relationship between the two sequences. Indeed, the main diagonal (or Line-Of-Identity, LOI) of the matrix represents synchronous points, i.e. points which correspond to the same time for both systems. Thus, if points around the LOI are blacks, it means that they are in the same state at the same time. The points below the diagonal correspond to moment happening sooner for the vertical (blue) system than for the horizontal (red) one. Hence, if the plot contains more black below or above the LOI, it shows that one system visits the same states than the second with some delay. In the resulting CR plot of figure 4.4 (right image), we see that the black parts are a bit more present above the diagonal, which means that there is a tendency for the blue (vertical) system to "follow" the red (horizontal) one: the blue one is in the same states as the red ones after some delay. In a general manner, CR plots provide a visualization of the coupling between two dynamical systems.

Visual interpretation of CR plot only provides qualitative insights about the coupling between the two streams of states. It is however possible to quantify this coupling by using cross-recurrence quantification analysis. It consists in measuring the amount of recurrence for different lags between the two streams and to plot these recurrence values in a graph according to the lag between the two streams. More specifically, each diagonal of the CR plot corresponds to a specific lag between the streams determined by its distance to the LOI. The recurrence rate for a given diagonal is simply the ratio of black points on the diagonal. Figure 4.5 shows how the quantification is performed. The resulting graph (figure 4.5, right part) shows the recurrence rate on the vertical axis for different lag values on the horizontal axis. A lag of 0 represents the synchronicity: to what extent the systems are in the same state at the same time, while positive (respectively negative) lag values represent "leading" dynamic for the vertical (respectively horizontal) system. In the example of figure 4.5, there is a tendency for the red system to "lead" the blue one. In other words, the blue system follows the same states as the red one after a bit more than 500 ms. Actually, there are several other quantification of CR plot that can be done and the one presented here, namely lag recurrence quantification, is not the most common one but, as we will see, it is a very interesting measure for gaze data analyses.



(a) Step 1

(b) Step 2

(c) Step 3

Figure 4.4: Schematic example of cross-recurrence analysis with discrete state values. On the left, we can see the two sequences of states (indicated by colors) of two systems (red and blue) that are compared. One of the sequences is rotated such that we obtain a matrix where every point corresponds to a given time for red and a given time for blue. The points of the matrix get filled when the corresponding states of both systems match (middle graph). Finally, we abstract the specific colors of the matching states and we obtain a cross-recurrence plot (right graph)

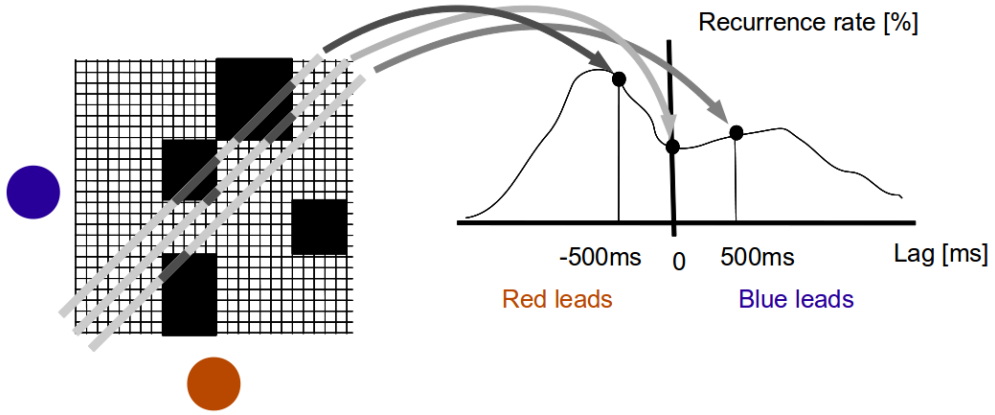


Figure 4.5: Schematic explanation of how cross-recurrence quantification is performed using the same example as in figure 4.4. Recurrence rates are computed for many diagonals around the central diagonal and their corresponding values are plotted in a graph showing the recurrence rate according to the lag between the two streams

4.3.2 Gaze cross-recurrence

As explained above, cross-recurrence can be used to analyze two streams of gaze data. There are however some specificities in the usage of cross-recurrence analysis in this context. First, gaze data, i.e. fixations lists, must be sampled in order to obtain a regular time series of values. This is generally done by choosing some time period and by taking for each sample the corresponding fixation. However, it may happen that a sample falls in between two fixations, i.e. during a saccade or during a missing value. In such cases, we can choose to take the value

of the previous or following fixation if the sample is close enough to one of these two. But sometimes, particularly in the case of missing data, the time between the current sample and the closest fixation is too large and we cannot give a value to the sample. Hence, the resulting time series may have several missing values and this has to be taken into account when computing the CR plot and the following recurrence quantification. Actually, these missing points will be propagated to the CR plot as vertical or horizontal bands of missing values, thus resulting in a three-value CR plots: missing, not recurrent or recurrent (see figure 4.6). For the recurrence quantification, these points are simply not taken into account for the computation of the ratio of recurrent points.

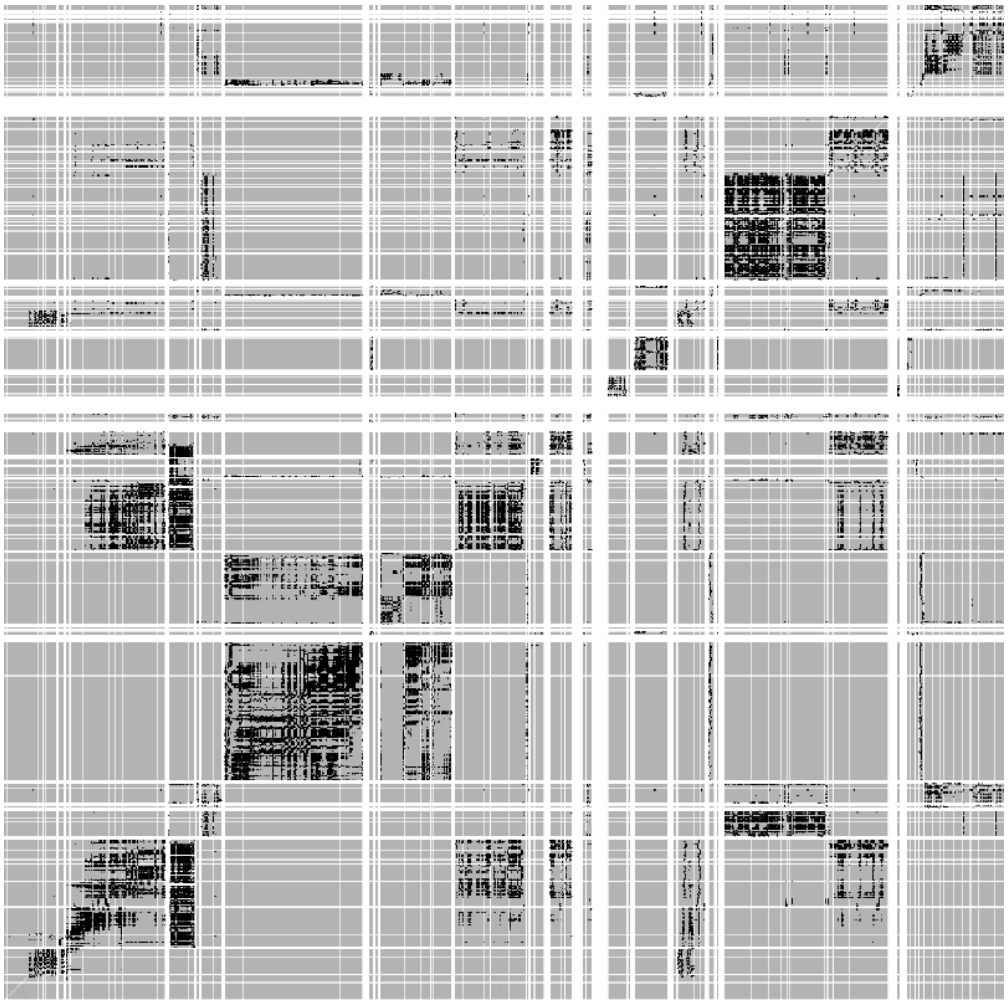


Figure 4.6: Example of a cross-recurrence plot computed from gaze data. White indicates missing data, gray represents non-recurrent point and black is for recurrent points.

In order to perform cross-recurrence analysis with gaze data, we also need to define how to test the similarity between two fixations. The idea is that we consider gaze position as the state of a dynamical system. This can be done, either by considering the corresponding AOI as the state value or by taking the fixation coordinates. If we consider fixation coordinates, similarity

will be assessed by testing whether the Euclidean distance between the two compared gaze is lower than a certain threshold. If we consider instead the AOI, then we will generally simply test whether the two compared AOI are the same or not. Finally, we can also use probabilistic AOI (see section 3.4.1) in which case we can compute the probability that they look at the same AOI and test whether this probability exceeds a certain threshold.

4.3.3 Gaze cross-recurrence and dialogue

Richardson and Dale (2005) used cross-recurrence analysis to study the coupling between the gaze of a speaker and of a listener during a monologue. They recorded one person speaking about a well-known TV show while looking at the pictures of the six main characters. Then, another subject was asked to listen to the recorded monologue while looking at the same stimulus. They used cross-recurrence analysis to compare the gazes of the speaker and the listener and they found that cross-recurrence rate peaked at lag of 2 seconds (see figure 4.7), meaning that the listener looked at the same pictures as the speaker 2 seconds later. They also compared these recurrence rates with a random baseline constructed by mixing the temporal order of the gaze data and showed that the peak significantly differed from this baseline while recurrence values at higher lags (>2s) were comparable to this random baseline. Finally, they also showed that the height of the peak was correlated with the level of understanding of the listener, which was assessed through a questionnaire. In other terms, listeners who looked more in the same sequence than speakers (with some delay) were also those who understood the best the monologue.

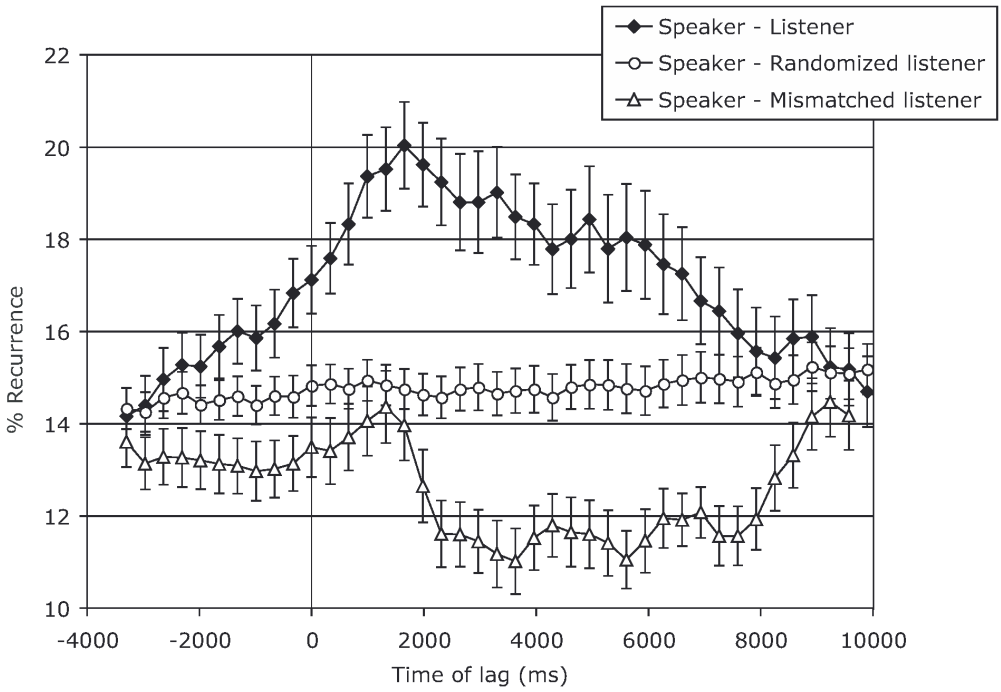


Figure 4.7: Average cross-recurrence for various time lags (from Richardson and Dale (2005))

In a follow-up study (Richardson et al., 2007), they conducted the same kind of analyses with gaze data from a dialogue situation. They asked two people to discuss about a painting and varied their level of common-ground by providing some pairs with the same information about the painter and other pairs with different information. Results showed a cross-recurrence rate that peaked at a lag of 0. They explained the difference of results by the fact that both participants spoke turn-by-turn which resulted in an average maximum recurrence for a lag of 0 second. Indeed, we could imagine that the situation consists in two curves such as the one shown on figure 4.7 with one being horizontally reversed. Hence, depending on the exact shape of the curve, the sum of the two curves could be the highest at a lag of zero. They showed that the average recurrence level for lags lower than 3 s was significantly higher when the subjects had a better common ground.

4.3.4 Gaze cross-recurrence and collaboration

In the present work, we used cross-recurrence analysis to study gaze coupling during real complex collaborative activity. This implied several issues that were not present in the previous works mentioned above. Indeed, it appeared that cross-recurrence measures in complex collaborative situations were not only affected by the short-time gaze coupling effect due to dialogue found by Richardson and Dale (2005) but also on a longer time-scale by some aspects related to the structure of the collaboration. This makes the raw recurrence values not directly interpretable and comparable between pairs in terms of short-time coupling.

Indeed, in some tasks, collaborators do not necessarily need to look at the whole shared workspace uniformly. Instead, depending on how they decide to solve the task, they may end up looking a lot at some specific zones of the workspace and very little at other places. The consequence of this is that it will affect the baseline, or random level, of recurrence, hence the level of recurrence for high lag values. Indeed, the random level of recurrence is directly linked with the probability of looking at the same place by chance. So, if the two collaborators look at a smaller area, they are more likely to look at the same place by chance, thus resulting in higher recurrence rates in general. Hence, if two pairs solve the problem differently with one pair focusing more on a specific part of the stimulus then, this latter will have a higher level of recurrence, disregarding the possible presence of a short-time coupling due to dialogue. This makes the cross-recurrence values not comparable across pairs.

A second related issue comes from the fact that during collaboration on a shared workspace, collaborators' attention, and hence gazes, may be preferentially located in specific zones at specific moments in time. Typically, they can solve different aspects of a task at different moments in time and thus, they focus on different parts at different moments. This time-dependent attentional focus can be synchronized between the two subjects if they collaborate, i.e. solve the task together all the time, or unsynchronized if they cooperate, i.e. each solve a part of the task on their own and then reassemble their individual work. This phenomenon is illustrated on figure 4.8 in which we identified with colored squares the main zones of

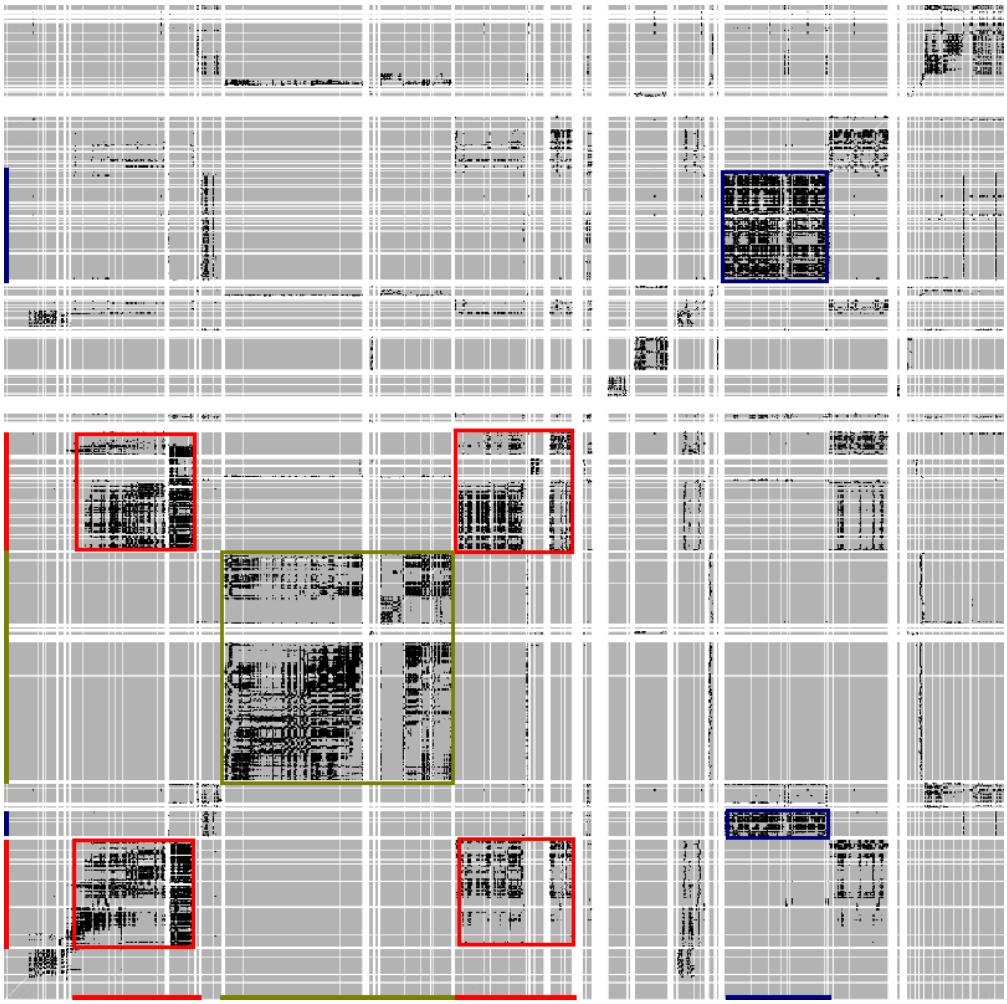


Figure 4.8: Cross-recurrence plot of figure 4.6 with zones of exploration highlighted in different colors. Color strips on the bottom and on the left of the plot indicate which zone each subject was looking at each moment. They were almost always collaborating, i.e. looking at the same zone at the same time. Only at the beginning, the vertical subject had a glimpse at the blue zone, which they explored jointly later on in the interaction.

the stimulus that were explored. Collaboration will induce a wide peak of cross-recurrence centered on lag 0. The cross-recurrence level will start to increase even for high lag values because the random chance to look at the same thing will be higher as long as they are in the same zone. In figure 4.8, the green zone will produce a wide central symmetric peak of recurrence because they spent a lot time looking at it together (the red and blue zones will also contribute to this central peak but only for smaller lags). If instead subjects cooperate more and each focus on specific zones at different moments, asymmetric peaks of recurrence may be formed at high lag values. Indeed, if one subject visits a zone that was previously visited by her partner, this will produce a peak of recurrence at a lag corresponding to the time between the visits of the first subject and the second subject. For example, in figure 4.8, the blue zone

will form a peak of recurrence at high lag values because of the square in the bottom right part, which is due to the fact that the vertical subject looked at the blue zone long before the horizontal one. Note also that such peaks at high lag values may also appear if they collaborate and visit zones that they previously visited. This is the case of the red zone in figure 4.8 as it produces two off-diagonal recurrence squares. Finally, it is frequent that a pair collaborates for some time and cooperates during other moments, as it is the case in figure 4.8, which results in the presence of both types of effects. These effects are also a form of gaze coupling because they look together at the same zone, but this is not the same kind of direct, short-time coupling due to dialogue. Instead, they are a longer time-scale coupling which reflect the style of collaboration instead of reflecting "gaze following" patterns as in the case of dialogue.

Finally, whether there is some zones exploration effect as described right above or not, there is a third sort of phenomenon that could affect resulting cross-recurrence. We refer to it as temporal trends in stimulus or zone exploration. Specifically, it is the fact that when we explore a particular zone or even the whole stimulus, there may be a trend in the temporal course of the exploration. The typical example is a text stimulus for which we clearly have a trend from the top to the bottom because of the reading direction. Indeed, we generally start to read a text from the top to the bottom. Hence, even if collaborators do not speak to each other, if they start more or less at the same time to read the text, they will tend to look roughly in the same area at the same moment. This will have as a consequence that the chance of looking at the same place will be higher for small lags than for high lags. Hence, it also produces a peak in the cross-recurrence curves.

These phenomena may affect importantly the interpretation of cross-recurrence values as the resulting values do not only represent gaze coupling but also the type, the structure and the course of the collaborative activity. Hence, if we have one pair with a high recurrence peak and another with a lower recurrence peak, it doesn't necessarily mean that the first one is more coupled, but it may simply mean that they were working on a smaller area of the screen. Or, if we have a pair who have a recurrence level that increases for lags of 40s, this certainly doesn't mean that they tend to look at the same thing 40s after their peer, but rather it is certainly explained by the fact that they worked together on small areas for periods that lasted up to 40s. These different effects are exemplified with artificially generated data in section 4.3.5. We propose hereafter some possible methods to cancel out these phenomena, so that we could really measure the short-time coupling due to dialogue within a complex problem solving task.

Random level correction

We developed a simple method to correct problems related to the first issue described above, namely the fact that collaborators may look at smaller portions of the shared workspace. Indeed, this effect was very present in on experiment we conducted (the Shoutspace experiment, see section 5.2.1). When we looked at the curves of cross-recurrence for each pair individually, we saw that some pairs were much higher than other pairs for any lag values, including for

lags of several minutes. We have first confirmed that this came from the phenomenon described above, i.e. that they look at smaller portions of the workspace area. Indeed, we have computed for all pairs the general gaze divergence independently of the time, by computing the JensenShannon divergence of the two overall gaze distributions over the whole map. This value is high when the gazes are distributed on different portions of the workspace and low if they are on the same region. We found out that this global gaze divergence was well correlated with their random level of recurrence computed for high lag values. The explanation is that a high divergence value reflects that they look together in a wider area, but also that they look differentially at some zones, i.e. that one subject looked a lot in one zone and the other more in another zone. In either case, we thought it is a good justification for the differences in the cross-recurrence baseline levels. Having realized that, we proposed a very simple correction which consists simply in subtracting, for each pair, their randomized cross-recurrence level from their cross-recurrence curve. Hence, the resulting values can be interpreted as the level of recurrence above chance level.

Zone-based cross-recurrence

The random level correction proposed above doesn't take into account the second type of phenomenon, namely the fact that collaborators explore the stimulus by zones, which can affect cross-recurrence values, independently of any low-level coupling. We propose another type of correction that aims to separate this effect from the short time coupling due to dialogue. The basic idea is that there are two kinds of coupling. On a short time scale, i.e. for small lag values, we can have a coupling due to dialogue which makes people following closely the gazes of their partner with a lag of roughly 2 seconds. On a longer time scale, we have a zone coupling, which makes collaborators exploring jointly the same zones of the workspace.

We propose to compute a zone based recurrence rate that combines one recurrence rate for each zone of the stimulus that is explored. The aim is to decompose the overall recurrence rate curve and to estimate the low-level coupling peak by normalizing the curve for the number of objects in each zone as well as for the structure of exploration. A zone is a collection of objects that are explored together either because they have a logical link (e.g. lines in a JAVA method or a paragraph) or simply because they are co-located on the same region of the screen.

Hereafter, we consider idealized situations of increasing complexity where two persons who look at one or several zones of N objects with a uniform probability distribution of gazing at any object. However, each person also has a coupling probability C of looking at exactly the same object than the other subject some time before. We discuss the use of cross-recurrence analysis to study such an artificial situation. The goal is to deduce the coupling probability from the lag-recurrence quantification analysis.

Case 1: single zone with N objects, coupling of probability C at fixed lag l First of all, we have to note that the baseline probability of recurrence in case of no coupling is $\frac{1}{N}$. Indeed, the

probability that both subjects look at given object O is $\frac{1}{N^2}$ and thus, the probability that both look at the same object is the probability of looking jointly at any of the N objects, hence $N\frac{1}{N^2}$. Concerning the coupling, it always occurs with the same time lag l: when a subject is coupled at time t, he takes the value of the other subject at time $t-l$. In such a simple situation, it is easy to compute the expected value of recurrence for a lag l. Indeed, the coupling probability C defines the ratio of points that will be coupled for this lag and thus, the resulting recurrence value will simply be the combination of those coupled points and the uncoupled ones, these latter having the baseline recurrence probability. Hence, the following equation gives the expectation of recurrence value at lag l:

$$R_l = C + (1-C) * \frac{1}{N} \tag{4.1}$$

The expectation of recurrence values for all other lags is equal to the baseline value $\frac{1}{N}$. And from formula 4.1, we can derive the formula to compute the coupling probability from the recurrence level:

$$C = \frac{R_l - \frac{1}{N}}{1 - \frac{1}{N}} \tag{4.2}$$

Case 2: single zone with N objects, coupling of probability C with variable lag of distribution L The difference with a fixed coupling is that the coupling may occur at different lags. More generally, we suppose that these lags follow a probability distribution L, such as a normal distribution. A simple way to solve this consists in seeing it as an extension of the fixed lag case. The idea is to decompose the coupling probability C into lag coupling probability C_l, the probability of having a coupling at a specific lag l, so that $C = \sum_{l \in L} C_l$ where L is the set of lags having a non-null coupling probability. We can thus rewrite equation 4.1 as

$$R_l = C_l + (1-C_l) * \frac{1}{N}, \forall l \in L \tag{4.3}$$

Again, the expectation of recurrence values for all lags $l \notin L$ is equal to the baseline value $\frac{1}{N}$. And from this formula, we can derive the formula for the general coupling:

$$C = \sum_{l \in L} \frac{R_l - \frac{1}{N}}{1 - \frac{1}{N}} \tag{4.4}$$

Case 3: multiple zones, coupling of probability C with variable lag of distribution L The approach that we take is to compute recurrence rates separately for each zone z (with N_z objects) in the set of zones that constitute the stimulus Z. We then combine them by weighting their importance by the time spent in each zone t_z. For each lag and for each zone, the coupling

can be estimated as the relative height of the recurrence peak above the baseline of the zone.

$$R_{lz} = C_{lz} + (1 - C_{lz}) * \frac{1}{N_z}, \forall l \in L, \forall z \in Z \tag{4.5}$$

Finally, the values for the coupling are averaged over L and added up with a weight proportional to the proportion of total duration that was spent in the corresponding zone t_z (the number of data points on which the diagonal R_L was computed). In such a way, it is possible to estimate an unbiased overall corrected gaze coupling.

$$C = \sum_{z \in Z} \left(t_z \sum \frac{R_{lz} - \frac{1}{N_z}}{1 - \frac{1}{N_z}} \right) \tag{4.6}$$

Note that the baseline recurrence level, i.e. $\frac{1}{N_z}$, is only theoretical when we assume equally-sized objects uniformly distributed on the screen. In practice, the probability to gaze at elements of the zone depends on their size and layout and thus, the computation of the baseline is not straightforward. It is instead possible to empirically estimate the baseline by averaging the recurrence rate R_l at extreme lags (-40 to -10 seconds and 10 to 40 seconds) or by shuffling the temporal order of the data.

4.3.5 Cross-recurrence simulation

In order to illustrate the issues presented in section 4.3.4 about the use cross-recurrence for gaze data in collaborative tasks, and to show the effectiveness of the proposed correction methods, we did some experiments by generating simulated data. More specifically, we showed that the zone based coupling measure we propose above is an indicator of the actual coupling between viewers that is relatively insensitive to the macro-structure of the observation sequences. We consider a simple situation in which we have a stimulus with N objects, possibly grouped by zone. The generated data are uniformly distributed over all objects, or over all objects of a zone. In addition, an artificial coupling with fixed probability is inserted in the data, i.e. some data points of one stream are forced to the value of the other stream some lag before. We also vary the type of collaboration (versus cooperation) and the possibility of revisiting a zone.

Parameters Four parameters determine the high level features of the cross-recurrence plot.

- Number of zones.
- Number of objects in each zone.
- Sequentiality of zones indicates whether viewers visit zones only once or whether they return to visit them on several occasions.

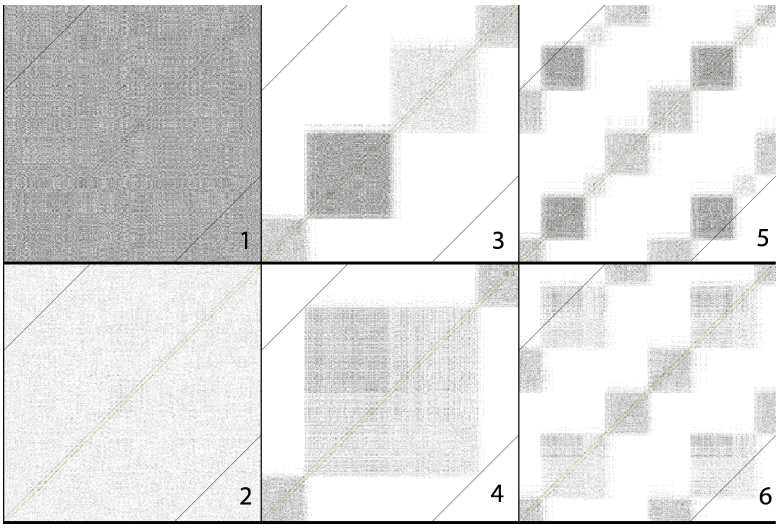


Figure 4.9: Cross-recurrence plots for six scenarios. Numbers refer to the scenario ID in table 4.1

- Synchronicity indicates whether the two viewers visit zones in the same order.

Four parameters determine the type of coupling that can be generated.

- *Level* of coupling determines the probability for a fixation of one viewer to be identical to a previous fixation of the other viewer.
- *Mean lag* of coupling determines the average lag at which coupling occurs.
- *Standard deviation* of coupling lag determines variations of the coupling lag, this latter being normally distributed.
- *Symmetry* of coupling determines whether subjects are equally likely to follow each other's gaze or whether one subject is leading the other.

Scenarios We designed six scenarios to illustrate the effect of the number of zones, the sequentiality as well as the synchronicity of exploration (see table 4.1). For the coupling parameters, only the level of coupling was varied systematically from 0 to 1 by increments of 0.1. The lag of coupling was held constant for all simulations at 30 with a standard deviation of 5.

Results The macro-structure of recurrence plots (figure 4.9) nicely reflects the sequentiality and synchronism of exploration. Off diagonal squares correspond to viewers returning on previously visited zones. This effect is absent when only one zone is visited (scenarios 1 and 2).

ID	Scenario	Z	Objects	Seq	Sync
1	uniform-10	1	10	-	-
2	uniform-70	1	70	-	-
3	sequential-sync	4	20,10,40,20	yes	yes
4	return-sync	4	20,10,40,20	no	yes
5	sequential-async	4	20,10,40,20	yes	no
6	return-async	4	20,10,40,20	no	no

Table 4.1: Scenario parameters used in the simulation. Z stands for the number of zones, Objects refers to the size of the zones, Seq indicates whether the exploration was sequential (one pass) or with returns on previously visited zones, Sync indicates whether the sequence of zones is the same for both viewers.

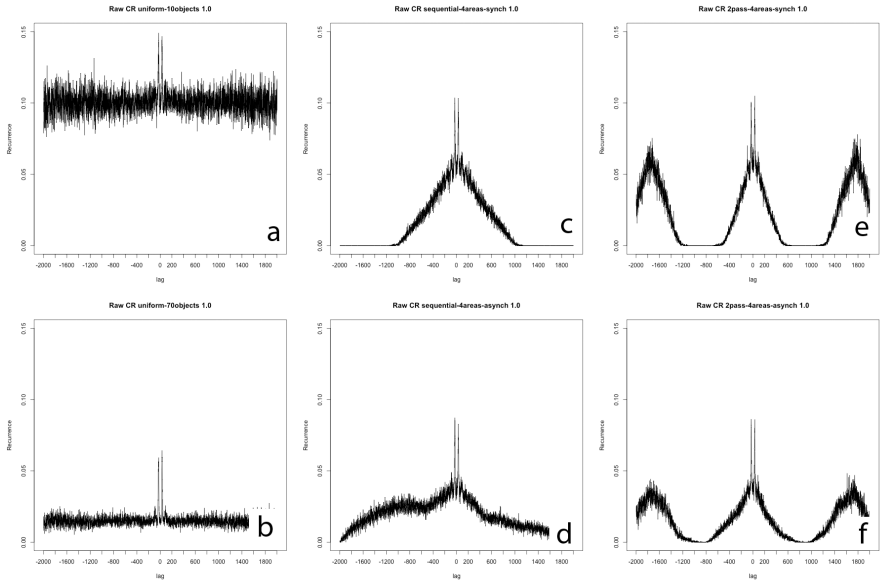


Figure 4.10: Diagonal recurrence rates for six scenarios. Subplots correspond to the recurrence plots in figure 4.9. Vertical scale is identical for all subplots.

The shade of gray is proportional to the baseline of the zone. The more objects are available for inspection, the less chance for recurrence at high lags, the lighter the shade of the plot.

The recurrence rates obtained from the simulated scenarios are depicted in figure 4.10. In all panels, a central recurrence peak is clearly visible in the graphs. This peak corresponds to the artificial coupling of the viewers that we have introduced in the simulation. The flat baseline in panels a and b are typical of the ideal case with only one zone of objects. The combination of zones containing various numbers of objects (and therefore various baselines) results in a sloped baseline best visible in panels c and d. The off diagonal rectangular shapes in the recurrence plots 5 and 6 (figure 4.9) create “bumps” in the recurrence rates represented in panels e and f.

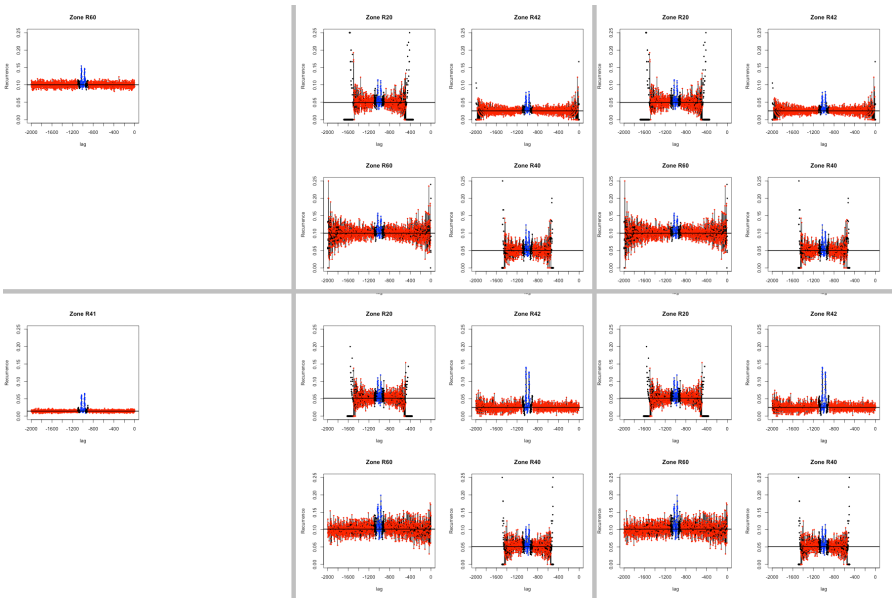


Figure 4.11: Decomposition of the recurrence into zones. Subplots correspond to the recurrence plots in figure 4.9

When diagonal recurrence rates are computed on a zone by zone basis, each zone features its own baseline and peak (see figure 4.11) and resembles the simple case with only one zone. The overall recurrence graph is a weighted sum of the zone recurrence graphs (proportional to the time spent in the zone). From these subgraphs, we apply the formula 4.6 to obtain the coupling.

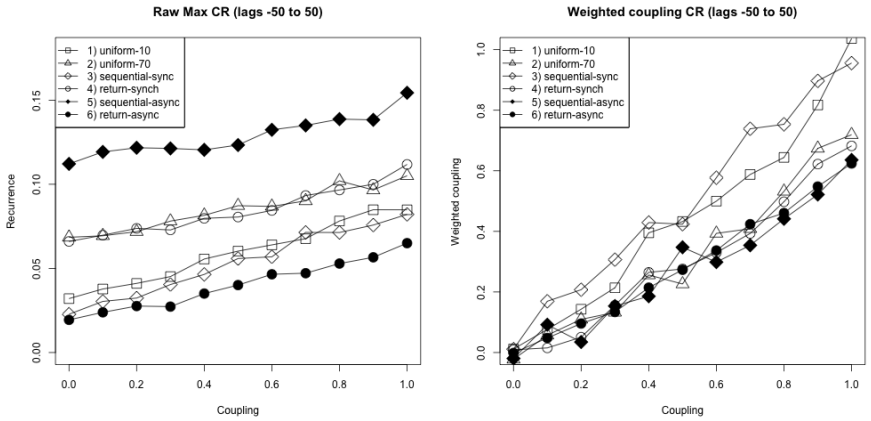


Figure 4.12: Recurrence rate and normalized coupling given a theoretical coupling from 0 to 1.

To estimate whether the measured coupling indicator is a good indicator for the actual coupling between viewers, we plotted values of the measured coupling as well as the raw recurrence rate against the values of actual coupling. Figure 4.12 shows that the measured coupling (right panel) is partially neutralizing the high-level effects of sequentiality and synchronicity of

the data streams and that it is a good estimator of the actual coupling between the viewers. On the contrary, we see that the raw recurrence rates on the left panel of the figure are sensitive to baseline variations due to the macro-structure of the recurrence plots and thus do not reflect the actual short-time coupling.

4.3.6 Discussion

We have seen that cross-recurrence is a very interesting indicator to study relationships between eye-movements of people interacting verbally. Indeed, Richardson and Dale (2005) have highlighted the existence of a gaze coupling phenomenon that is related to the type and quality of the discussion. However, we have also seen that cross-recurrence analysis may be affected by other phenomena than the short-time coupling due to dialogue. Indeed, in real collaboration tasks, there are several issues that can affect cross-recurrence analysis. The source of the various problems is generally the fact that when collaborators look at a smaller area, or in an area which contains less objects, then they have a greater chance of being recurrent than with a bigger, or denser, area. This effect may be present in different ways.

First, when comparing different pairs of collaborators, it may be the case that some pairs worked by focusing on a smaller part of the stimulus than other pairs. This makes the general level of recurrence not comparable between pairs. A simple correction for this issue consists in computing a random recurrence level by shuffling the gazes of the collaborators, or by computing the average recurrence level for high lag values, and then by subtracting this random level from all recurrence rates.

Second, depending on the task, collaborators may explore the stimulus by zones which results in recurrence peaks for lags corresponding to differences between moments when subjects explored the same zone. Here, we propose a more general correction that would take into account the zones and compute the recurrence separately in each zone. The resulting recurrences per zone are then combined by taking into account the size or density of the zone to correct for the random level. We have proven the effectiveness of this correction method with simulated data but it is difficult to assess how well it would work with real data. Moreover, its application to real situation is difficult because we have no good and objective way of defining the zones. A possible solution would be to define zones by applying a spatio-temporal clustering of the fixations in order to identify regions that are looked more or less uniformly for some time. Roughly, this would consist in detecting "meta-fixation", i.e. periods during which the fixations remain in a certain zone.

Finally, a last problem is that zone, or stimulus may be explored with some temporal trend, i.e. that there are more chances to explore a part of the zone at the beginning of the exploration than at the end, which increases artificially the recurrence level for small lags. This issue remains unsolved.

4.4 Discussion

In this chapter, we have presented some of challenges related to the analysis of eye-movements in collaborative situations. We have also offered some possible solutions to these issues. First, on the technical side, we showed that dual eye-tracking requires to synchronize in some way the gaze data recorded by two eye-trackers which do not have any specific synchronization mechanism. In section 4.2.2, we proposed and experimented various setups to accomplish this and we ended up with a very satisfactory solution which makes both eye-trackers to use the same timer.

The second challenge concerned the analysis of dual gaze patterns, i.e. the analysis of the relationships between eye-movements of two collaborators. We focused on a specific type of analysis, first used by Richardson and Dale (2005), which is cross-recurrence analysis. Indeed, such analyses have been used to show and measure the coupling between eye-movements of two people discussing together (see section 4.3.3). In section 4.3.4, we present and discuss this analysis in details and we show the difficulty of using it in real complex collaborative situations. We finally propose different possible solutions to overcome effects which are not due to the phenomenon of interest. The most promising solution consists in computing a zone-based cross-recurrence which permits us to cancel out the effects of an exploration of the stimulus by zones. Finally, we present some simulation experiments which illustrate these difficulties and show the effectiveness of the correction proposed (see section 4.3.5).

Eye-tracking applications Part II

5 Dual eye-tracking experiments

The second part of this thesis is dedicated to the presentation of experimental studies using dual eye-tracking. These studies have been conducted to identify patterns in the collaborators eye-movements which reflect the collaborative activity. This chapter present four exploratory experiments that have been conducted in this goal and which cover a wide range of tasks ranging from very conceptual activities such as concept-map building to sensory-motor tasks such as collaborative Tetris game. They illustrate how we progressively gained experience in dual eye-tracking. The next chapter will present the main experiment which has been designed from the experience gained during this exploratory phase.

5.1 Introduction

We first present a detailed description of these different experiments. We then expose the main results grouped according to which aspect of collaboration they are related to. The two first parts concern collaboration at the macro-level, i.e. relating eye-movements to the whole interaction. More specifically, we are interested in making relationships between collaboration quality on one hand and group composition, i.e. the expertise and difference of expertise between collaborators, on the other hand. The two other aspects are related to the micro-level, which consists in the analysis of short interaction episodes. First, we are interested in finding relationships between eye-movement patterns and the type of episodes, such as conflict or explanation episodes. Secondly, we study in more detail explicit referencing, i.e. when collaborators explicitly reference visual objects.

Two types of results are presented. On the one hand, we do classical statistical analyses that relate gaze features and collaboration indicators. This allows us to get some insights on how these two things are related. On the other, hand, we sometimes managed to do actual predictions about some aspects of collaboration from eye-movements patterns using machine learning algorithms. These results show the potential of dual eye-tracking for gaze aware applications. Indeed, by being able to make automatic predictions about the ongoing collaborative process, it may be possible to provide in real-time some help or feedback to

the collaborators to enhance their interaction. However, this type of results can difficulty be interpreted in cognitive terms.

5.2 Experiments

We conducted four main experiments which all consisted of a task performed collaboratively by two people on remote computers through a shared application. Two experiments were primarily designed to test the effect of a specific tool used to improve computer-supported collaboration and hence, have two or more experimental conditions. The two others were done only for the analysis of eye-movement patterns and have no special condition. One experiment, namely the KAT experiment, is about collaborative learning. Subjects have to learn a difficult topic by drawing together a concept-map representing the domain. Two experiments are on collaborative problem solving: the Shoutspace experiment in which pairs have to organize a festival on a campus, and the RavenBongard experiment which is about solving two kinds of short logical problems. Finally, the Tetris experiment is a collaborative version of the well-known puzzle game of the same name. We detail these various experiments in this section. Two of these experiments, namely the KAT experiment and the Shoutspace experiment, were associated to other doctoral works, namely Sangin (2009) for KAT and Cherubini (2008) for Shoutspace. More precisely, the analyses of the main results of these experiments, which were not directly related to eye-movements, were performed in these cited works while we were responsible for the analysis of the eye-tracking data.

All experiments had a similar technical setup which consisted in two separate workstations equipped with the same eye-tracker and in some cases, a third server computer. Each computer was either in different rooms or separated by a shelf so that participants could never see each other. The specific implementation used for gaze recording and synchronization was always one of the four setups described in 4.2.2. In three experiments, KAT, RavenBongard and Tetris, the subjects could speak to each other, either through a conferencing tool using a microphone and a headset (KAT experiment), or directly because they were in the same room (RavenBongard, Tetris). In all these cases, speech was recorded separately for each subject. In the fourth experiment, Shoutspace, participants could not talk but could use a chat tool to communicate. Table 5.1 summarizes the four experiments and their properties.

Table 5.1: Summary of the main properties of the experiments: task type, communication type and gaze synchronization method (numbers refer to the four setups presented in 4.2.2).

Experiment	Task	Communication	Synch. method
Shoutspace	Organize a festival on a map	chat	1
KAT	Build a concept-map	audio-conferencing	1
RavenBongard	Solve logical problems	direct speech	2
Tetris	Tetris game	direct speech	3

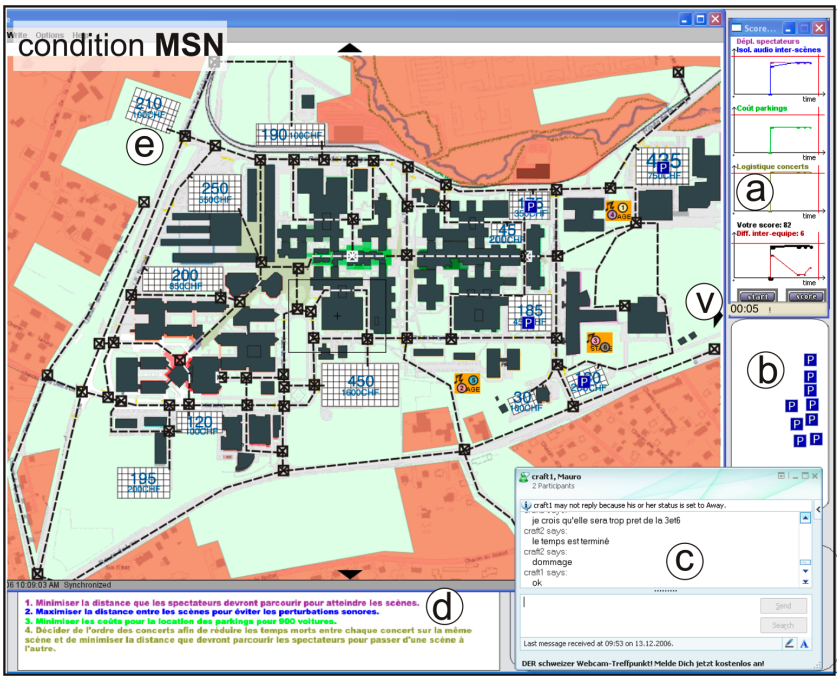
Gaze data of each experiment were processed with the adaptive fixation filter presented in section 3.2. Fixations offsets were corrected only in the KAT experiment with the brute-force method developed in section 3.3. Hits detection was performed with the simple, closest-object, method in all experiments.

5.2.1 Shoutspace

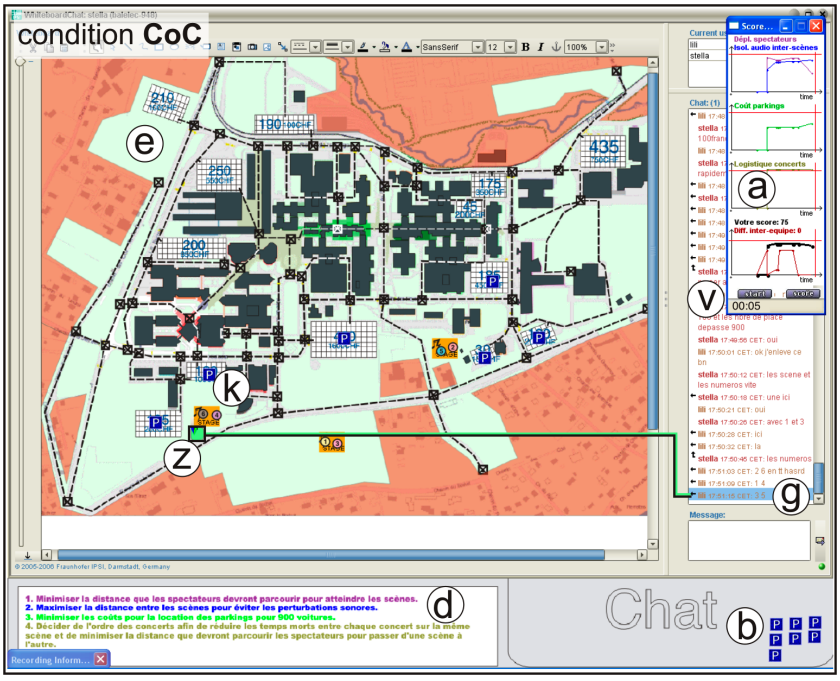
The experimental task (Cherubini and Dillenbourg, 2007; Cherubini et al., 2008; Cherubini, 2008) consisted in organizing a festival on a university campus map by collaborating remotely using a chat tool (see figure 5.1). This required subjects to decide which parking lots would be used by the festival attendants, where to position the three stages of the event, and how to allocate six artists to the three available stages. Subjects had to perform a number of optimizations, such as minimizing the distance between the chosen parking areas to the initial stage, minimizing the budget for the concert or minimizing the waiting time of the spectators. The subjects had to position a series of icons on a campus map: a number of 'P' signs to mark the chosen parking lots, three stage icons and six small circled numbers, one for each event to be allocated (part (*b*) of figure 5.1). The positions of these icons were not synchronized across the participants' displays: a subject could not see where the other would position her icons. This task was artificially made complex (*e.g.*, not WYSIWIS[1]) so as to augment the difficulty in finely positioning the icons between the two screens and so that we could observe how arising conflicts could be solved at a linguistic level.

The main focus of this experiment was to study explicit referencing mechanisms and how they are used depending on the communication tool. Hence, there were two different experimental conditions: the MSN condition (see figure 5.1, left) in which collaborators were provided with a simple MSN chat tool and the ConcertChat condition, in which they could use a special chat tool, ConcertChat (see figure 5.1, right), allowing them to link messages to specific points on the shared workspace. Each pair of collaborators was assigned randomly to one of these conditions.

[1]The acronym WYSIWIS stands for "What You See Is What I See" and refers to a paradigm in the design of multiuser interfaces where multiple users share the same visual perception of the work area.



(a) MSN condition



(b) ConcertChat condition

Figure 5.1: Shoutspace experiment setup in the two conditions: (*a*) feedback tool; (*b*) icons used during the task; (*c*) MSN chat message window; (*d*) reminder of the task goals; (*e*) map window; (*g*) ConcertChat chat message window; (*k*) example of how a stage icon is positioned and two concerts are assigned to that stage; (*v*) score button and countdown timer; (*z*) in ConcertChat it is possible to connect the message window to a point of reference with an arrow.

Participants

Hundred-and-twenty students (55 women and 65 men) of the Swiss Federal Institute of Technology in Lausanne volunteered to participate in the experiment. The subjects did not know each other and were randomly matched from different faculties. Students from Architecture or Civil Engineering were excluded because they could have biased the results as they are more used to working with maps. Each participant was remunerated 30 Swiss Francs.

Procedure

On arrival, participants were each given an instruction sheet containing the rules they had to respect in placing the elements on the map, information on how to evaluate their solution, and the principles behind the calculation of the score. Then, they were asked to watch a short video summarizing the paper instructions and explaining the particular communication tool they had to use to collaborate. Finally, they could ask questions to the experimenter if they had any doubts about the instructions.

During the task, participants had at their disposal: a feedback tool (part (*a*) of figure 5.1), a map of the campus (part (*e*) of figure 5.1) and a chat application to communicate with the partner. The feedback tool offered a score button (part (*v*) of figure 5.1), to display a number between 0 and 100. This score was computed by comparing the proposed solution with the optimal solution that was calculated once for all the experiments. This tool also presented four graphs that display four sub-scores one for each goal as well as the combined team-score. Each graph presented a horizontal red line, representing the maximum score that could be achieved with the given constraints and a vertical red line marking the time limit of the task. The tool also showed the remaining time to complete the task in the bottom-left corner (part (*v*) of figure 5.1).

The task lasted 45 minutes. As the task required multiple optimizations, we allowed each pair to submit multiple solutions to solve the task, ultimately selecting the best score for each team. Pairs were instructed to follow a collaborative paradigm. In fact, the pairs were warned that every time they pressed the score button, the system compared the position of the icons between the two machines and that the number of differences found was subtracted from the obtained score. They were also advised to take advantage of the feedback tool and of the available time to test the maximum number of different configurations.

Data collection and analysis

Gaze data were recorded using setup 1 of section 4.2.2. In addition, all chat messages were logged as well as all the actions of the collaborators, such as the use of the feedback tool. Gaze data of both subjects, actions and chat messages, were synchronized afterwards by detecting correspondences between chat events and input events synchronized with gazes (see section 4.2.2, setup 1). The final score of each pair was also logged as a performance measure.

5.2.2 KAT

The task (Sangin et al., 2011; Sangin, 2009; Sangin et al., 2008) consisted in building a concept-map[2] collaboratively to synthesize a text about "Neural transmission" read by both of them individually beforehand (see figure 5.2). The goal was to understand the concepts presented in the text by sharing their knowledge while constructing the concept-map

This experiment was designed to investigate the effects on the collaborative learning process of a technical support providing co-learners with cues about their partner's prior knowledge. More specifically, it aimed to study how such an awareness tool would affect their perception of their partner's knowledge (mutual model) and how this would change their way of interacting. There were two experimental conditions and each pair was assigned to one of them. In the KAT condition, they were provided with a knowledge awareness tool (see figure 5.2 bottom part) which indicates how well their partner had understood the text under concern at the pretest, while in the control condition, they had no such tool.

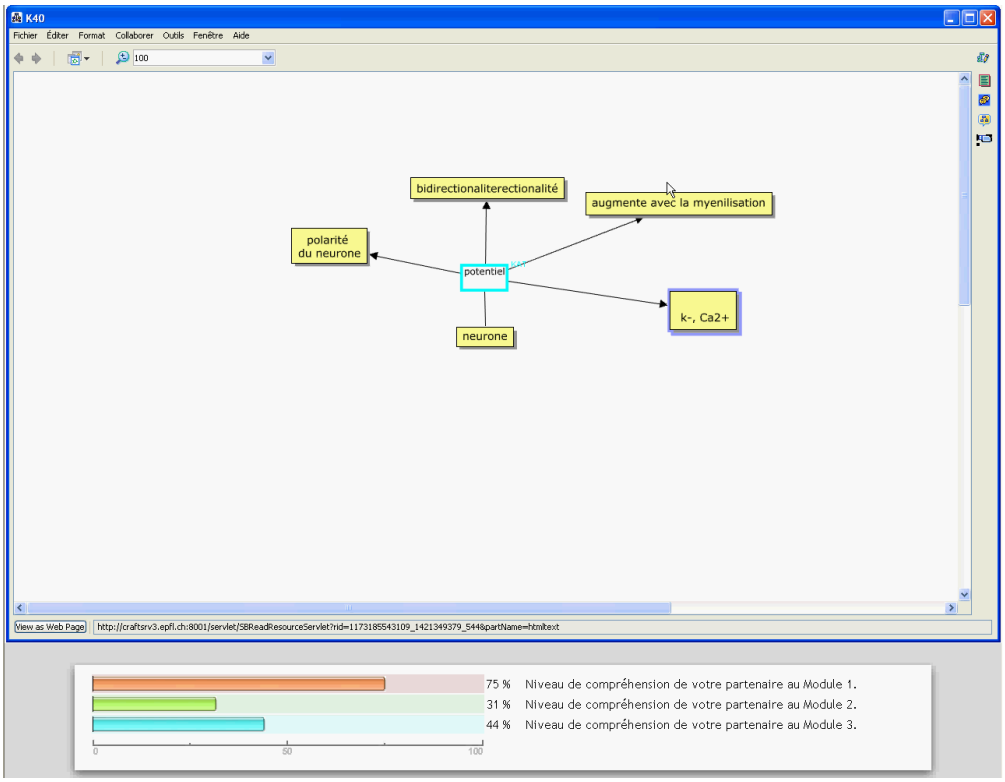


Figure 5.2: Screenshot of the KAT experiment. We can see the beginning of a concept-map. On the bottom part, the knowledge awareness tool indicates the level of expertise of the partner. This tool was not present in the control condition, leaving this space empty.

[2]Concept-maps are simple diagrams consisting of boxes representing concepts and labeled links representing relations between concepts.

Participants

Sixty-four first semester university students (18 women and 46 men) were recruited and remunerated to participate to the study. Learners with a high degree of knowledge about the instructional material (i.e. the neural transmission) were filtered through a prior knowledge test and were excluded from the sample. Peers within same pairs did not know each other before the experiment.

Procedure

The two subjects were installed in front of two identical computer setups running the instructional material. The instructional material consisted of an explanatory text about the phenomenon of neural transmission, based on textbook materials and developed with the help of two experts of the domain. During the collaborative phase, we used an online concept-map building software, CmapTools[3]. The experimental session lasted about 90 min and consisted of six phases including two main learning phases: an individual explanatory reading phase and a remote collaborative concept-map building phase. We briefly describe the main phases:

First, to detect and remove data from potential experts of the domain, we asked the participants to write down everything they knew about the phenomenon of neural transmission through an open-ended question. Participants had four minutes to answer this prior knowledge verification question. After this verification, participants were asked to carefully read the instructional texts in order to understand and remember as much as they could about the neural transmission. They had 12 min to complete this phase and were provided with a countdown in the form of a reverse-bar. After the individual reading phase, the first learning test was administered. It consisted of an automated multiple-choice questionnaire of 30 items. The questionnaire comprised two types of questions: six multiple-choice questions and 24 inference verification questions.

Then, participants were provided with instructions about the collaborative concept-map building phase and a short video tutorial (2:34 min) on how to use CmapTools. During the collaborative task, participants were invited to draw a collaborative concept-map about what they learned during the individual task. This phase lasted 20 min. Peers were able to communicate orally using a headset. After the collaborative task, the second learning test was administered. It consisted of the same items as the pretest, presented in a different order. Finally, participants were provided with a knowledge estimation questionnaire. We asked the participants to estimate their partner's knowledge with a 7-point scale.

[3] IHMC: http://cmap.ihmc.us/

Data collection and analysis

Gaze data were collected using setup 1 described in section 4.2.2. In addition to the gaze and speech of both subjects, all actions on the concept-map were also logged by the collaborative concept-mapping tool. These were logged within the same file and with the same time base for both subjects. A post-synchronization was performed first to match the time of the gaze with the time of the concept-map by finding correspondences between specific concept-map events and input events logged by the eye-tracker software (see section 4.2.2, setup 1). A second post-synchronization was also accomplished to match the time of the audio recordings and the time of the gaze data. Finally, a spatial synchronization was accomplished as the two users could scroll independently in the shared workspace, which made their gaze coordinates not directly comparable.

Speech recordings were transcribed and coded manually by two independent coders. The coding scheme used was designed to answer specific questions about the use of the knowledge awareness tool and was focused on the following aspects of the verbalization: elaborative information exchange (i.e. information seeking and providing), conflict resolution, knowledge negotiation, mutual regulation and mutual modeling. The corpus was first and foremost divided into four main categories regarding the content of the utterances: domain knowledge, collaboration management, mutual knowledge modeling and miscellaneous. This resulted in 23 coding categories in total.

Finally, for each subject, the difference of scores between the pre-test and the post-test was used as a measure of the relative learning gain which was considered as a performance score.

5.2.3 Raven-Bongard

This task (Nüssli et al., 2009) is composed of two parts, each consisting of five small logical problems to solve collaboratively. The first kind of problems, namely the Raven progressive matrices (see figure 5.3 top-left), consists in finding out the last element of a 3-by-3 matrix which exhibits certain logical patterns over its rows and columns. The second kind of problem is called Bongard problems (see figure 5.3 top-right). The goal of these problems is to find a common pattern or rule among the six images on the left (examples) which doesn't work for the six images on the right (counter-examples). What makes these problems quite hard is that the rule may involve different kinds of image features. It may be the relative position of the objects, their relative size, the orientation or it may also be a kind of higher level shape formed by lower level shapes.

The two kind of problems described above have been slightly modified for the purpose of this study. In order to make them more interactive, the images have been split between the two participants. For the Raven matrices, only six (out of nine) cells were shown to each participant, the other cells remaining blank. One subject saw only the upper-right part of the matrix while the other one saw only the lower-left part of the matrix. Thus, they each had

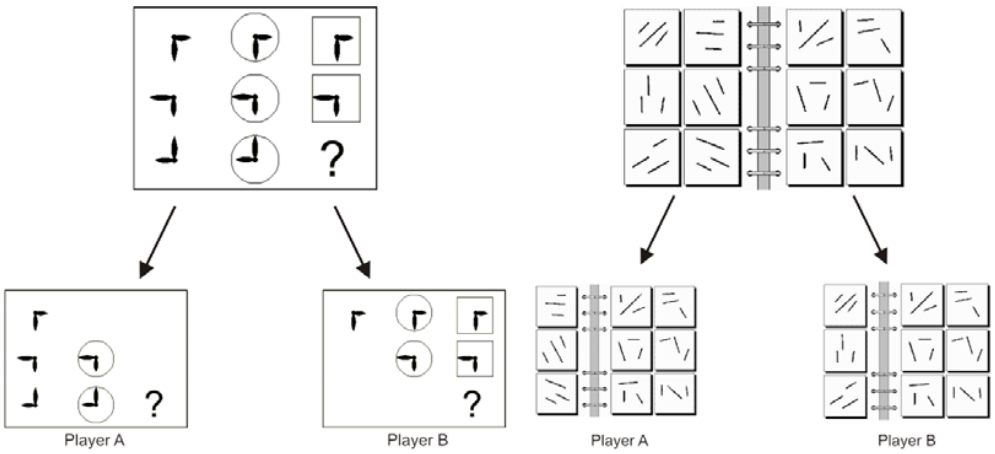


Figure 5.3: The two types of problems that were used in the RavenBongard experiment and how they have been modified to become more collaborative. Left side is a Raven matrix example, for which the answer is "clock indicating nine o'clock inside a square" as there is a shape progression along the row and a clock rotation along the column. Right side is a Bongard problem example for which the rule is "the lines are parallels" and no rule for the right side.

three personal cells which were not seen by the other participant and three shared cells (on the diagonal). One of the shared cells was the missing cell which had to be discovered by the collaborators. For the Bongard problem, the split was a bit different. Each participant could see the six counter-examples (right images) but each participant saw only three out of the six examples (left images). Thus, they had six shared cells and three personal cells.

This experiment aimed to study eye-movements patterns in the context of joint construction of abstract representations. This was an exploratory study with no specific experimental conditions.

Participants

Nine dyads (10 men and 8 women) were recruited among campus collaborators and students. Subjects' ages vary between 17 and 53 with a median of 27 years. They were asked whether they already heard about Raven matrices or Bongard problems and none of them was aware of any of the two.

Procedure

Subjects were first asked to fill in a short questionnaire about general information like age and sex and how much they know each other. The experiment was composed of 12 static images which could be passed by simply pressing the spacebar at least one time on each computer.

The first and the seventh slides were instructions for the Raven matrices problems and the Bongard problems respectively. Slides 2 to 6 presented the Raven matrices and slides 8 to 12 were the Bongard problems. The problems were in order of increasing difficulty in order to allow subjects to familiarize themselves with the problems. The subjects had to solve the problems together, agree on a solution and then, say it out loud before pressing the spacebar to go to the next problem. There was a maximum time limit of 5 minutes for each problem.

Data collection and analysis

Gaze data were recorded using setup 2 from section 4.2.2. Hence, gazes of both collaborators were already synchronized as they were started simultaneously. The logging of the actions and stimulus changes as well as speech recordings was done by the eye-tracker software, ClearView, so that their timestamps were already synchronized with gaze data. The correctness of the answer given by the subjects was assessed manually by the experimenter. The number of correct answers in each category of problems was used as a performance score.

5.2.4 Collaborative Tetris

We implemented a multiplayer, collaborative, version of the classical Tetris game (see figure 5.4). This game consists in positioning falling puzzle pieces so that they match at best and that no holes appear between or under the pieces. In this version, two pieces fall at the same time in the same game area, one on each side of the area. Each player can control one of the two pieces (either the left piece or the right piece) and they can execute the usual Tetris movements: left or right translations, rotations, down accelerations and drop. Of course, the game prevents collisions between the two pieces, thus producing novel situations compared to the original game. For example, if a player drops a piece while the second player has its piece right below, the drop of the first player's piece will be stopped just above the blocking piece. Note that, like we can see on figure 5.4, the two pieces do not necessarily fall synchronously as some players may play their pieces faster than their partner. The game is collaborative and thus, both players share the same score independently of who plays what.

This experiment (Jermann et al., 2010) is also an exploratory study about eye-movements patterns in collaboration. The task has been chosen because on the one hand, it is very visual and implies a lot of coordination between players and on the other hand, it has a fun aspect which motivates subjects to do it well.

Participants

Sixty-four dyads were recruited among campus students. They all reported to have played Tetris before. They were remunerated with pizza-coupon worth 10 swiss francs.

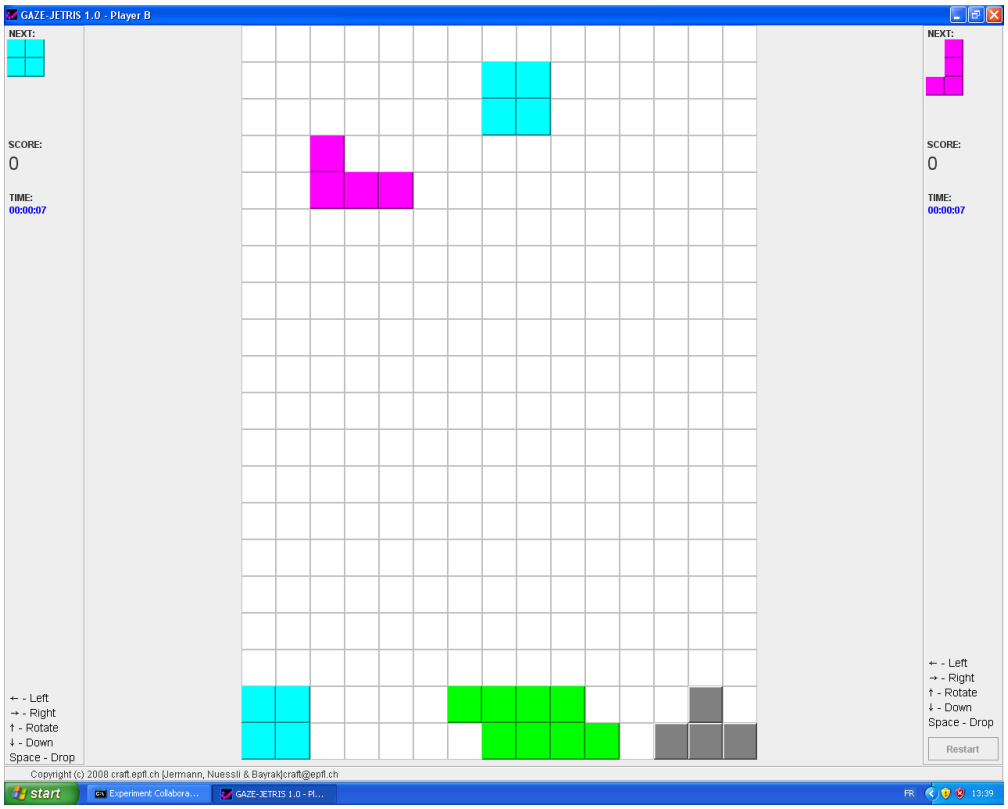


Figure 5.4: Screenshot of the collaborative tetris game. Two pieces are falling simultaneously and each player controls one of them.

Procedure

Subjects were first briefly instructed about the commands to control the game. Then, they played normal Tetris game individually in a training phase for 5 minutes. Then, they played in pairs for 10 minutes in a collaborative training phase and finally, they played again together for 15 minutes in a collaborative testing phase. All phases were separated by a one minute pause.

Data collection and analysis

Gaze data were recorded with setup 3 of section 4.2.2. Player actions and speech were all recorded through the same application that was also responsible for the gaze collection as well as of the game engine. Hence, all data were already synchronized. The maximum score achieved during the individual training phase was used to determine the level of expertise of the player.

5.3 Collaboration quality

We now analyze the experiments with regards to their high level factors. Collaboration quality can be evaluated on two aspects: process or product. Unfortunately, collaboration process quality is much more difficult to assess. Indeed, first, it is not well defined what characterize a good collaboration compared to a bad collaboration, and secondly, it often relies on subjective judgments from experts which makes the rating process both difficult and possibly biased. Hence, we will focus on the quality of collaboration product, i.e. what results from the collaborative activities. This is generally a performance score for collaborative problem solving activities but in the case of learning activities, this can be the learning gain of the collaborators, i.e. how much they learnt from the collaboration between pre and post-test.

5.3.1 Performance and recurrence in Shoutspace

In the Shoutspace experiment, we were interested in the relationship between the gaze coupling and the performance at the task (Cherubini et al., 2010). More specifically, we asked whether pairs that succeeded well in the task had a higher degree of gaze coupling, i.e. of cross-recurrence.

Method

We conducted a cross-recurrence analysis (see section 4.3) on the gaze data during the whole experiment. To this end, we sampled the fixations of each subject every 200 ms in order to obtain a regular continuous time-series of gaze coordinates and we used the resulting sampled streams to compute a CR plot (4.3.1). We used a recurrence threshold of 30 pixels which corresponds roughly to the average size of the polygons on the map. Figure 5.5 shows the average recurrence rate per lags for each pair. As the collaborators were in purely symmetric situation, we considered in the same way positive and negative lag values. Hence, we had only absolute lag values and the recurrence rates are the average between the rates for the positive and the negative lags. Graphically, this consists in folding the curve around lag 0 and averaging the two superimposed curves. Finally, in order to account for differences of collaboration strategy between pairs, we corrected the average recurrence rates by subtracting the random level obtained by shuffling the same gaze data (see section 4.3). We measured two characteristics of the peak of each experiment's recurrence curve: the maximum recurrence for any lag lower than 60 s and the average recurrence between lags of 0 s and 60 s (many messages took one minute to be composed since we used written communication).

Results

We computed a linear regression of the score in relation to the two measures of recurrence. The regression of the maximum recurrence was a good fit, describing 56% of the score variance (R^2_{adj}= 56%). The overall relationship was statistically significant (F[1,11]=12.60, p<.05). The

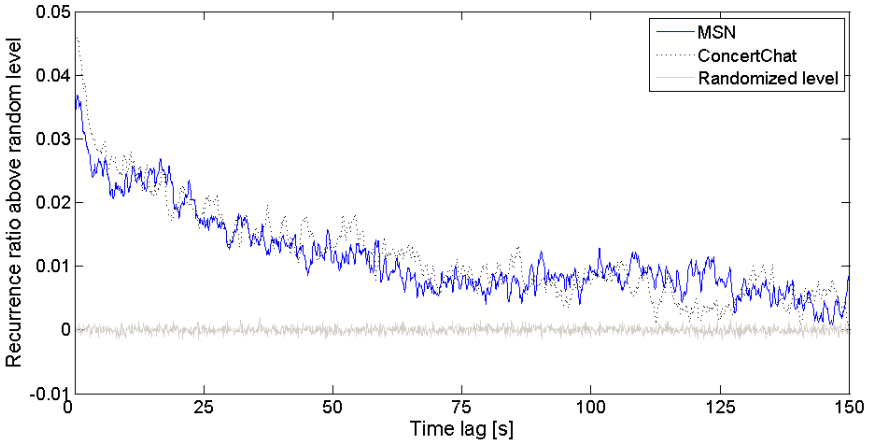


Figure 5.5: Average cross-recurrence for each experimental condition between 0 and 150 seconds. As collaborators were in a symmetric situation, curves have been folded around lag 0 and the average recurrence rate between positive and negative lag has been computed.

pair score was positively related with the maximum cross-recurrence, increasing by 1.48 points for every extra percent of recurrence (β_{std}=.75, p<.05). This implies that the more the gaze movements of the collaborators are coupled the higher performance their interaction may reach. Figure 5.6 shows the relation between the maximum recurrence and the score. The average recurrence calculated between 0 seconds and +1 minute was also positively related with the score (R^2_{adj}=.17, β_{std}=.49, p<.05). There was no effect of the condition, i.e. the availability of deixis mechanism in the chat tool, on the level of cross-recurrence.

Discussion

The regression reported in these results shows that the degree of coupling of eye movements was related to the performance obtained by the pair. This implies that pairs who look more often at the same thing at the same time, or with a constant lag, obtain higher scores, even when people are not co-located. This effect is explained to some extent by messages containing explicit references. Indeed, further analyses showed that there was a significant correlation between the average level of recurrence and the number of explicit references done by the pair. When doing explicit references, participants look at the place they refer to, which causes recurrence at lags corresponding to the time between the writing and the reading of the message containing the reference.

This result extends the findings of Richardson and Dale (2005). While they found a correspondence between the degree of cross-recurrence and the scores of a post-hoc comprehension questionnaire in a simple comprehension task, we measured the relation of cross-recurrence and performance in a collaborative problem-solving task. Furthermore, while they proved this correspondence in the case of oral communication, we could verify the same finding in

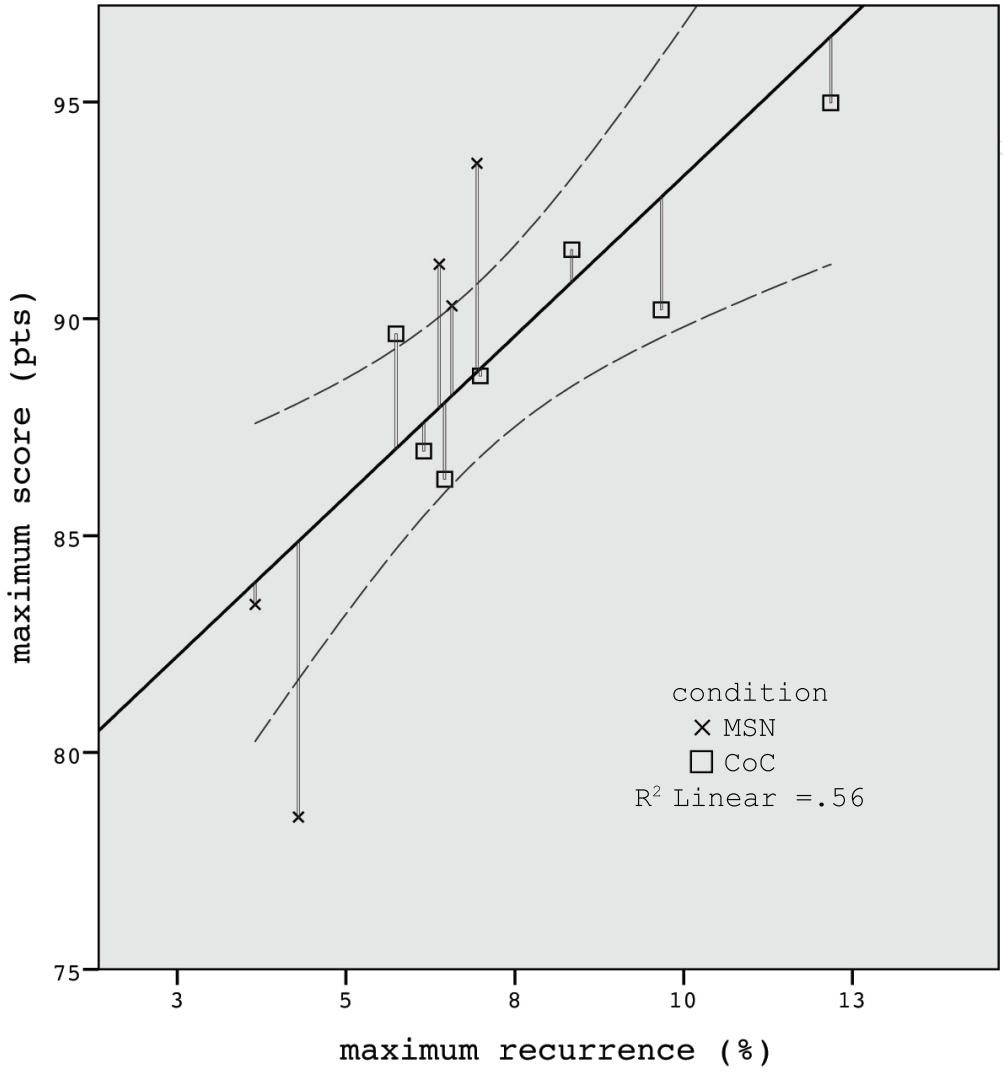


Figure 5.6: Scatter plot of the maximum recurrence and the maximum score with the regression line. The curves indicate the mean confidence intervals

the case of written communication. This shows that this simple measure of eye-movements coupling and its relation with collaboration quality can be extended to very different contexts.

5.3.2 Success in Raven-Bongard

In the RavenBongard experiment, we explored the possibility of building a predictive model of collaboration outcomes using eye-movements (Nüssli et al., 2009). We used machine learning algorithms to predict success in solving the task from dual gaze indicators, possibly combined with speech indicators.

Method

We defined four gaze features to represent how subjects looked at the cells composing the problems. We identified every sequence of at least 3 fixations with at least one back and forth movement between one cell and another. We defined the *comparisons* feature as being the ratio of all fixations which belong to such sequences. A related feature is the *comparison intensity*. For each of these comparison sequences, we computed the number of transitions between the two cells and then, we averaged this number over all comparison sequences. We also computed the standard deviation of a vector representing the 9 cells on the screen and containing aggregated fixation times to obtain the *gaze dispersion* feature. Finally, we computed the cosine between such vectors of aggregated fixation times of each subject to obtain a dual gaze feature called *gaze divergence*. Examples of these features are shown on figure 5.7.

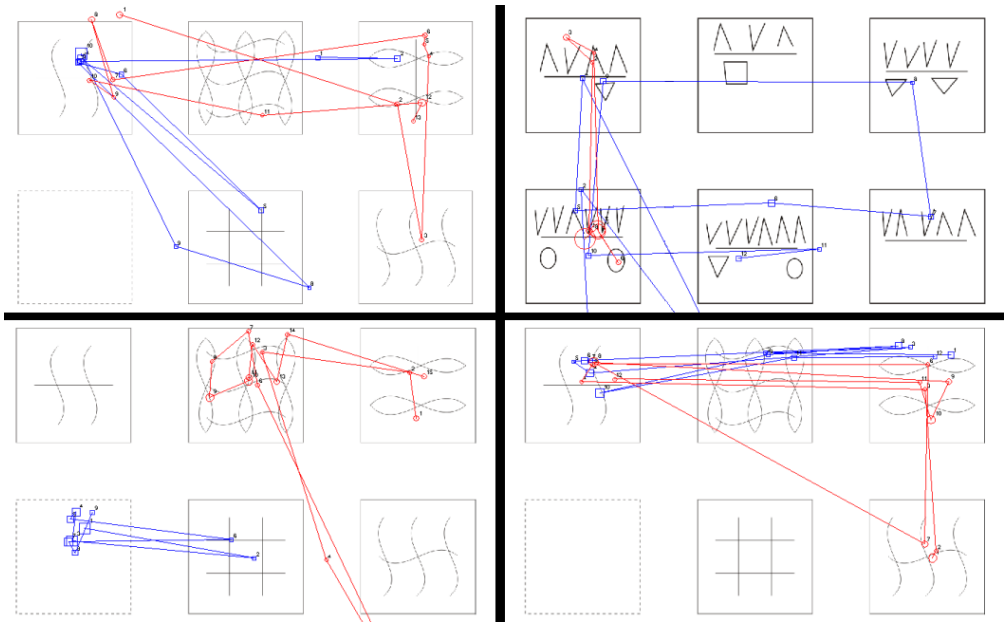


Figure 5.7: Illustration of gaze features. Top-left picture depicts one subject (blue) doing an intense comparison between the upper-left cell and middle cell and the other subject (red) doing a weak comparison between the upper-right cell and middle-right cell. On the top-right picture, we can see a dispersed subject (blue) and a not dispersed one (red) Bottom images illustrate high gaze divergence (left) and low gaze divergence (right).

The recorded audio was automatically labeled, second by second, as speech or no speech according to a simple threshold test on the volume. From this, we extracted two simple features: the *speech time*, i.e. ratio of time spent at speaking, and *speech time asymmetry*, i.e. difference of speech time between subjects.

We used these features to train two types of classifiers: Binary decision tree and Naive Bayesian classifier. These two types of classifier were chosen because it is possible to analyze the

resulting model in order to get insights on the way the prediction is done. The predicted variable was the outcome of the problem: solved or unsolved. Algorithms were fed with gaze and speech features computed over one minute time windows and the time of the window was also used as a predictor. However, we discarded for each problem the last minute before the solution was announced in order to avoid the effect of speech which may due to the explanation of the solution. We used a 10-fold cross-validation procedure to assess the performance of prediction. We made three different prediction experiments: one to predict the success independently of the task and two to predict the success in each of the two classes of problems separately.

Results

Table 5.2: Results of the machine learning algorithms for both problem classes combined. The values in parenthesis are the corresponding kappa scores.

	Speech	Gaze	Both
Both problems classes			
Naive Bayesian classifier	77% (45 %)	74% (35%)	78% (50%)
J48 Binary decision tree	86% (65%)	68% (10%)	79% (51%)
Raven problems only			
Naive Bayesian classifier	78% (56%)	68% (32%)	78% (56%)
J48 Binary decision tree	91% (81%)	68% (34%)	91% (81%)
Bongard problems only			
Naive Bayesian classifier	76% (29%)	77% (37%)	76% (34%)
J48 Binary decision tree	75% (21%)	77% (37%)	75% (25%)

The prediction results are presented in table 5.2. We indicate both the % of correct predictions and as this is not a balanced classification task, the kappa statistic, which can be considered as the % of correctness above chance level. First, it is very interesting to note that we can obtain good results (50% above chance level) when predicting success in both tasks at the same time. This suggests that there may exist some patterns in gaze and speech which are partially task independent. We can also see that at this level, speech features play a larger role than gaze. Indeed, we see that models using only gaze features are the worst for both algorithm types. However, for the Naive Bayesian classifier, gaze seems to slightly improve the performance compared to speech only, indicating that it can still play a role. The results concerning Raven problems only are surprisingly high, producing up to 80% above the chance level with 91% of correctly classified instances. Moreover, we can see that these results are explained only by speech features. For Bongard problems, the situation is the opposite, although the results are much lower than for Raven and even lower than for both classes combined. This suggests that the correct predictions made on both problem classes taken together are essentially explained by correct predictions of Raven problems. However, it is interesting to note that the best prediction performances for Bongard problems are explained mainly by gaze features, as the best models are achieved by taken only gaze features without speech features.

We also tried to make predictions by using only the data from the first two minutes of solving, the results were either similar to those presented or a little bit (3 or 4%) lower. These results are even more interesting because they suggest that it could be possible to detect after one or two minutes if the pair will succeed or not. Moreover, we also tried to predict if success will happen the following minute. In this case, the results are clearly lower than the previous ones but they are still sufficiently high to be considered. We obtain kappa-scores of 40% (instead of 50%) for both problems combined. It suggests that there exist some phases in the solving processes which are distinguishable by using gaze patterns and this is consistent with the results found using usual statistical methods.

Discussion

As we have seen, we can predict up to a certain point collaborative problem solving outcomes by using only raw measure of speech and dual-gaze features. Moreover, we see that we may be able to predict the moment of resolution one minute before it happens, suggesting a certain ability to detect phases in the collaborative process. These results have interesting implications as they tend to show that it could be possible to build gaze-sensitive applications, possibly combined with simple automatic speech analysis, in order to provide meaningful feedback to users. Of course, these results must be taken with care as the number of subject is low.

5.4 Expertise and group composition

In this section, we are interested in characterizing expertise level from eye-movements patterns. It has been shown that often experts have very different eye-movement patterns than novices (see for example, Law et al. 2004). Another interesting question is how the group composition, i.e. the expertise levels of both collaborators, affects eye-movements. For example, does an expert working with another expert has different patterns than an expert working with a novice.

5.4.1 KAT experiment

In collaboration with researchers from the IBM Watson Research Center, we conducted analyses about the prediction of prior knowledge levels of subjects in the KAT experiment (Liu et al., 2009).

Method

We chose to use hidden Markov model (HMM)[4] to predict the expertise level of participants from their gazes. This was motivated by the fact that such models are particularly well adapted

[4] Hidden Markov models are statistical models consisting in a set of hidden states with some transition probabilities between the state. Each state has also associated emission probabilities for the different kind of observations.

to sequential data and also because it is possible to analyze the resulting model. Participants were separated into two groups, low ability or high ability, according to their score at the pretest. All concepts of all concept maps were then clustered using a cosine similarity measure between concept names so that similar concepts form a single category. Fixations were then associated to their corresponding cluster and the resulting sequences of fixated clusters were considered as observations to train a hidden Markov model. We trained two HMM, one with gazes of the low-skilled subjects and one with the high-skilled subjects' gazes. We divided the 52 sequences into training and testing sets of different size. Given the limited amount of data, we repeated each analysis 30 times with randomized training sets and reported average results as well as standard deviation. We simply used prediction accuracy for the evaluation, i.e. the % of correct predictions, as our task was a balanced classification problem.

Results

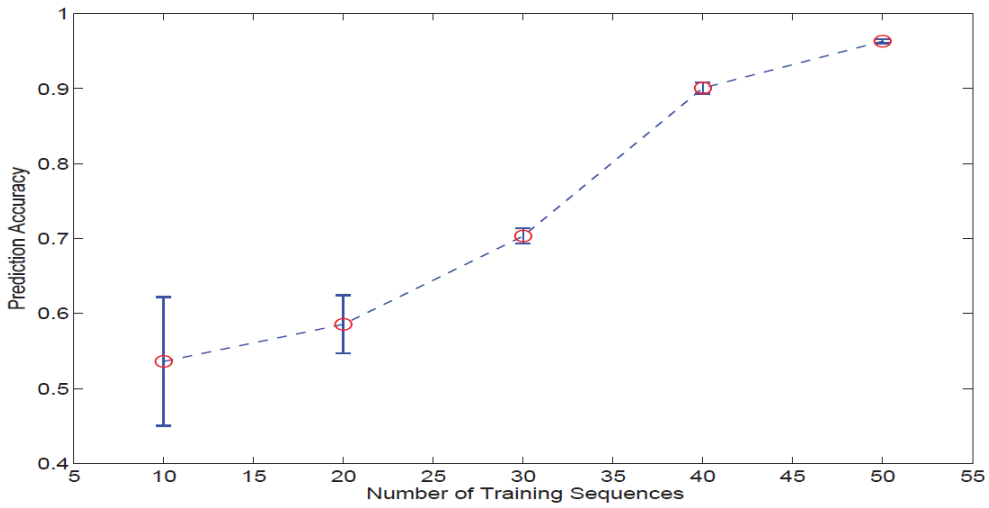


Figure 5.8: Skill level prediction accuracy changes with the number of training sequences

Figure 5.8 shows the prediction accuracy as a parameter of the number of training sequences. The accuracy improves dramatically with the number of training sequences. The best result is achieved with 50 training sequences, reaching the accuracy as high as 0.96. We also tried to test the resulting model using only partial sequences. With only 5% of the data, we were able to achieve almost similar performance as using the whole dataset. This corresponds to around 1 minute after the experiment starts.

Finally, by inspecting the resulting model, we found that the higher skilled users tend to focus on more content-oriented concepts while low-skilled users are easily trapped in the uncertain concepts, i.e. concepts that had vague or undefined names, and spend too much time on them.

Discussion

Our findings confirm the feasibility of using eye gaze data to determine the relative expertise level of paired users in collaborative learning settings. Given the automatically generated concept clusters and HMMs, we are able to achieve high accuracy (96%) and make predictions as early as 1 minute into the experiment. These are encouraging results for the development of gaze-aware tools that could give real-time information to collaborators about the relative level of knowledge. We also identified distinctive gaze and lexical patterns that might contribute to the success of our models and these identified patterns coincide with previous findings on expert-novice eye gaze (Law et al., 2004). For example, while higher-skilled participants concentrate longer on specific concepts, lower-skilled participants have shorter attention spans and scattered gazes.

5.4.2 Tetris experiment

In the Tetris task (Jermann et al., 2010), we analyzed how the group compositions affect eye-movements patterns. We first pointed out statistical differences in their patterns of actions and eye-movements and then, we built machine learning classifiers to do predictions about group compositions using the same features. We also compared our prediction abilities when using only gaze features, only actions features or both to find out which aspect is the most predictive.

Method

Players were split in two groups, novices and experts, which resulted in four possible group compositions, EE, EN, NE and NN. Note that the first letter represents the player under consideration, while the second letter corresponds to its partner. Hence, EN represents an expert player playing with a novice, while NE represents a novice playing with an expert. We computed several gaze features which are the percentage of fixation time on the different parts of the game (self's piece, other's piece and stack of already fallen pieces) as well as simple properties of fixations and saccades: average fixation duration, average saccade length and the ratios of horizontal and vertical saccades. Action features are composed of the frequency of the different types of basic actions (translations, rotations, drops) done by each player and higher-level features representing how fast and how direct players played their piece. These higher-level features are the horizontal directness, which measures the amount of additional useless horizontal movements that have been done to place the piece correctly, and the vertical acceleration, which measures whether and how much players accelerated the fall of their piece.

Results

We first present results from a principal component analysis which allowed us to reduce the features space in two dimensions by still retaining almost half of the variance and to understand what variables are affected by the group composition. In a second part, we show results of classification using machine learning algorithms on the same features.

Principal Component Analysis We conducted a Principal Component Analysis (PCA) to investigate the structure of the various gaze and action features. The analysis included five gaze related variables as well as three action related features. We retain two dimensions with eigenvalues higher than 1 for the presentation of the results. The first dimension accounts for 23% of the total variance, the second dimension accounts for 20%. The correlations between the initial variables and the orthogonal dimensions of the PCA are represented in figure 5.9. The correlational structure uncovers two rather independent aspects of the behavior. We use the mapping of group compositions variables onto the two components to guide the interpretation of the dimensions (see figure 5.10).

The first dimension reflects more the individual style of play and could be seen as piece control versus solution search. Two gaze features are strongly correlated with the first dimension. It represents the opposition between fixations on the players' own piece (self) and fixations on the contour. The players fixate their own piece when it first appears as well as when they "drive" it through rotations and translations down towards its final position. They fixate the contour to search for a "good" spot to places their piece, as is suggested by the correlation of the ratio of horizontal saccades with the dimension. In figure 5.10, this dimension opposes novices (NN) and experts (EN and EE). Interestingly, NE players are situated to the left of NN players, indicating that when novices play along with an expert, they fixate their own piece less than when they play with another novice.

The second dimension has a more collaborative aspect and represents speed versus coordination. Vertical acceleration has the strongest correlation with the second dimension. Horizontal directness also correlates positively with the dimension, indicating that few superfluous translations are done. Rotations and fixations on the partner's piece correlate negatively with the dimension, indicating a slower, but more collaborative pace of play. In figure 5.10 experts (EE and EN) are positioned higher on this dimension than novices (NN and NE), although it appears that EN experts slow down because they play with novices.

Automatic predictions We then experimented with the prediction of the roles of the player and their contexts using both generative Gaussian Mixture Model (GMM)[5] and discriminative

[5]GMM (Bishop, 1995) is a parametric probability density estimation technique which has been successfully used to deal with the speaker verification and speaker recognition.

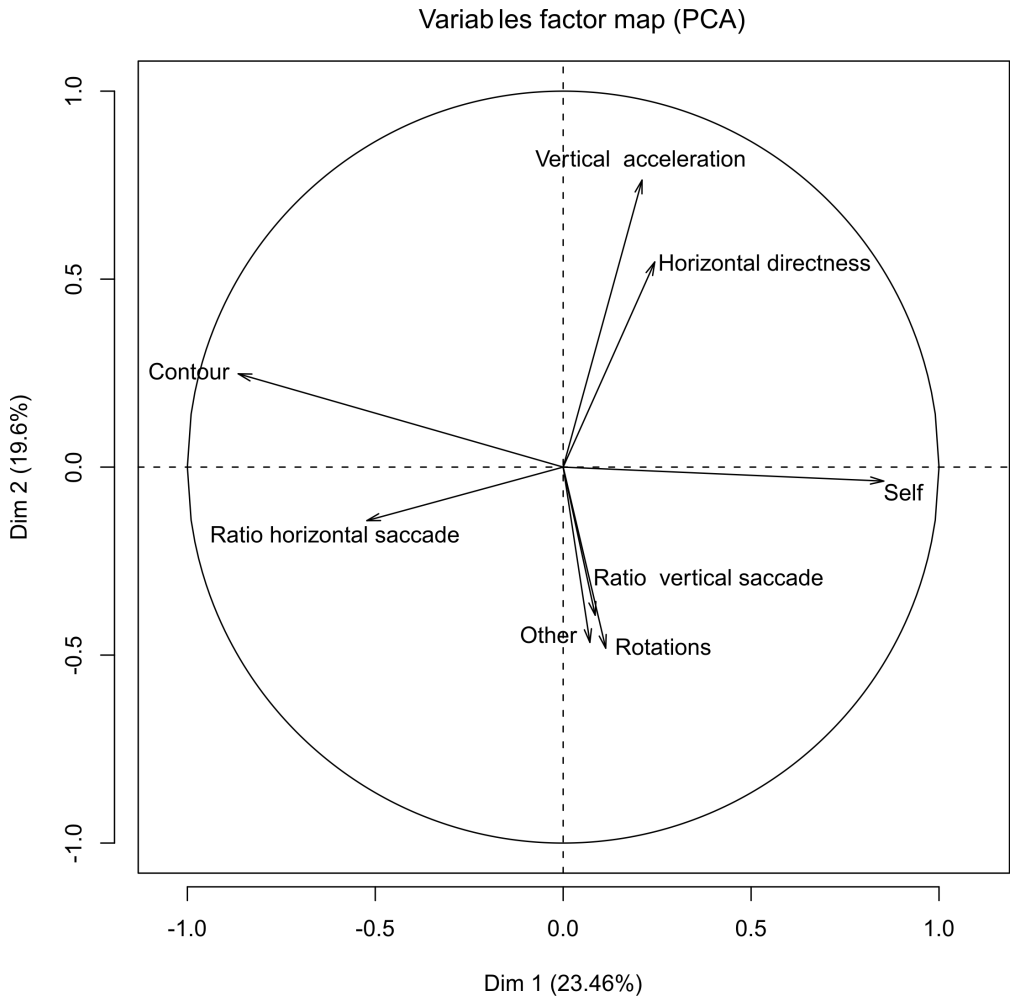


Figure 5.9: Correlation circle for the two first dimensions. The arrows represent the correlation between original variables and the dimensions of the PCA.

Support Vector Machines (SVM)[6]. We present here the classification results using these two algorithms because they appeared to be the most effective algorithms in this situation compared to others that we tested, such as HMM or binary decision tree. In our experiments, each group composition category (i.e. NN, NE, EN and EE) is represented by a GMM and is referred to as model. To test data, the likelihood of data is computed across all these GMMs, and the one providing the largest log likelihood is found as the recognized group composition. For support vector machines, as they are originally binary classifiers, we used Max-win strategy for multi-class recognition.

Different sets of features were used for the recognition of group composition: action-based features, gaze-based features, and their combination. The whole data for each group com-

[6]SVM (Cristianini and Shawe-Taylor, 1999) belongs to the class of maximum margin based discriminative classifiers. They perform pattern recognition between two classes by finding a decision surface that has maximum distance to the closest points in the training set which are termed as support vectors.

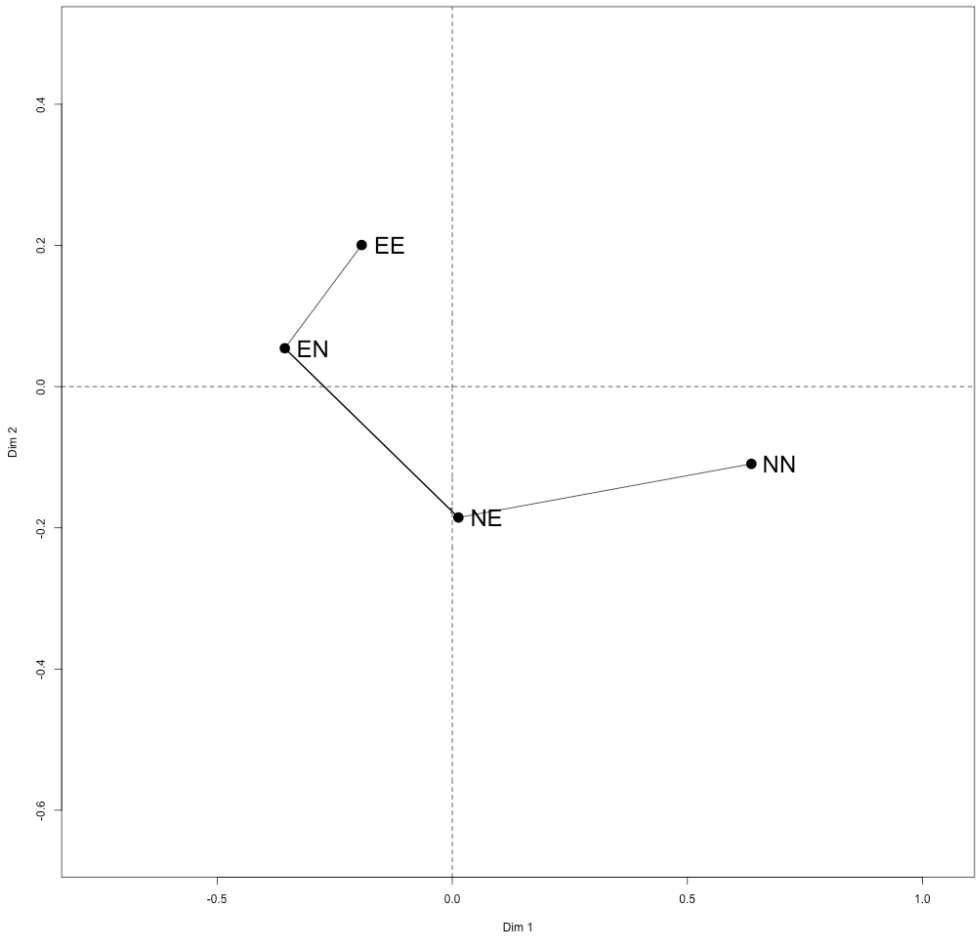


Figure 5.10: Centroids of the group compositions variables in the space defined by the two first principal components.

position was divided by five folds, and five-fold cross validation was used for the evaluation. Table 5.3 summarizes the recognition performance using different classifiers and different feature sets, in terms of prediction accuracy (% correct predictions) and Kappa statistic. Gaze-based feature sets performs slightly better than action-based feature sets. This demonstrates that gaze-based features are more important for classifying different group compositions. Combining the two features sets significantly improves the recognition performance, which shows that the information from the two feature sets are complementary. It can be also found that SVM performs better than GMM. This might be explained by the fact that SVM is a more discriminative model than GMM which is based on the estimation of the probability density function (pdf) statistically[7].

[7] In order to achieve accurate pdf estimation, generally large amount of training data is needed.

Table 5.3: Recognition results using different features in terms of correct recognition rate (CRR) and Kappa coefficient (the higher the better).

	Action-based	Gaze-based	both
GMM			
Accuracy [%]	51.34	56.62	70.84
Kappa	0.34	0.42	0.63
SVM			
Accuracy [%]	54.45	60.12	75.28
Kappa	0.39	0.49	0.68

Discussion

Results of the Principal Component Analysis show that players adapt their behavior and gazes to the social context of interaction. A possible explanation is that experts tutor novices by telling them where and how to place their piece on the stack, thereby showing the novices the tricks of the trade. Another explanation is that experts feel responsible for avoiding collisions or solving conflicts. As a consequence, they play slower and monitor their partner's piece more than if they interacted with another expert. In parallel, novices who interact with experts look less at their own piece and monitor the contour more than if they interacted with novices.

Moreover, we have shown that these differences allow us to computationally detect the group composition of the players. We could thus envisage the design of adaptive gaze awareness tools. For instance, experts could see whether their novice partner looks at the place they are referring to. The expert could point the novice to the correct place by gazing and verbally signaling a deictic reference (e.g. saying "here").

5.5 Interaction episodes

We now report analyses at the micro-level, i.e. that use of dual-gaze features for the characterization of specific interaction episodes which are relevant for socio-cognitive processes of collaboration. Such episodes include explanations, elaborations, conflict resolution or mutual regulation, which are all known to play an important role in collaboration.

5.5.1 Coordination episodes in Tetris

In Tetris, we studied two specific types of episodes which require increased coordination efforts. Conflicts correspond to moments when the players potentially need to put their piece at the same place and crossings correspond to moments when the players had to inverse their side. We compared gaze patterns when there is such a conflict or crossing with moments when there is no conflict or crossing. We were interested in finding characteristic patterns of eye-movements that reflected these types of episodes.

Method

Conflicts are defined as moments when players have pieces that fit the same places and when the number of places that fit is small. More specifically, we define the number of positions that fit for the piece of one player, N_{self} (This value have an upper bound of 15, the total number of possible positions) and the number of positions that fit for both players, N_{shared} (This value cannot exceed the number of positions for one player, N_{self}). Then, we defined the *conflict* value for this player according to the following formula:

$$\text{conflict} = \left(1 - \frac{N_{\text{self}}}{15}\right)\left(\frac{N_{\text{shared}}}{N_{\text{self}}}\right)$$

Hence, this value is high if the player has few positions that fit and if all these positions also fit for the other player. Note also that the *conflict* value is defined at the player level and can be different for each player. For example, a player may have only few positions available for her piece and the second player may have many available positions including all the available positions of the first player. In this case, the first player will have a high *conflict* value, as the few places that fit also fit for her partner, and the second player will have a low *conflict* value, as she has other places that fit. We then defined conflict episode by detecting every moment where one player has a conflict value bigger than the median conflict value, which is 0.23.

Crossing is a simple boolean value indicating if a crossing between the two players (for example, the left player going on the right side of her partner) occurred. Thus, it allows to identify episodes which required a certain amount of coordination between the two players.

We analyzed how these two contextual variables affected gazes and actions of the players. More specifically, we studied the variables previously used to study group composition (see section 5.4.2). This included the ratio of fixation times on the various zones of the game, fixation duration and saccade length, horizontalness and verticalness for gaze features. Concerning actions, we analyzed the amount of basic actions, i.e. translations, rotations and drops, as well as directness of horizontal movements and vertical acceleration.

Results

We fitted linear mixed effect models to assess the influence of conflict on the action and gaze features (see table 5.4). It appears that in case of conflict, subjects play slower and do more translation moves. They also focus relatively more on the contour, less on their own piece or their partner's piece.

We also conducted linear mixed effect analyses of variance to investigate the effect of crossing on the action and gaze features (see table 5.5). While crossing, players rotate their piece more often, vertically accelerate less, and make more horizontal moves. The coordination that is required when crossing pieces also impacts the fixation targets, as players fixate slightly less on the contour and slightly more on their partner's piece.

Table 5.4: Effect of conflict on action (upper part) and gaze features (bottom part). All F-tests performed with 1 and 2871 degrees of freedom. Numbers represent means.

Var	Conflict		ANOVA	
	High	Low	F	p
Rotations	1.7	1.8	3.5	NS
Vert. accel.	0.54	0.59	27.7	<.0001
Horiz. direct.	0.71	0.77	23.7,	.0001
Contour	0.56	0.51	36.6	<.0001
Self	0.3	0.33	10.9	<.001
Other	0.076	0.094	13.21	.0003

Table 5.5: Effect of crossing on action (upper part) and gaze features (bottom part). All F-tests performed with 1 and 2871 degrees of freedom. Numbers represent means.

Var	Crossing		ANOVA	
	No	Yes	F	p
Rotations	1.7	2	5.4	<.05
Vert. accel.	0.58	0.5	23.1	p<.0001
Horiz. direct.	0.75	0.67	5.1	<.02
Contour	0.54	0.51	23.9	<.0001
Self	0.32	0.30	0.49	NS
Other	0.075	0.14	93.5	<.0001

Discussion

We have seen that both conflict and crossings episodes affect behavior and eye-movements patterns in a significant way. Interestingly, while both situations require a higher amount of coordination, the responses of the players to crossings and conflicts are different. Players handle crossings by monitoring their partner's piece and avoiding each other whereas conflict is rather solved by exploring the contour for alternative positions. In any case, it shows that eye-movements can provide insights about the type of episodes collaborators are engaged in.

5.5.2 Gaze dispersion and divergence in KAT experiment

We analyze the relationships of two dual-gaze features with the type of interaction episodes in the context of the KAT experiment. The gaze divergence reflects how much they look globally at the same things, and the mutual gaze dispersion reflects whether they look at many objects or whether they stayed focused. Our hypotheses were that divergence could be an indicator of cooperation, i.e. each participant working on a separate part of the problem, versus collaboration, i.e. both participants working together on the same aspect, while dispersion

would be an indicator of the granularity of reasoning with a low dispersion indicating a local, specific reasoning and high dispersion indicating a higher-level, global reasoning on the whole problem.

Method

Gaze divergence is computed as the Jensen-Shannon divergence[8] between gaze density matrices of each subject. These density matrices represent the distribution of fixation times spent in the different parts of the shared workspace. Gaze dispersion is estimated by computing the average saccade length. The mutual gaze dispersion is defined as the sum of the individual dispersions. These features were computed for time chunks of 6 s. so that we can study their evolution in time and how they are related to the verbal interaction.

We analyzed the type of verbal interactions that occurred during episodes in which one of these features was at a high-level. More specifically, we defined a high level episode as a sequence where the overall respective gaze feature level was situated above the 8th decile. We then compared the distribution of utterance types during these episodes with the general distribution of utterance types using a Chi-square test. Utterance types were grouped into thirteen categories. Eight of these categories were related to knowledge exchange and were coded along two dimensions. The first dimension refers to the content of the knowledge exchanged, either task completion oriented (T) or knowledge negotiation oriented (K) and both can be either elaborated (E) or non-elaborated (N), hence resulting in four categories, TE, TN, KE and KN. These four categories are crossed with the second dimension which tells whether the utterance is a demand of information (or information seeking IS) or an information providing (IP). The combination of the first dimension (knowledge content) and the second dimension (transfer direction) lead to our eight categories: TE-IP, TN-IP, KE-IP, KN-IP, TE-IS, TN-IS, KE-IS and KN-IS. In addition of these eight categories, we had five categories to describe non-domain related types of utterances. These were contradiction (CT), i.e. when one collaborators contradicts its peer, information management (IM) which is related to the management of turn-taking and task flow, concept-map management (CM), i.e. when they discuss about the use of the concept-map tool, mutual modeling (MM), i.e. when they speak of their peer's knowledge and finally, strategy (ST) which is about strategy to build the map. This resulted in a total of thirteen categories.

Results

Table 5.6 summarizes the amount and percentage of utterances of each type during high-divergence episodes. The first row (overall) reports the overall distribution of utterances of each type produced by the participants. The second row (hi-DIV) reports the distribution

[8] The Jensen–Shannon divergence is a method of measuring the similarity between two probability distributions. It measures how reliably one can decide if a given response comes from the joint distribution or the product distribution, given that these are the only possibilities.

Table 5.6: Distribution of utterance types during high-divergence episodes

	KE-IP	**KE-IS**	**TE-IP**	**TE-IS**	**KN-IP**	**KN-IS**	**TN-IP**
Overall	324 (15.9%)	72 (3.5%)	131 (6.4%)	23 (1.1%)	189 (9.3%)	39 (1.9%)	310 (15.2%)
Hi-DIV	54 (14.3%)	16 (4.2%)	33 (6.4%)	5 (1.3%)	35 (9.2%)	2 (0.5%)	67 (17.7%)
Residuals	−0.79	0.72	1.76	0.35	−0.01	−1.95	1.25

	TN-IS	**CT**	**IM**	**CM**	**MM**	**ST**	**Total**
Overall	55 (2.7%)	109 (5.3%)	224 (11%)	119 (5.8%)	126 (6.2%)	321 (15.8%)	2042 (100%)
Hi-DIV	8(2.1%)	23 (6.1%)	23 (6.1%)	24 (6.3%)	12 (3.2%)	77 (20.3%)	379 (100%)
Residuals	−0.69	0.62	***-2.88***	0.41	***-2.35***	***2.26***	

of utterances of each type produced during the high divergence episodes. The third line represents the statistical residuals, which are generally considered as significant when they exceed 1.96 in absolute value. The Chi-square test showed that the two distributions differed ($\chi 2 = 29.7$, $p < .01$). A closer look to the residuals shows that three of the categories (gray columns in table 5.6) significantly contributed to establish the difference of distribution. First, learners seem to produce fewer interaction management (IM) utterances during high divergence episodes than during the overall collaboration. Second, they seem to produce fewer mutual modeling cues (MM) as well. Finally, the learners seem to produce significantly more utterances related to the strategy with regards to the construction of the concept map (ST).

Table 5.7: Distribution of utterance types during high-dispersion episodes

	KE-IP	**TE-IP**	**KN-IP**	**TN-IP**	**IS**
Overall	324 (16.8%)	131 (6.8%)	189 (9.8%)	310 (16%)	189 (9.8%)
Hi-DISP	36 (19.6%)	10 (5.4%)	22 (12%)	16 (8.7%)	16 (8.7%)
Residuals	0.93	-0.7	0.95	**-2.49**	-0.47

	CM	**IM**	**MM**	**ST**	**Total**
Overall	119 (6.2%)	224 (11.6%)	126 (6.5%)	321 (16.6%)	2042 (100%)
Hi-DISP	6 (3.3%)	45 (24.5%)	14 (7.6%)	19 (10.3%)	463 (100%)
Residuals	-1.58	**5.13**	0.58	**-2.09**	

We applied the same method to compare the distribution of utterances produced during high-level mutual-dispersion episodes and the overall distribution of utterances. As some of the information seeking categories of the high-level mutual-dispersion episodes presented less than five utterances, we decided to aggregate the four information seeking categories (i.e. KE-IS, TE-IS, KNIS and TN-IS) into a main information seeking category (IS). Hence the following categories were distinguished for this analysis: KE-IP, TE-IP, KN-IP, TN-IP, IS, CM, IM, MM and ST. Table 5.7 summarizes the amount and percentage of utterances for each of these types of utterances. The chi-square test showed that the distribution of utterances during high-level dispersion is different from the overall distribution of utterances ($\chi 2 = 42.2$, $p < .000001$). A closer look at the residuals shows that the absolute value of three of them are higher than 1.96. First, co-learners seem to produce fewer task completion non-elaborated information providing utterances (TN-IP) during high-level dispersion episodes (8.7%) than the overall collaboration (16%). Second, co-learners seem to produce significantly more interaction and task management utterances (IM) during dispersion episodes (24.5%) than in overall (11.6%). Finally, co-learners seem to produce significantly fewer construction strategy utterances (ST) during high-level dispersion episodes (10.3%) than overall (16.6%).

Discussion

Two interesting aspects should be pointed out in these results. First, the amount of *task completion non-elaborated information providing* and *strategy* utterances is much lower during episodes of high-dispersion. It is reasonable as we could expect those kinds of utterances when learners are intensively working on the construction and local planning of the concept-map and thus have low gaze dispersion. Secondly, co-learners seem to be focused on the interaction and task-management aspects during these high-dispersion episodes. Indeed, the results suggest that almost 25% of their utterances during high-dispersion episodes concern interaction and task management. We also know that interaction management utterances reflect mainly metacognitive processes such as monitoring the task. A closer look at the utterances produced during high-level mutual-dispersion episodes suggests that co-learners are mainly involved in metacognitive processes where they seem to "zoom-out" from the negotiation of knowledge and the construction of the concept-map to see the broader picture and figure out the state of the task and what to add next. Hence our exploratory hypothesis seems to be confirmed.

To sum up, results show that to some extent, simple dual gaze features already provide some information about the interaction dynamic by reflecting the type of verbal interaction that occurs. However, it should be pointed out that such socio-cognitive phenomena in this task are very complex and highly conceptual and hence it seems clear that they could certainly not be fully captured only by eye-movements.

5.6 Speech-gaze link

In chapter 2, we reported eye tracking studies that showed the tight coupling between eye movements and the production (Griffin and Bock, 2000) or the comprehension of verbal references (Allopenna et al., 1998; Tanenhaus et al., 1995). Griffin and Bock (2000) showed for instance that the average eye-voice span, i.e. the mean time lag between the fixation on a to-be-referred object and the onset of the verbal production of the reference, is approximately between 800 and 1000 ms. Allopenna et al. (1998) showed that the voice-eye span, i.e. the mean time-lap between hearing the reference and looking at the object of reference, for the listeners is between 500 and 1000 ms. Finally, Richardson and Dale (2005) showed through cross-recurrence analysis that the overlap peak of speaker and listeners' eye movements is at about 2000 ms lag. Hence, on average, the listener seems to look at the referred object two seconds after the speaker.

Building upon these findings, we investigated the eye-voice as well as the voice-eye spans of the eye movement and verbal interactions data collected during two experiments. First, we tried to replicate these results in the specific collaborative situation of the KAT experiment, since this situation contrasts a lot with the strictly controlled experimental settings used in the studies presented above. In our case, the participants were involved in a two-way problem solving dialogue and a far more complex task in terms of social and cognitive processes. Hence, while we expect less systematic results, we believe that replicating these results in more complex and ecologically valid settings may provide insightful results and perspectives, if we keep the richness and complexity of the underlying processes in mind. Second, we also explored the existence of similar phenomena with written communication in the Shoutspace experiment. Indeed, while the processes involved in written communication are very different from verbal communication, it is nonetheless very interesting to see whether we find some similar effects on eye-movements. In particular, we look for a kind of a text-eye span and respectively an eye-text span and compare them with the verbal case.

5.6.1 Explicit referencing in KAT experiment

In the KAT experiment, we are interested at moments when subjects make explicit references to concepts or links on the map. The fact that concepts and links are labeled facilitates the detection of such references in the speech corpus as we could directly match the objects' label with the speech transcript. However, not all utterances matching a concept or link name are necessarily references. It may happen that subjects say such a name while simply discussing about the topic. Hence, we did two similar analyses, one by detecting manually such explicit references in a small subset of our data, so that we could have a set of references for which we have high confidence and one by detecting automatically references on our whole corpus, in order to have a much higher number of cases for analysis.

Method

To estimate the eye-voice span for a specific explicit reference, we need to match the timing of the gaze fixation of the speaker on the referred concept-map object with the timing of the verbal reference in the speech. To do so, we computed the ratio of matching fixations for many possible values of eye-voice span. We define the set V, of size N_V, of all objects, v_i referenced verbally at time t_i and the object fixated by the speaker at time t as $G_S(t)$. Then, for a given eye-voice span δ_S (called speaker's span hereafter), the corresponding ratio of speaker's fixations on referred object $R_S(\delta_S)$ is computed with the following formula:

$$R_S(\delta_S) = \frac{\sum_{v_i \in V} \mathbf{1}_{G_S(t_i - \delta_S) = v_i}}{N_V} \tag{5.1}$$

Where $\mathbf{1}_p$ is indicator function returning 1 when predicate p is true and 0 otherwise. Similarly, we define the object fixated by the listener at time t as $G_L(t)$. And we can define, for a given voice-eye span δ_L, the corresponding ratio of listener's fixations on referred object $R_L(\delta_L)$ (called listener's span hereafter) with formula:

$$R_L(\delta_L) = \frac{\sum_{v_i \in V} \mathbf{1}_{G_L(t_i + \delta_L) = v_i}}{N_V} \tag{5.2}$$

In order to define the set V of all verbal references, we detected all references in the speech to objects of the shared concept-map, as well as their onset time. As explained above, we did this task in two different ways. First, we manually detected explicit references and their precise onset time-stamps across the dialogue of two selected dyads (11.2% of the overall corpus). We detected 49 verbal references to objects of the shared map for the first pair and 107 references for the second pair. Accordingly, the hand-coded analyses reported hereafter are made on a set of 156 verbal references produced by the peers of the two selected dyads. This was done in order to have a set of references for which we are sure that they really are references. Second, we developed and used an automatic transcript-speech alignment software using the Sphinx speech recognition library[9]. This engine was used to segment the speech files into individual words and to get logs of the onsets for each individual word detected. Then, we identified every word that could be a reference by matching words with the content of the concept-map. This technique was applied on the dialogue of 18 pairs for which we had workable data (i.e. sufficient quality speech data and eye-gaze data). With this method, we were able to detect a total of 431 explicit references with a mean of 23.9 explicit references per pair. Of course, this data was less precise and contained some errors compared to the manual method, but it offered a much bigger number of verbal references to analyze.

[9]CMU-Sphinx (http://cmusphinx.sourceforge.net/html/cmusphinx.php) is an open-source general speech recognition engine developed at Carnegie Mellon University.

Results

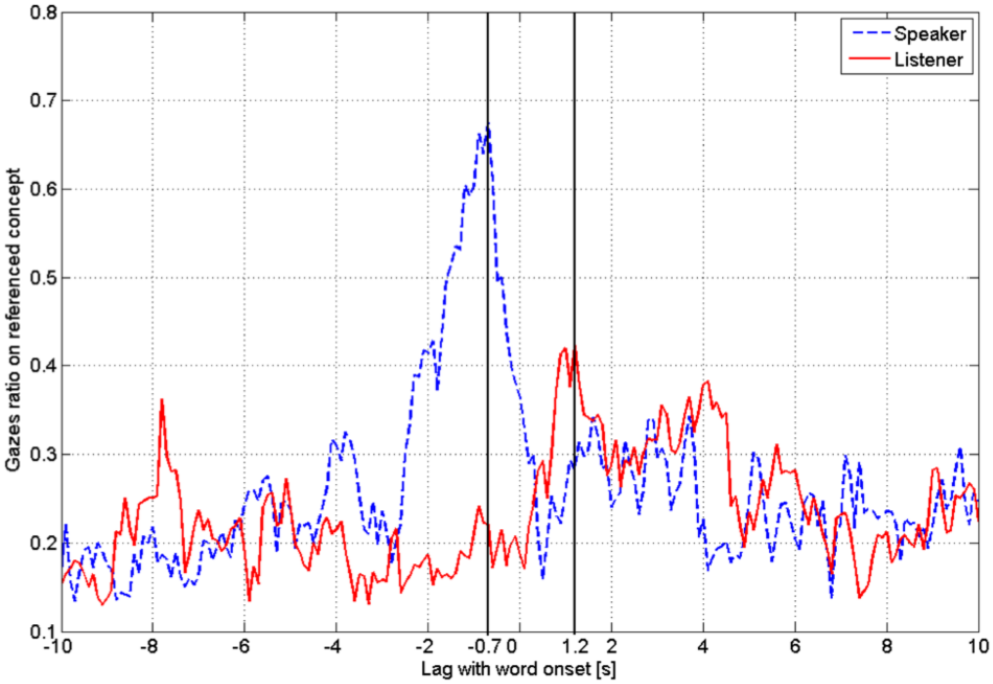


Figure 5.11: Average gaze ratio on referred concept for different time-spans from the verbal reference onset for manually detected verbal references.

We computed the ratio of matching fixations for various spans between -10s and +10s using equations 5.1 and 5.2 in order to find out if there was a peak for certain span values. With regards to the hand-coded verbal references produced by the two selected dyads, we plotted the gaze ratios of the speakers and the listeners for different time-spans in figure 5.11. The origin of the graph has a span of 0, which corresponds to the onset time-stamp of the verbal references. Consequently, the negative values of the X-axis correspond to positive (respectively negative) values of the speaker's (respectively listener's) span; the positive values of the X-axis correspond to the positive (respectively negative) values of the listener's (respectively speaker's) span. The dotted-blue line (ratio of speaker's fixations on reference) peaks at -700 ms with an associated average gaze ratio of 0.675. In other words, there is 67.5% chance of getting a speaker's fixation on the to-be-referred concept 700 ms before she formulates the verbal reference. With regards to the listeners' gaze ratios on reference (red line), the peak is not as well defined as for the speaker. The highest average gaze ratio on the referred concept peaks at 0.425 at 1200 ms. Accordingly, there is 42.5% chance that the listener looks at the referred concept 1200 ms after the speaker started to pronounce its name. In both cases, the peaks are not very sharp which indicates that these spans are variable.

We replicated the same procedure for the automatically detected verbal references and the results are shown on figure 5.12. The corresponding optimal speaker-span is 900 ms with a corresponding gaze ratio of 0.49. With regards to the listener, the peak is even more diffuse

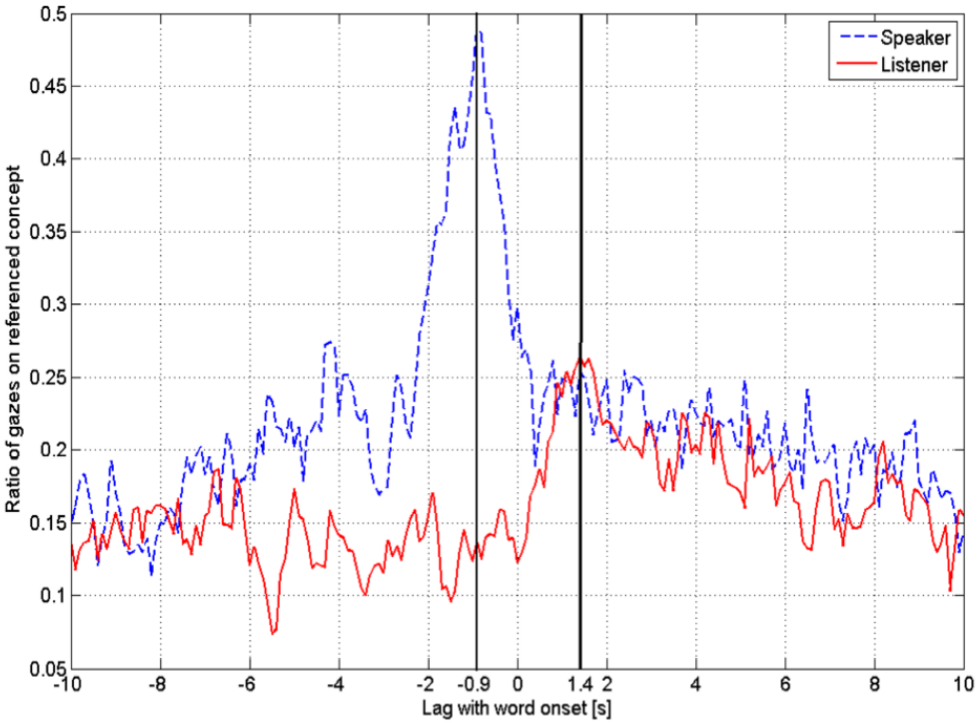


Figure 5.12: Average gaze ratio on referred concept for different time-spans from the verbal reference onset for automatically detected verbal references.

and reaches 0.27 of average gaze ratio for a span of 1400 ms. Furthermore, we computed the probability of getting at least one fixation on a referred object during a four second and a two second time-window before (or after in the case of the listener) the onset of the verbal reference (namely between -4 s and 0 s and - 2 s and 0 s). There is a probability of 0.74 that the speaker fixates the referred-object within four seconds before producing the verbal reference and a probability of 0.69 within two seconds. For the listener, there is a probability of 0.66 that the listener fixates on the referred-object four seconds after the verbal reference and a probability of 0.55 that the listener looked at the object within two seconds after the reference.

Finally, we also used this data to compute the cross-recurrence only during the moments of references (see figure 5.13). Moreover, we rearranged the data so that it was always the speaker of the reference that was the first subject. Hence, the lags correspond to a speaker-listener lags. We found a peak of recurrence for a lag of around 2 s which means that the listener of the references look the most at the same place than the speaker two seconds after him.

Discussion

Overall, both the manually detected and automatically detected datasets provide results that are fairly supportive of the robustness of the speaker's span (eye-voice span) and listener's span (voice-eye span) mechanisms, even in realistic and complex collaborative situations.

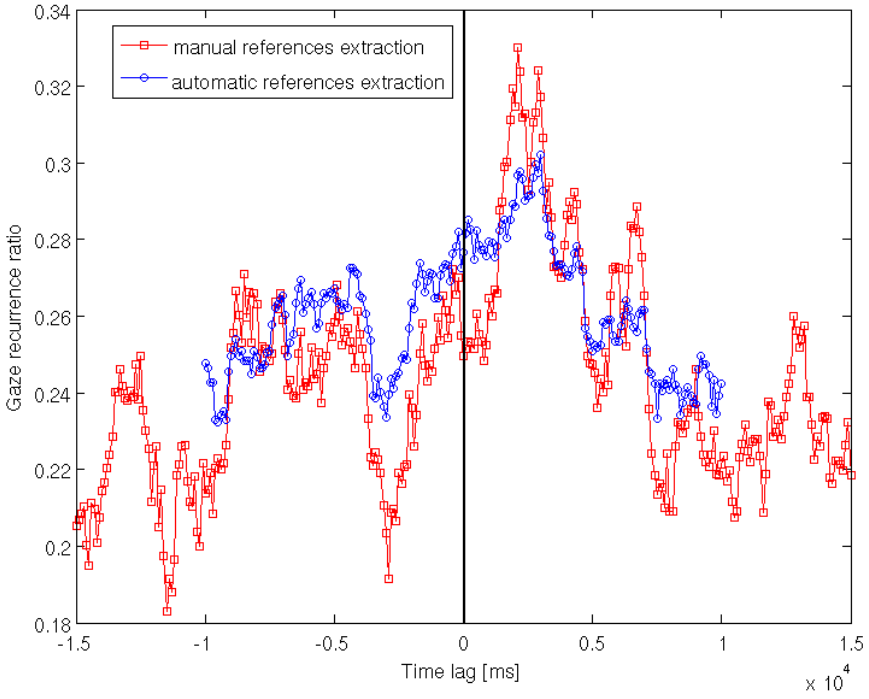


Figure 5.13: Cross recurrence during references on concepts for both the manually detected references and the automatically detected ones. This has been computed so that the speaker is always the first subject.

Indeed, we found that the ratio of speaker's fixations on referred object peaked at 700 ms (900 ms with the automatically coded data) before the onset of the reference. This is relatively consistent with the findings of Griffin and Bock (2000) who found an eye-voice-span between 800 ms and 1000 ms. Similarly, for the listener, we found that the ratio of fixations on the referred object peaked at 1200 ms (1400 ms for the automatically coded data) after the onset of the reference. Again, this is relatively consistent with the estimation of Allopenna et al. (1998) who found a voice-eye span between 500 ms and 1000 ms. The longer span that we found could be explained by the fact that we have a complex visual stimulus for which it can take more time to localize the referred object, while Allopenna et al. (1998) used a very simple stimulus with only few objects to be looked at. The more diffuse nature of the listener's peak can be explained by the fact that some references may be easier to localize than others. This could make gazes on reference to happen for different span values. Another complementary explanation is that some references may be longer to produce verbally leading to a longer delay after the utterance onset.

The ratios of fixations on references for the automatically coded data are much lower than for the manually coded references. This is mainly explained by the fact that the automatically coded data contains a number of false references and possibly also words which have been badly aligned with the audio file because of errors from the speech recognition engine. This

could be certainly improved by further developing the alignment module and having better rules for the detection of references in speech. In any case, this opens promising perspectives for potential applications that would use the gaze and the speech in real-time to automatically detect explicit references. We present a prototype of such an application, called REGARD, in the following section.

Finally, it is very interesting to see that by computing cross-recurrence only during these moments of references, we have been able to replicate closely the results of Richardson and Dale (2005). Indeed, we also found a maximum recurrence value for a speaker-listener lag of 2 s. This is of a great importance as we replicated this result in a real complex collaborative situation.

Application: REGARD model

From these results, we developed an algorithm, called REGARD, that aims to identify automatically explicit references present in the dialogue of two collaborators working on a shared workspace. The principle of this algorithm relies on the existence of the eye-voice and voice-eye spans. The idea is that, if a word, which is an explicit reference to a visual object, is pronounced several times, the corresponding speaker and listener's gazes before and after these utterances should be essentially distributed on the referred object. Hence, by monitoring the gaze data of the speaker (and of the listener) before (respectively after) each pronounced word and by computing how well they match over multiple pronunciations of the same word, we should be able to decide whether the word is a reference or not and if it is, to which object it is directed.

In practice, this consists in creating for each pronounced word, a gaze density vector over all the possible objects and in aggregating at each pronunciation of a given word the speaker's and listener's gazes inside the corresponding vector. More specifically, we aggregate the fixations with weights which depend on the relative time of the fixation compared to the word onset. The rationale is that the closer is a speaker's (or listener's) fixation from the optimal eye-voice (respectively voice-eye) span, the more chances it has to be on the referenced object. Hence, we give more weight to fixations which are close to the optimal span and less weight to those that are further. More precisely, we use a Gaussian function centered on the optimal spans to compute the weight of a fixation (see figure 5.14). After each aggregation of gazes in the gaze density vector associated to a given word, the algorithm performs some computations with the vector values to check whether it seems to be an explicit reference. More specifically, it first compares the total amount of gaze data that have been aggregated in the vector to a *minimum gaze threshold* to test whether these accumulated data may be considered as meaningful and thus if a decision can be taken for this word. Secondly, if enough data are present, it tests whether one cell of the vector contains a high amount of data compared to all other cells by comparing the sum of the maximum cell, i.e. the cell having the highest value, and the difference between the two maximum cells to a *matching threshold.* Indeed, if the word was a reference to an object, then most the accumulated gazes should be in the cell of

the corresponding object and all other cells should have low values. Hence, if one cell contains a high amount of gazes relatively to all other cells then the algorithm marks this word as being a reference to the corresponding object. Otherwise, it tags the word as not being a reference to an object.

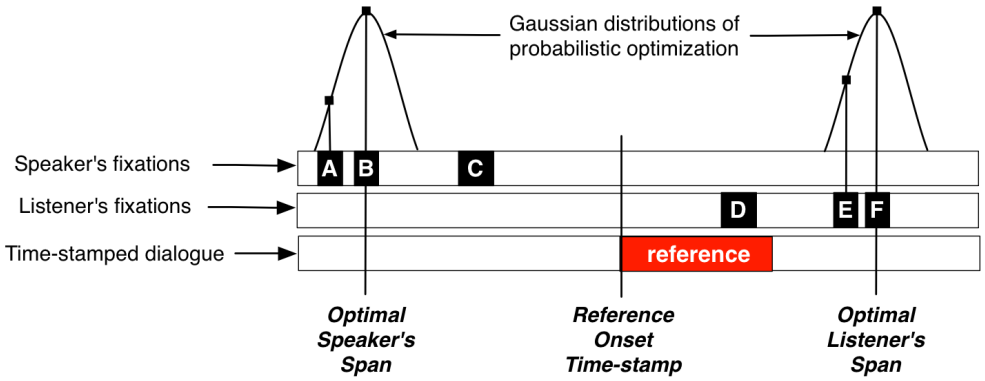


Figure 5.14: Gaussian weighting functions used to aggregate the speaker and listener's gazes at each word utterance. Fixations B and F fully contribute to the model, whereas fixations A and E partially contribute to the model. Fixations D and C do not contribute to the model.

The algorithm was tested and optimized using the data of the KAT experiment. Indeed, as explained in the previous section, the speech was transcribed and the timestamps of each word onset was determined by using a speech recognition engine. Hence, we had a list of words with their precise time onsets and their corresponding concept on the map if the word is a reference. This datasets consisted in 1038 words among which 301 were not explicit references, we called them common-words, and 737 were explicit references of a concept, hence called concept-words. We could thus run the algorithm on these 1038 words and tested whether the results of the algorithms fit reality.

The first step was to optimize the various parameters of the algorithm. First the mean and the standard deviation of the Gaussian's used as weighting functions. Second, the *minimum gaze threshold* used to assess whether a gaze density vector contains enough accumulated gazes. Finally, the *matching threshold* to assess whether the relative amount of gazes in a cell is sufficient to consider the word as an explicit reference.

The Gaussian parameters were optimized by using only the 301 concept-words. The procedure consisted in applying this algorithm, without the two decision steps, on these words and in computing for how many of these words, the highest cell in the gaze density vector corresponded to the associated concept. This was done for several values of mean and standard deviation so that we found the optimal values for both of these parameters (see figure 5.15). The peak value suggests that the optimal values for the speaker's span mean is 800 ms and the optimal speaker's span standard deviation, the optimal value is 100 ms. A similar procedure was undertaken to detect the optimal listener's span mean and standard deviation which resulted in values of 1600 ms for the mean and 350 ms for the standard deviation.

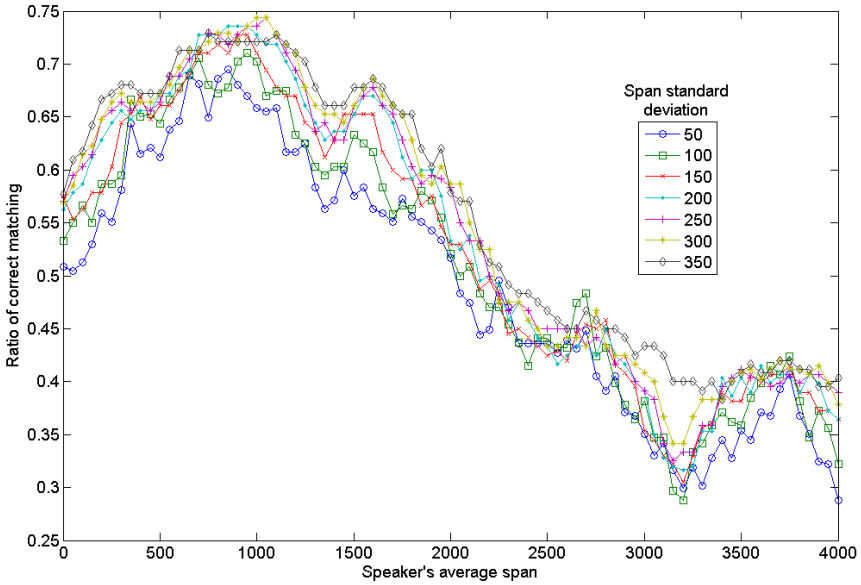
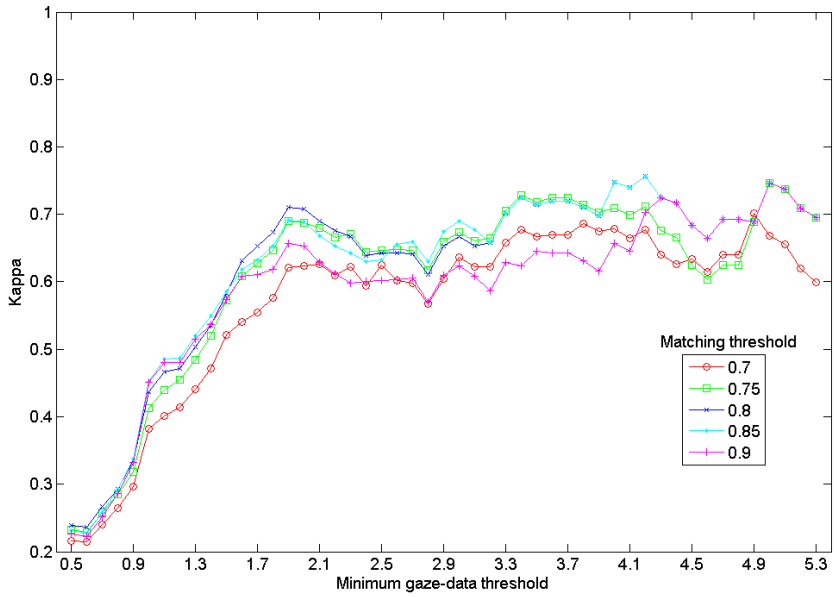


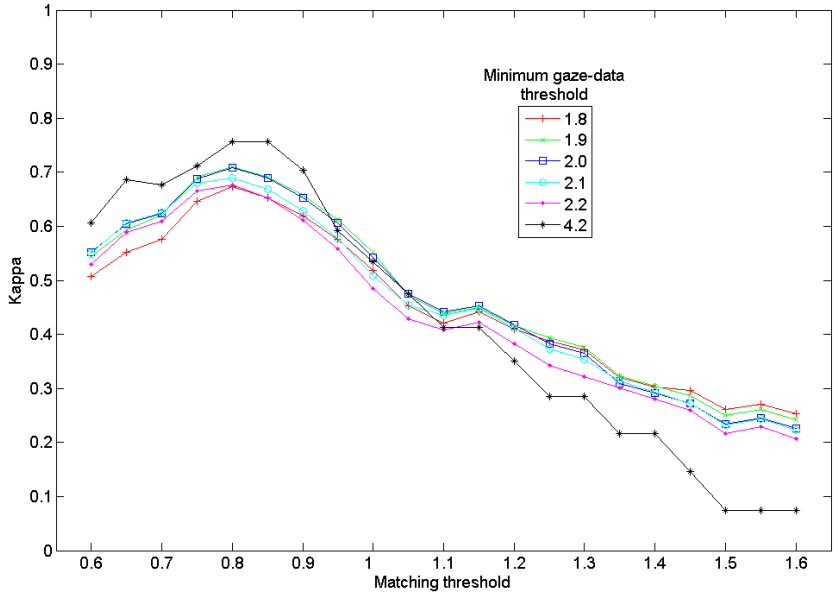
Figure 5.15: The speakers' average span and plotted against the model's detection score. The various curves represent different values for the standard deviation.

For the optimization of the two thresholds used for the decision, namely the *minimum gaze threshold* and the *matching threshold*, the whole dataset was used and the kappa statistics measuring the agreement between the algorithm output and the reality was used as a measure of fitness. This allowed us to find the *minimum gaze threshold* above which the kappa score do not increase anymore (see figure 5.16a) which corresponds to a value of 2. We also extracted the optimal *matching threshold* that maximizes the kappa score (see figure 5.16b) which is 0.8.

We ran the final optimized algorithms on the whole dataset. Among the 1038 words composing the dataset, 182 passed the *minimum gaze threshold* allowing the algorithm to classify them as common or concept words. Table 5.8 shows the confusion matrix between the actual classification and the output of the algorithm for these 182 words. It indicates first whether the words were correctly classified as common or concept word and secondly if the identified concept-words were correctly associated to their concept on the map. The algorithm makes several types of errors which have different severity. First, it can classify concept words as common words (yellow cell), i.e. fail to detect that a word is a reference to an object, which is not a very serious type of error. Secondly, it can classify correctly a concept-word but associate it to the wrong object (orange cell) which is more serious type of error. Finally, it can incorrectly classify a common-word as being a concept word (red cell) which is also a serious type of errors. These two latter types of errors are the most serious as they correspond to incorrect references detected. However, they are quite few with only 13 cases over 182 words which represent 7% of the words. Overall, the performance of the algorithm is quite good with a kappa score of 0.71. We should also note that some errors are due to the fact that the word



(a) *minimum gazes threshold*



(b) *matching threshold*

Figure 5.16: REGARD classification performance plotted against matching the decision thresholds. This data was used to find the optimal *matching threshold* and the minimal *minimum gazes threshold* above which decisions are not better.

onset timestamps, found by aligning automatically the speech transcript with the audio file, were not always correct and also that the concept-word are not always correct as they were also detected automatically by comparing words and concept's names on the map

Table 5.8: REGARD model's classification and matching performance details for the 182 words that passed the *minimum gaze threshold*. Incorrect cases (non-green cells) have been colored according to the severity of the error

	Classified as Common-words	Classified as Concept-words			**Totals**
		Correctly matched	Incorrectly matched		
Actual Common-words	112	9			**121**
Actual Concept-words	14	43	4		**61**
Totals	**126**	**56**			**182**

This algorithm is a promising step towards a dual-gaze aware application. Such an application could be used, for example, to annotate automatically images from the dialogue and gazes of two persons discussing about an image. It could also serve as a grounding detector, by identifying which terms are used as explicit references in a coherent way by both the speaker and the listener.

In practice, there are two points that still need some works in order to make this algorithm really generalizable. First, the version presented here is aware of which word is said while in a real situation, only the audio stream would be available. Thus, it would require to be coupled with a speech recognition engine to be usable. An alternative would be to work at the sound signal level without recognizing explicitly the words. Instead, the algorithm could simply identify when a certain sound is repeated in an audio stream and then consider this sound as a word. It would have the advantage of being much more general as it would also accept non-word or pseudo-word but it is not sure that it is technically less complex to accomplish than speech recognition. Finally, the second aspect that makes it not usable in general situation is that the algorithm has to be aware of where the potential referenced objects are. Here a potential solution would be to work at the area level. Instead of having a gaze density vector representing all the possible objects. We would have a gaze density matrix representing the screen or, more generally, the shared workspace and instead of detecting a peak in the vector, we would have to detect a high-density cluster in the matrix.

5.6.2 Explicit referencing in Shoutspace

In the Shoutspace experiment, we did similar analyses for references in textual messages (Cherubini et al., 2008). The general idea remains the same, we analyze the gazes on referred objects around the time of the message writing for both the reader and the writer. We asked whether gazes of writers, before or during the message writing, were localized on the referred objects and similarly, whether gaze of readers, during or after reading the message, were also localized around the referred place.

Method

We parsed all chat messages in order to extract the ones that contained explicit references to objects of the shared workspace. More specifically, we identified both direct references, i.e. when one refers to a specific object or zone in the displayed map using a single word or expression (e.g. "the parking 250" or "the upper-left corner") and relative references (e.g., "The green space on the right of parking 250"). In this latter situation, we identified separately the referent ("The green space"), the relation ("on the right of") and the relatum ("parking 250"). These detections were accomplished in three parts. First, we removed typos using a collection of common typing errors in French. Then we looked for the relation extracting prepositional phrases (e.g., "on the right hand-side of", "below the", etc.). Finally, using a lexicon containing references to movable objects (e.g., 'the stage', 'the concert') and to fixed landmarks (e.g., "P200", "chemistry building"), we extracted all references. This lexicon also contained correspondences between references and polygons on the map. This allowed us to map each reference word to its corresponding polygon on the workspace and thus each message was associated to one (for direct references) or several (for relative references) *polygons of interest*.

The second step consisted in aggregating gaze data in order to infer zones attracting users' attention over a certain time period (e.g. before, during or after the writing or reading of a message). Our approach was to not go through the usual fixation identification step because we were only interested in general zones of interest over quite long period of time. Thus, we developed a gaze clustering method based on raw gaze data density disregarding the temporal order of the data. First, the raw data were accumulated on a two-dimensional grid representing the screen in order to obtain a spatial gaze density matrix. Then, this grid was smoothed with a Gaussian filter to eliminate discontinuities. Finally, a contour function was applied to this resulting density grid. This resulted in a list of isodensity lines (see figure 5.17). We took the center of the highest isolines as the *gaze density peak*. More specifically, we could identify several possible peaks by picking the highest isoline first and then by removing all isolines which include the picked one. This process is repeated until there were no more isoline and this resulted in a list of peaks ordered by their height. This clustering technique was applied for the reader's gazes and writer's gazes of each message. More specifically, we aggregated the writer's gazes during a time window beginning 5000 ms before the start of editing and 1500 ms after the end of editing. Similarly, we aggregated the reader's gazes during a time-window of

the same length and starting 1500 ms before the end of editing, or before the first reader's gaze on the message if it happened after the end of editing.

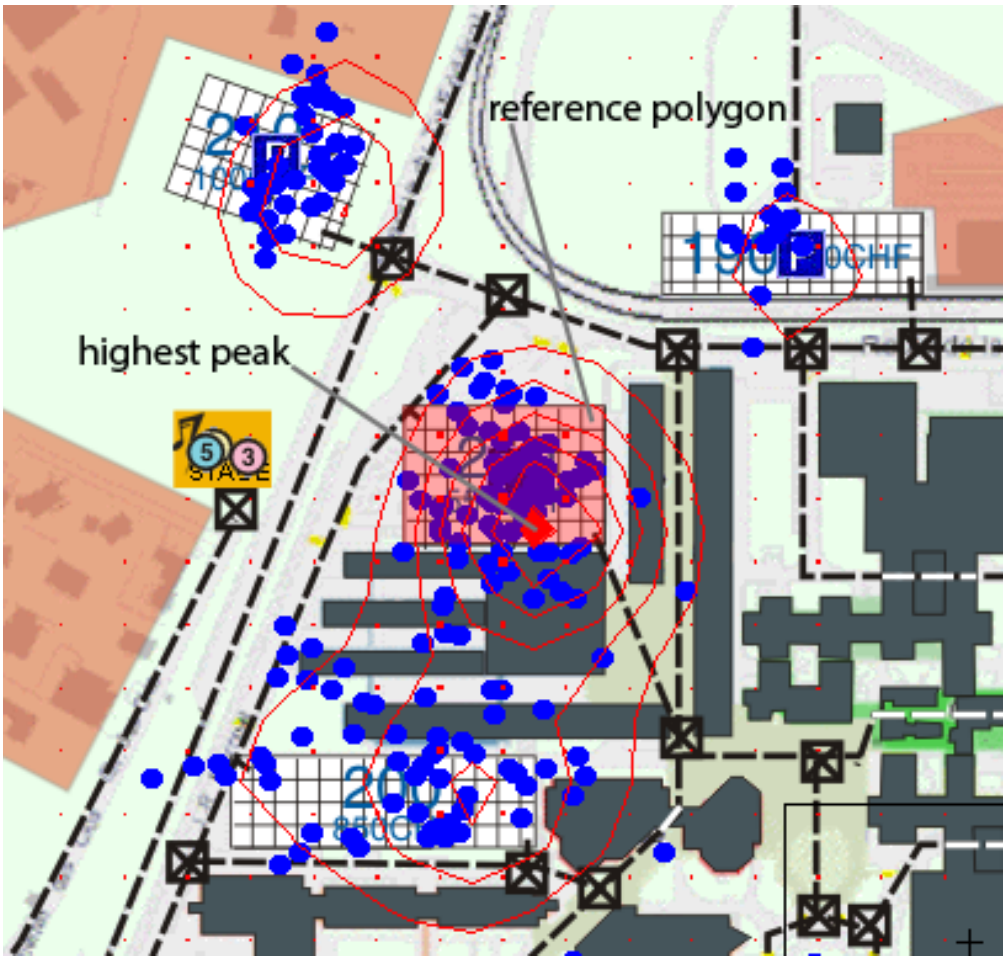


Figure 5.17: Example of contour plot used to calculate the gaze clusters. The contour with the isoline corresponding to the highest density was associated to the interest polygon referenced in the message. Blue dots represent eye-tracker raw data

The last step was to match the identified *gaze density peaks* with the *polygons of interest* of a message. We simply took the N highest *gaze density peaks*, where N is the number of *polygons of interest*. Then, we computed the distances between each selected peak and each polygon. Finally, we associated every peak with the polygon at the smallest distance with the constraint that each peak must be associated to one, and only one, polygon. We also computed a score of overlap between the writer's gazes density peaks and the reader's ones. The only linguistic information used for this overlap score computation was the number of interest zones (N). The N highest peaks of the writer were simply associated with the N highest reader peaks using the same methods than for associating peaks to polygons. Then, the mean distance between the emitter's *gaze density peak* and the reader's *gaze density peak* was taken as the overlap score for that message.

Results

First, we tested whether emitters of a message actually looked the most at the polygons they referred to. We analyzed the proportion of cases for which the *gaze density peaks* associated to the polygons of interest were also the highest ones. For multiple interest polygons we matched the highest polygons with the first extracted references. Contour-clustering of messages leading to a single *gaze density peak* were excluded from this analysis as they could not be incorrect. A proportion test revealed that our assumption was correct 71% of the time ($F[1,198] = 66.27$, $p < .001$), suggesting that the zones named in the messages were also those attracting the highest gazes density of the writer.

Then, we conducted an analysis of the distance of the emitter/receiver's *gaze density peak* to the *interest polygon* over time. More specifically, we moved the time-window defined above, between -2 minutes and +2 minutes from the middle editing time. Figure 5.18 shows the average distance between the *gaze density peak* and the interest polygon at different time lags for the writer and the reader of a message.

The differences between the period of time when the message was edited or read and moments further or back in time were supported by a 2 (phase: writing–reading) × 99 (lag times) mixed-effects analysis of variance (ANOVA) (lag and phase as a repeated measure factors). Results revealed a significant difference between the extremes of the curves of figure 5.18 and their central depression, i.e. a main effect of lag, $F[98,50060] = 23.78$, $p < .001$. The interaction of lag and phase was not significant, $F[98,50060] = 0.37$, $p = .54$, $n.s.$. But, the effect of reading or writing phase revealed to be significant, $F[1,50060] = 23.78$, $p < .001$. We measured a mean distance between gazes peaks of polygons of interest in the reading phase of 256 pixels vs. a mean distance in the writing phase of 241 pixels for the period before -24 seconds and after +24 seconds. On the contrary, the minimum distance in the reading phase was 165 pixels vs. 89 pixels in the writing phase. This implies that emitters and recipients of a message containing spatial references to the shared map were looking at the interest polygon, during writing or reading, at above chance level.

Finally, we analyzed the reader-writer overlap score, i.e. the distance between the *gaze density peaks* calculated for the writer and those of the reader. More specifically, we compared this distance for messages that were followed by a repair act, i.e. by some utterances used to correct a possible misunderstanding, and those not followed by such a repair act. We performed a Kruskall-Wallis test which accounted for the difference of the distance between the emitter's *gaze density peaks* and those of the recipient of a message and the presence of a repair act in the following messages. Messages not followed by a repair act presented an average distance of 85.65 pixels, *vs.* messages followed by a repair act with an average distance of 231.37 pixels ($\chi^2[1,307] = 206.03$, $p < .001$).

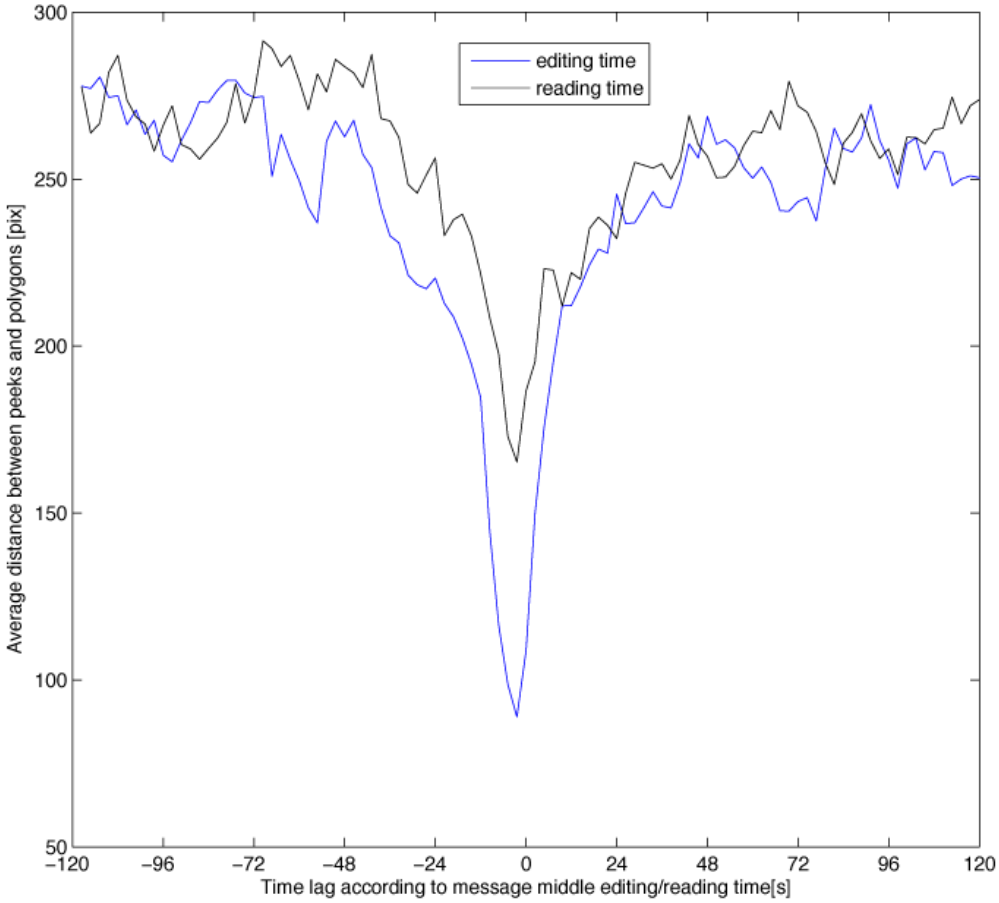


Figure 5.18: Relation between time and distance peek-polygon for the reader and the writer

Discussion

These results bring interesting perspectives by showing that we found similar effects during written communication and verbal communication. However, it is difficult to define the existence of precise eye-text and text-eye spans comparable to the eye-voice span found in verbal communication. It is nevertheless interesting to see that we have the same tendency to monitor the content of our written references and that this monitoring starts before the writing of the message. Finally, the fact that the distance between the attentional zone of the writer and the reader is related to the presence of a misunderstanding reminds the results from Richardson and Dale (2005) who found a relationship between the cross-recurrence and the level of understanding. This opens also perspectives for the design of gaze-aware applications that could possibly detect potential misunderstanding.

5.7 Discussion

This chapter presents several results relating dual eye-movements patterns with collaborative activity in various types of tasks. The tasks that were analyzed offer a variety of cognitive activities performed in collaboration, ranging from highly conceptual tasks to low-level sensory-motor coordination. Through these different experiments, we have shown that dual gaze patterns may inform us about collaborative processes at different levels of granularity. In section 5.3 and 5.4, we have seen that macro-level aspects of collaboration, namely collaboration product and expertise of collaborators, are reflected in gaze patterns. Although expertise is not a product of collaboration, it plays a very important role for the course of collaborative activities. Indeed, depending on the relative expertise of collaborators, we expect different patterns of interaction.

Particularly interesting results come from the automatic prediction of success in RavenBongard and more specifically, of expertise in Tetris and KAT. Hence, the ability of automatically detecting expertise level of collaborators is a very interesting feature for the development of tools aimed at enhancing computer-supported collaboration such as awareness tools or group mirrors. Such tools can provide differential support depending on the group composition of the collaborators. Moreover, we saw that in the case of Tetris, we have been able to detect the group composition from the data of only one subject, which is interesting as it suggests that we could monitor a single collaborator and make inferences about the relative expertise of the dyad. The main limitation of these results is that they are based mostly on task-dependent features and it is thus difficult to generalize these findings to collaboration in general. Finally, let's also note that generally we reach much better classification results by combining gaze information with other modalities such as actions and speech.

On a micro-level, we have seen in section 5.5.1 how gazes are affected by the current type of interaction. Indeed, in specific situations requiring more coordination, gazes patterns were slightly different than during other situations. In KAT (section 5.5.2), we have shown that episodes defined from particular gazes patterns correspond to specific types of verbal interactions and possibly of cognitive activity. These results are only statistical differences and do not allow predictions but they still shed light on the relationship between interaction and eye-movements.

The strongest findings concern the precise relationships between speech and gaze. Indeed, we have been able to replicate very closely results about the eye-voice span and voice-eye spans mechanisms found in very controlled settings (Griffin and Bock, 2000; Griffin, 2001; Allopenna et al., 1998; Tanenhaus et al., 1995) in our complex collaborative situations (see section 5.6.1). This shows that this eye-voice relationship is a reliable and robust phenomenon that also happens in real collaboration. This opens interesting perspectives for gaze-aware applications, such as REGARD, which could take advantage of this close interrelation to make inferences on how the visual content is referenced. Finally, in section 5.6.2, we have found similar effects with written communication and we have shown that there exists a monitoring

of the references both by the writer and the reader of messages and moreover, that a bad monitoring may lead to misunderstanding. The close interplay between speech and gaze is certainly at the origin of the gaze coupling between speaker and listener demonstrated by Richardson and Dale (2005) through cross-recurrence analysis. Again our results support this idea by showing in KAT the existence of a similar coupling peak (see section 5.6.1) for a lag of 2 seconds. Also, in section 5.3.1, we have found that the height of the peak is correlated with the overall performance at the task which is complimentary with results of section 5.6.2 which show that, at micro-level, less good gaze overlap lead to more misunderstanding. Thus, cross-recurrence is a very promising candidate for the analysis of collaboration through eye-movements. However, as explained in section 4.3, it is probably necessary to investigate in more details the results of such analyses in order to obtain meaningful results out of it.

Throughout these results, we validated the dual eye-tracking method by showing that it could bring insight on the cognitive processes underlying collaboration. Indeed, we have been able to get meaningful results for many different types of tasks at different level of granularity. We also faced the limitations of these analyses. It appeared that in most cases, meaningful gaze indicators are specific to the concerned task, which makes difficult to generalize these results to collaboration in general. The exception is cross-recurrence which is a generic dual gaze feature that appears to be well related to collaboration. More generally, the key result is the important effect of explicit references on eye-movements, as they act as a link between the gazes of the two collaborators.

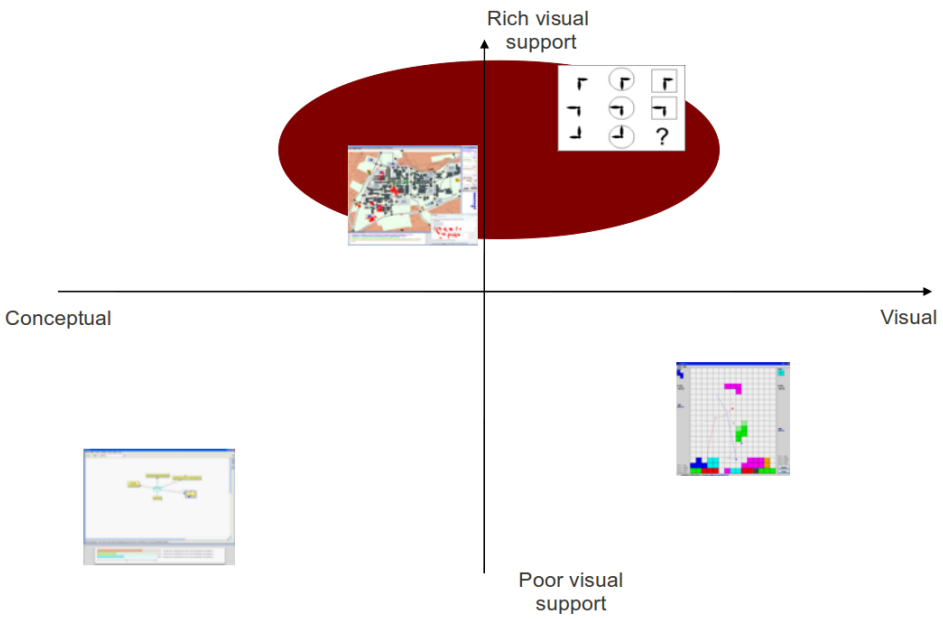


Figure 5.19: Taxonomy of the tasks studied according to two dimensions: type of cognitive activity and type of visual content. The zone of interest for dual eye-tracking studies is shown in red.

These experiments also highlighted the necessity of a careful choice of the task. First, the type of visual stimulus is important to perform meaningful analyses. If we consider, for example, the KAT experiment, the stimulus is empty at the beginning and is progressively built while the collaborators perform the task. This imposes constraints on the analyses that could be performed and requires great care as we have a more and more complex stimulus. On the opposite, in a task like RavenBongard, the whole stimulus is present from the beginning, which makes analyses much easier. Second, the type of task is also important. Indeed, very conceptual task, like the KAT experiment, have much less chance to affect eye-movements than more perceptual, highly visual task, such as Tetris. But on the other hand, non-conceptual tasks have less chance to inform us about the cognitive activities of the subjects. From these considerations, we have built a tentative taxonomy of the tasks (see figure 5.19) in which we represented our different tasks and we identified a zone of interest for dual eye-tracking studies. We used this result to choose the task of our main experiment: pair-programming.

6 Pair programming and dual eye-tracking

6.1 Introduction

This chapter presents the GREPP (Gaze-aware Remote Pair-Programming) experiment which is the main experiment of this thesis. This last study benefits from all the various methodological developments presented so far as well as of the experience gained through the exploratory studies presented in the previous chapter. Pair-programming has been chosen in accordance with the previous findings, i.e. it is both conceptual and has a rich visual aspect present from the beginning, and ecological validity. While pair-programming is often accomplished in co-located settings, technical constraints (the fact that each eye-tracker has its own screen) forced us to study it in a remote environment.

We first briefly present pair-programming and then, we discuss the motivations for choosing this task as well as the main goal of this study.

6.1.1 Pair-programming

Pair-programming consists in having two programmers working together in front of a same screen to perform programming activities. Usually (Williams, 2001), there is one person, called the driver, which controls the keyboard and who is in charge of the low-level activity, i.e. typing the code. The second collaborator, called the navigator, has a higher-level role which consists in monitoring and planning the program structure from a general point of view. While drivers are focused on the specific aspect of the program they are currently writing, navigators keep a global view to see how specific parts get included in the overall program structure and to think of strategic directions for the next part. Generally, the two programmers frequently exchange their role.

While it is clear that pair-programming leads to better structured and less error-prone programs (Canfora et al., 2007), it is difficult to assess whether the overall productivity of programmers is better or not as it requires two programmers working on a single development.

Nevertheless, several companies nowadays impose this methodology to their employees. Hence, it is a realistic collaborative activity.

6.1.2 Motivation

We have chosen to study programming because it is both a conceptual and a visual task. Indeed, programming requires building up a mental representation of a real-world problem in computational terms. In other words, one has to make an analogy between an already existing representation of some real-world aspects with a representation of the same thing in computer language terms. This involves abstractions to translate one representation into the other. Indeed, to write a program which implements a specific problem, such as a board game, we need to translate the objects of the real-world into datastructures and to abstract the essence of the game. On a more practical side, programming is a visual activity that requires reading or writing the code that translates the mental program model.

In this study, we decided to focus specifically on code reading and understanding. First, it is much more adapted to eye-tracking studies because programmers do not need to type and look at the keyboard while performing the task, which would produce big holes in gaze data, thus making reliable analyses much more difficult. Second, it also offers greater opportunity of analysis because we deal only valid programs and we can benefit from automatic code analysis offered by compilation tools to analyze the precise relationship between gazes and code content. Finally, code understanding is a very conceptual aspect of programming while keeping a fully visual aspect opposed to code design or writing which can have a non-visual component.

The main objective of this study is the identification of eye-movement patterns caused by collaboration. More specifically, we aimed at comparing gazes when reading code alone and gazes when reading code in pair. Hence, this experiment has been conducted in two parts. First, we recruited individuals to perform the tasks alone and then, we did a second experiment with the same tasks but with pairs. The rationale is to compare eye-movements of individuals with eye-movements of collaborators on the same task in order to see how they change. More specifically, we first want to identify eye-movement patterns of individual programmers during code reading and understanding tasks. Second, we want to see how these patterns are affected by collaboration. Finally, as in the previous experiments, we want to relate collaboration characteristics with dual-gaze features.

6.2 Questions

First, we wanted to characterize **eye-movement patterns during program reading** in general. More specifically, we were interested in how the general gaze indicators of code reading vary with the type of activity. We looked mainly at the data from the solo experiment to answer this question.

Second, we were interested in how the expertise level affects these characteristic patterns. More specifically, we looked for **specificities of expert** programmers in the eye-movements pattern during code reading. At the same time, we were also interested in **how the collaboration modifies the gaze patterns** of programmers and also, whether it was related to the expertise level. To answer these questions, we compared several indicators along two dimensions: the expertise and the condition, solo or duo. Indeed, we did not want to compare the condition alone as the numbers of experts was not balanced between the solo and the duo condition.

Finally, we also analyzed dual-gaze indicators, namely divergence and cross-recurrence, and we looked at how they are related with the collaboration process. On the one hand, we used gaze divergence to **identify episodes of coordination** to see what type of reading is performed during these phases and on the other hand, we **related the collaboration quality and performance with dual gaze indicators**.

6.3 Method

We describe the general method used in both solo and duo experiments. In a general manner, both experiments were almost identical both in terms of task and of methods. Globally, the solo experiment was simpler to realize as there were no constraints of dual eye-tracking setup and of using shared application. For this reason, we describe the method of the duo experiment, the solo experiment being the same without the shared aspects.

6.3.1 Task

In the experiment, individual subjects or pairs of subjects had to solve several program understanding questions of different types about a program implementing a simple two players arithmetic game using a simple text interface. This game consists in drawing turn-by-turn numbers from a list of numbers between 1 and 9, until one of the players obtains three numbers that add up to 15. If no player obtains such a combination before the drawing list is empty, then the game is over. Subjects were not informed about the rules of the game and had to infer them from the code.

Subjects had to read two different implementations of this game with the same general code organization which is shown on figure 6.1. The parts that differed (in red on the figure) were the way the game state (implemented as class fields) was represented and how the game rules (*rule* methods) were implemented. The similar aspects were the *game flow* functions and the user interface (*UI*) functions. The main *flow* function (method *run()*), which was similar across the two implementations, consisted of a loop which was repeated until there was a winner. The loop body consisted of first asking the current player to choose a number, then checking if this was a winning move, then changing the current player and finally, displaying the current game state to the players before starting again for the next player. The *UI* functions were basic

functions to display either the current game state or the game end (winner/looser or game draw) or to ask for a number to choose. Concerning the aspects that differed between the two versions, in the first version, the game state was represented by three lists of numbers (two for the numbers drawn by each player and one for the remaining numbers) and the winner checking function was testing all possible triplets in each player's list until it found one triplet that summed to 15. In the second version, the game state was represented by an "ownership" array where the indexes corresponded to the nine numbers and the value corresponded to who drew the number with a special value to indicate that the number was remaining. The winner checking was done by constructing all possible winning triplets and by checking for each if it belonged to one of the player. Finally, the move validity checking, i.e. testing whether the drawn number was still available, was accomplished in the simplest way in accordance with the game state representation. The complete listings of the two versions of the program can found in Appendix.

There were two main phases in the experiment. In the first phase, participants had to first read and understand the first implementation of the game. They had to identify and describe the rules of the game, with the sole indication that it was a two player turn-based arithmetic game. They had 10 minutes to perform this task. This was followed by three short questions on the same code. These short questions consisted either in finding specific errors in the game implementation and proposing a possible fix, or in describing a specific aspect of the implementation. These questions lasted two minutes. In the second phase, they first had to read the second implementation of the game. They had to identify the differences with the first version and to explain them. They had 5 minutes to perform this task. Finally, this question was followed by five short questions similar to the first phase (finding a bug or describing a specific aspect). Subjects answered to each question orally after having pressed a button to indicate the beginning of their answer.

These tasks were designed to reflect different aspects of program understanding. We consider that during programming activity, people have two linked representations, or models, in mind: a world-model corresponding to a usual representation of a real-world problem (in our case, a game) and a program-model corresponding to the formal, computational version of the same problem (von Mayrhauser and Vans, 1995). The cognitive activity of programming consists in building the program-model so that it reflects the world-model at best. The first question of the first phase consists in reading a program to identify how it works and how it implements a specific real-world problem (in our case an arithmetic game). Hence, subjects had to build a mental representation of the program and from it, a representation of the associated real-world problem. In the first question of the second phase, subjects already had a mental representation of the real-world problem that is implemented, as they know that it is the same game. Thus, they have to modify, or update, their program-model associated to this problem in accordance with the new code. Finally, for short questions in both phases, they already had existing program-model and world-model's. Hence, their activity consisted either in refining their program model in order to describe a specific aspect of it, or in checking how well it matched with the actual game it implements, to find possible bugs and fixes.

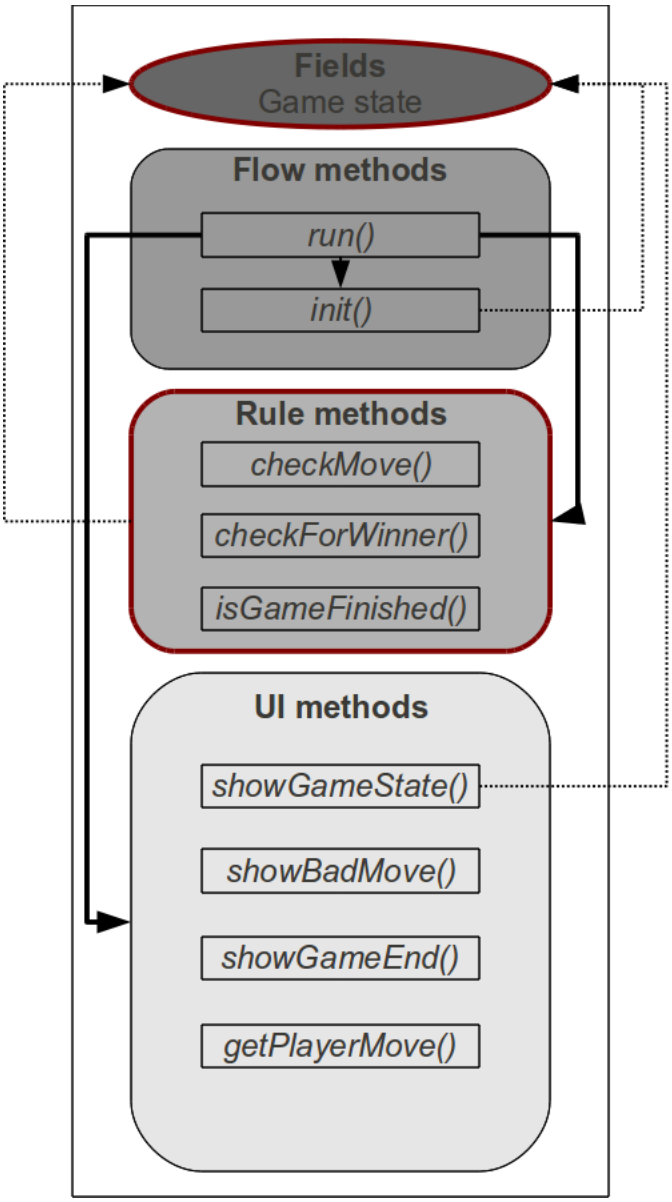


Figure 6.1: General structure of the program provided to the subjects. Methods have been grouped according to their semantic function (Game flow, game rule, UI). Continuous arrows show method calls and dashed arrows show references to class fields (global variables). The graphical arrangement (order and relative sizes of the blocks) reflects the arrangement of the code. The elements that varied between the two implementations are surrounded in red.

6.3.2 Software

We implemented a custom application, using Eclipse framework[1], dedicated to this experiment. This collaborative software aimed at showing the problems to the subjects in a syn-

[1] Eclipse is a general, multi-language Integrated Development Environment (IDE) which offers a huge programming API that can be used to develop plugins or even standalone applications.

chronous way. It consisted of a main server application, which was in charge of the general control of the application and of all data logging and of one or two lightweight client applications that were mainly the user interface. The main interface (see figure 6.2) consists of a small yellow text area on the top, which contains the instructions for the current problem, a main central text area containing the program to read and some buttons and information on the bottom to control the application.

The main area in the center shows the JAVA code to read with basic syntax highlighting. It is similar to a usual code editor but in read-only mode and without any functionality that can be present in modern editors, such as mouse-over tooltips or highlighting of all occurrences of selected words. Only selection was still present The goal was to have a code viewer as clean as possible so that we could really observe eye-movements related to code reading and understanding that were not affected by the various visual artifacts that are present in modern editors to help programmers. The counter-argument is that programmers can be so used to these artifacts that removing them could modify their way of reading code. This code area was scrollable as the whole code was larger than a single screen height. In the duo version, the scrolling was linked between the two participants. This means that if one participant scrolled up or down, this automatically scrolled the view of the other subject in the same way. This is also distant from reality but justified by the need to maintain a WYSIWIS (What you see is what I see) situation, which facilitates a lot the analysis of dual eye-movement patterns.

At the bottom, two buttons allowed subjects to answer the question, "Start recording answer" and "Finish recording answer", and one button allowed to skip the question. On the bottom-right corner, there is a time countdown indicating the time left for the current question. Above these buttons and information, a small panel reminds the general instructions on how to answer the question.

6.3.3 Participants

Fifty-seven (for the solo experiment) plus eighty-two (for the duo-experiments) students from the departments of computer science and communication science were recruited to participate in the study. They were each paid an equivalent of 20 USD for their participation in the study. The participants were typical bachelor and master students aged from 18 to 29 years old with a median of 23 years old. They had a programming experience from 1 to 13 years with a median of 4 years. The participants were paired into forty pairs without further consideration of their level of expertise, gender, age or familiarity.

6.3.4 Procedure

Upon arrival at the laboratory, subjects had to read and sign a participation agreement form. Then, for the next 3 minutes, the experimenters calibrated the eye-trackers for each of the subjects. Once both subjects were ready, the main experiment program was started. The

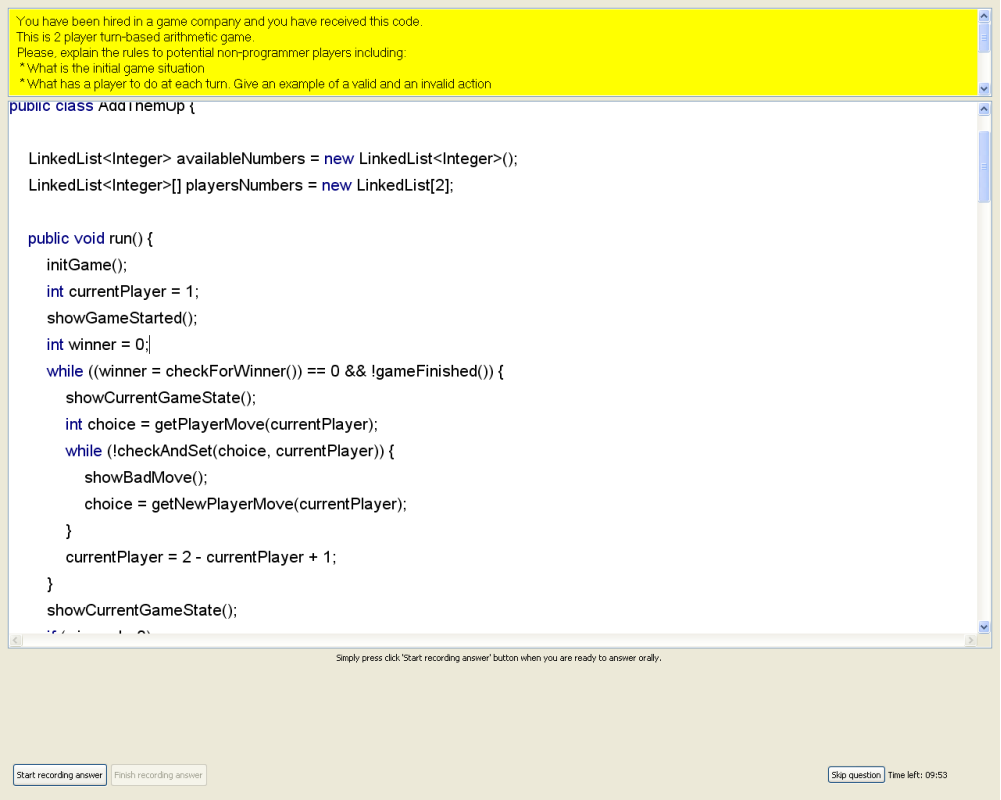


Figure 6.2: Screenshot of the software used in the experiment.

program allowed the subjects to go automatically through all the phases of the experiment without any intervention of the experimenter.

The sequence of steps they had to do was the following. First, they had to fill a short questionnaire about general information, such as age or sex, and about their programming skills and experience. Then, they received some instructions about the next phase which was the pretest. This latter consisted of answering individually to thirteen short programming multiple choice questions.

At the end of the pretest, participants received a short explanation about the following phases and for the duo experiments, about the fact that the scrolling in the code window was shared and about how text selection worked in the programming editor. Then, the main phases started with first the long program understanding question followed by the three short related questions and then, after a pause of 5 minutes, the second long question followed by five short questions.

6.3.5 Data analysis

We present in this section the various variables that were analyzed in this study. First, we present the independent variables, the task types and various code indicators, and then, the dependent variables, speech, expertise, collaboration rating and the numerous gaze features.

Task types

We categorized each question according to the type of mental processing it required. As explained in section 6.3.1, there were four different types of activity. In the first long question, they had to build up mental models of both the program and the game. We called this task *model building*. In the second long question, they had to update their model of the program to match the new implementation and we referred to this task as *model updating*. Finally, the short description question were grouped under the task type *model refining* as they had to refine their representation to explain a specific part of the program and the short bug questions were qualified as *model checking* as they had to check how well their program model matched the game model.

Programming code analysis

The Eclipse framework provides a specific API, called Java Development Tools (JDT), which offers functions for parsing JAVA code. It allows the identification of JAVA tokens[2] and to build an Abstract Syntax Tree (AST)[3] of a JAVA program. While ASTs are designed to have a meaningful representation of the code in computational terms, they do not necessarily represent well the code as it is seen by programmers. For example, several statements, such as class field declarations, are described by several AST nodes, while for a programmer, it is generally considered as a single statement. Therefore, we developed another representation that we called Semantic Tree, which aims to offer a more meaningful view of the code in cognitive terms. This tree is built from the abstract syntax tree and we describe it in the following paragraph.

Semantic tree The semantic tree consists of a root node representing the JAVA class and of several child nodes representing the methods and fields defined in the class. Each method node contains a list of child nodes representing the statements contained in the method. Each statement can be one of the following nodes: expression statement, return statement, variable declaration or control statement. Control statements are loop structures (*for* or *while* statements) or conditional structure (*if-else* statement) and contain list of child nodes representing the statements within the control structure. Expression statements comprise all

[2] A token is a string of characters, categorized according to the language rules as a symbol, e.g. identifiers, literals, syntactic signs. It represents the finest grain of elements used by the compiler.

[3] Abstract Syntax Tree is a data structure used by the compiler to represent a program before translating it into binary code.

other types of statements, essentially variable assignations and method calls. Each statement node can have one or several expressions associated to it. For example, control statements have at least one expression for describing their condition. The various types of semantic nodes are summarized in table 6.1 and a part of the tree corresponding to the program of the experiment is shown in figure 6.3.

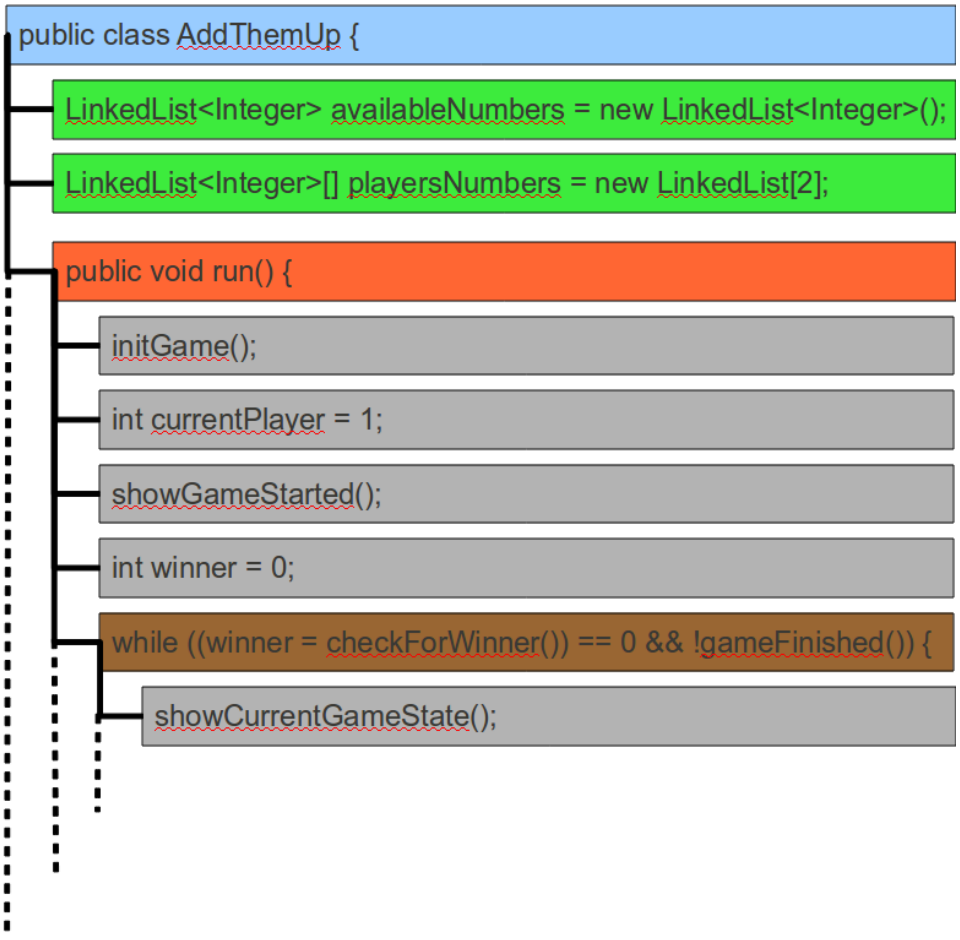


Figure 6.3: Small extract of the semantic tree of the program used in the experiment. Class node is in blue, fields nodes are in green, method node in red, expression nodes in gray and control node in brown.

Table 6.1: List of the different semantic node types. For each type, we give a description of their children if any and of their expressions if any.

Type	**Description**	**Child(ren)**	**Expression(s)**
ClassNode	The whole class	List of fields and methods contained in the class	-
FieldNode	A field definition in a class	-	The variable initialization expression
MethodNode	A method definition in a class	The list of statements composing the method	-
WhileStatementNode	A *while* loop	The list of statements in the loop body	The condition
ForStatementNode	A *for* loop	The list of statements in the loop body	The variable initialization, the condition and the increment
IfStatementNode	An *if* or *if-else* statement	The list of statements within the *if* and within the *else* if any	The condition
VariableDeclarationNode	A local variable definition in a method	-	The variable initialization expression
ReturnStatementNode	The *return* statement of a method	-	The returned expression
ExpressionStatementNode	A general simple statement, e.g. method call, variable assignation	-	The expression of the statement

Expression tree Expressions are represented separately using a specific type of tree. Each expression node represents an operator and its children are the operands. Operators are the usual unary and binary mathematical operators as well as the various usual programming operators, such as assignation, method call or array access. Leaf nodes are either variable references or literals, such as numbers or strings. From these expression trees, it was possible to compute various properties for each expression: the number of operations, the number of variable references, the presence of a variable assignation and the presence of an object creation (operator *new*). We defined the complexity of an expression as the sum of the number of operations and of the number of references

These two types of tree, semantic and expression trees, are used to relate gazes to program structure. Gazes are associated to code tokens and each token is associated to its corresponding semantic node and expression node, if there is any.

Speech

Audio volume was used as an indicator for speech episodes in the interaction. The volume for each participant was normalized as a z-score to remove differences in volume range due to the specificities of the subjects' voices. Finally, the variable was dichotomized by considering samples with a normalized volume above 0.1 as "spoken" and below 0.1 as "silent". This method is somewhat coarse and sometimes identifies parasite noise as speech (e.g. a subjects tapping their fingers on the table). However, qualitative analysis revealed that, in most cases, this simple method provides a very good estimate of the speech activity and attains similar performance as a speech detection algorithm implemented in PRAAT4[4].

Gaze data

Gazes have been recorded with the Tobii1750 using setup 4 described in section 4.2.2. Fixations were identified using the adaptive algorithm that we developed (see section 3.2.4). Systematic location errors were corrected with the brute-force method described in section 3.3.

Hit detection was accomplished on two levels, both using a probabilistic model (see section 3.4.1). First, fixations were assigned probabilities of being in the three main zones: instructions, code or bottom panel (see figure 6.2). Because the zones were big and well separated from each other, this often resulted in one zone having a fixation probability of 1. For some fixations, the probabilities were distributed between two zones, either code and instructions or code and bottom area. In a second step, fixations that had non-null probabilities of being in the code area were assigned probabilities of being in code tokens. As the code tokens are small and numerous, this generally resulted in probabilities distributed among several tokens (3 to 10).These probabilities were normalized so that the total probability over all hit tokens of one

[4]Praat is a free scientific software program for the analysis of speech in phonetics, see http://www.fon.hum.uva.nl/praat/.

fixation was always one. From the probabilities of falling into tokens, several gaze indicators characterizing the eye-movements pattern during code reading are computed. These features, except those related to expression reading patterns (see below), were computed over time-windows of 10 seconds, as well as during a whole question.

Fixation time ratio Several features represent the ratio of fixation time spent on different kinds of objects. All these features are computed with the same technique. First the hit probabilities of each fixation are grouped according to the objects on which the feature is computed. For example, to compute the ratio of fixation time on the different semantic nodes, all tokens belonging to a same semantic node are grouped into a new category and we sum their probabilities to get the probability on the corresponding semantic node. Second, we aggregate the hit probabilities of all fixations within the defined time-window (in our case, either on 10 seconds, or during a whole question). For each object of interest, the aggregated probabilities are computed as the average of the probabilities of each fixation weighted by the fixation duration. Hence, the resulting aggregate represents a probability distribution over the objects of interest which can be seen as the fixation time ratio based on probabilistic hits values. We computed several such features which aim at representing different levels of specificity of the solving process:

- The *area time ratio* is related to the general problem solving strategy. It is the ratio of fixation time spent in the three main areas: instructions, code, time and buttons. It is computed by considering only the first level of hits (zones).
- The *code element time ratio* is aimed to reflect general code reading patterns, independently of the code semantic. It is computed by grouping the tokens according to the type of element they represent. More specifically, we defined three categories of elements: structural, identifiers and expressions. Structural elements refer to keywords and syntactic signs. Identifiers comprise method and arguments names and expressions are all tokens which belong to an expression.
- The *method time ratio* characterizes eye-movements according to the semantic content of the code. It groups the tokens according to which function they belong to, with an extra category representing tokens outside any method (e.g. field declaration). The method types are defined according to which aspect of the program they are related to (see figure 6.1). These methods groups are *game flow*, i.e. methods describing the general flow of the game, *game rule*, i.e. methods implementing specifically the rule of the game such as checking a move validity or checking for a winner, and finally, *UI* methods are all methods devoted to the input and output necessary for the user interface.

Gaze semantic depth We defined the semantic depth of a semantic node as the depth of the node in the semantic tree. For example, fields and methods had depth of one, while statements

in method had depth of 2 and statements within a control structure had depth of 3, or more if several control structures are nested. We computed the *gaze semantic depth* feature as being the average depth of the semantic nodes being looked at. We proceeded by first computing the fixation time ratio on semantic nodes in the same way than for the other fixation time ratio (see above). Then, we computed the expectation of the semantic depth using the resulting probability distribution on semantic nodes.

The rationale behind such a feature is that deeper elements are generally where the actual computation is done while less deep elements are more related to structural aspects of the code. Hence, this feature somehow reflects the type of processing done on the code (e.g. scanning or checking).

Gaze dispersion The *gaze dispersion* indicates how much the fixations are dispersed on the code. It is computed as the entropy[5] of the probability distribution over the tokens aggregated over the time window of interest using the fixation duration as weights. This value is maximal when the distribution over tokens is uniform, i.e. when all code tokens are looked at in the same way and minimal if only one token is looked at. Finally, this value is corrected to take into account the area covered by the tokens concerned. Indeed, if several large tokens are looked at with a certain distribution, we should consider the dispersion as being higher than if several small tokens are looked at with the same distribution. The correction consists simply in multiplying the entropy value by the ratio of the area covered by tokens with non-null probabilities over the total area defined by all tokens.

The *gaze dispersion* reflects how focused the gazes of a subject are and we used it mainly to define episodes of different level of focus. Similarly to results of section 5.5.2, we expect it to be related to the granularity of processing with low-dispersion episodes indicating deep processing of specific expressions and high-dispersion episodes reflecting general processing of the whole code.

Expressions reading features We defined several features by analyzing dwells on expressions. An expression dwell corresponds to one or several consecutive fixations on a same expression. Each expression dwell was defined by its expression and it duration. We also computed for each expression, the *total dwell duration*, i.e. the sum of durations of all dwells done on the expression, as well as the *dwell counts* on the expression.

Due to the probabilistic nature of the hits, these features are a bit more complicated to compute. Indeed, in order to identify dwells on an expression, we need to find sequences of fixations on this expression, which is problematic as each fixation may potentially be on several expressions. Our approach was to first group token probabilities of each fixation by expression so that we obtain probabilities on expressions. From that, it is possible to

[5] In information theory, entropy is a measure of the uncertainty associated with a random variable. In our case, the random variable is the token being looked at.

compute all possible sequences of expressions dwells with their corresponding probabilities. For example, if we have two consecutive fixations each one being on two expressions, A and B, with probabilities, P_A^1, P_B^1, P_A^2 and P_B^2, then we have four possible cases: either a single dwell on A with probability $P_A^1 * P_A^2$, or a single dwell on B with probability $P_B^1 * P_B^2$, or a dwell on A followed by a dwell on B with probability $P_A^1 * P_B^2$, or finally, a dwell on B followed by a dwell on A with probability $P_B^1 * P_A^2$. From this, we can compute the expectation of our indicators, e.g. the *expression dwell duration*, by computing it in each case and by multiplying it by the case probability. While this is clearly the best approach to take full advantages of the probabilistic hits, it is difficult to apply in practice because the number of cases grows exponentially. Indeed, this makes a tree in which each fixation makes a new level with K leaves where K is the number of different expressions associated to the fixations. A computationally more feasible approach is to sample the expression sequences. Instead of building all possible expression sequences, we sample a certain number of sequences (in our case 1000) by selecting randomly for each fixation an expression according to the expression probabilities associated to the fixations. Then, we simply computed the average of the indicators over the sampled sequences.

These expression dwells indicators aim to characterize code reading patterns by focusing on the most important reading activity. Indeed, expressions generally represent the core of the program as it is where the specific operations are implemented.

Gaze divergence We also computed a dual gaze feature, the *gaze divergence*, to measure the overall non-similarity between the gazes of the two subjects. It is defined as one minus the cosine between the vectors representing the probability distribution over all tokens of both subjects. Thus, the resulting value is one if subjects looked only at different tokens and 0 if they looked at the same tokens in exactly the same way, i.e. they spend the same amount of time on each token.

This *gaze divergence* is a measure of gaze togetherness and thus, should be related to moments of joint processing. Hence, we used it to define episodes of high and low joint attention in order to see how code exploration is affected by collaboration.

Cross-recurrence Finally, we also performed cross-recurrence analyses on a per question basis. We used a zone-based approach (see section 4.3.4) by considering the different areas (code, instructions, buttons) as well as the different methods in the code as zones. We sampled the fixations every 200 ms and for the recurrence test, we tested whether the probability that the two fixations were on the same token was higher than 0.3. We also applied a special correction to take into account the scrolling so that we do not compare gazes which could not be recurrent because they happened on different viewport. Indeed, as we have to compare fixations happening at different moments in time, it is possible that the screen scrolled in-between. More specifically, when comparing two fixations, we checked that they were on an area that was also reachable by the other fixation and we considered as missing data when

it was not the case. The rationale is that we should not consider points as not-recurrent if they have no chance to be recurrent. Finally, we also computed the cross-recurrence during speech episodes and during non-speech episodes separately. We used the normalized speech feature defined from the volume (see above). We used the gaze data only during the seconds tagged as speech (respectively non-speech) to compute the cross-recurrence during speech (respectively non-speech). The value that was finally analyzed was the gaze coupling computed using formula 4.6 (see section 4.3.4).

Answers rating

Answers of the first main question (*model building* question) were rated manually for pairs of the duo experiments. Each pair was assigned a score between 0 and 2 indicating how well they understood the program. A score of 2 indicated that they understood correctly the main rules of the game, i.e. that the players had to obtain three numbers that add up to 15. A score of 1 indicated that they got some idea about the rule, for example, that we should have a sum of 15 to win and score of 0 indicated that they had no idea about what the program does.

Collaboration flow rating

The *model building* question was also rated for the quality of the flow of interaction. We assigned to each pair a general score, 0, 1 or 2, which assessed how good was the collaboration overall. It is a very general collaboration process score which differentiates between pairs having good collaboration flow, i.e. good grounding, joint attention, mutual coordination, good turn-taking, versus pairs having bad collaboration (e.g. one subject completely lost and the other doing the whole job alone). This rating was inspired from a more detailed rating scheme consisting of 12 dimensions developed by Meier et al. (2007). Here, we used a simplified version with a single general dimension in order to get a rough idea of the collaboration quality. Hence, it should not be considered as a strong measure but rather as a general indicator.

Expertise and group composition

We defined the expertise level of each subjects of both experiment from their score at the pretest questions. More specifically, we considered as experts all subjects having at least 8 (out of 13) correct answers at the pretest and the others as novices.

6.4 Results

We first present gaze indicators from the solo experiment to identify patterns of eye-movements related to code reading and understanding and we analyze the differences in these patterns between the tasks. Second, we analyze the effects of collaboration and expertise on some of the code reading patterns, namely we focused on the patterns that exhibited significant

differences along these dimensions. Finally, we look at dual-gaze features and their relation to the collaboration process and collaboration quality.

6.4.1 Individual patterns

To get a good general overview of eye-movements during program understanding task, we analyze in detail gaze features from the solo experiment. The results are presented in order of decreasing generality. First, we present general indicators related to the problem solving in general and progressively, we analyze more specific variables related to code reading in general and to the understanding of the specific code used in the experiment.

Problem solving strategy

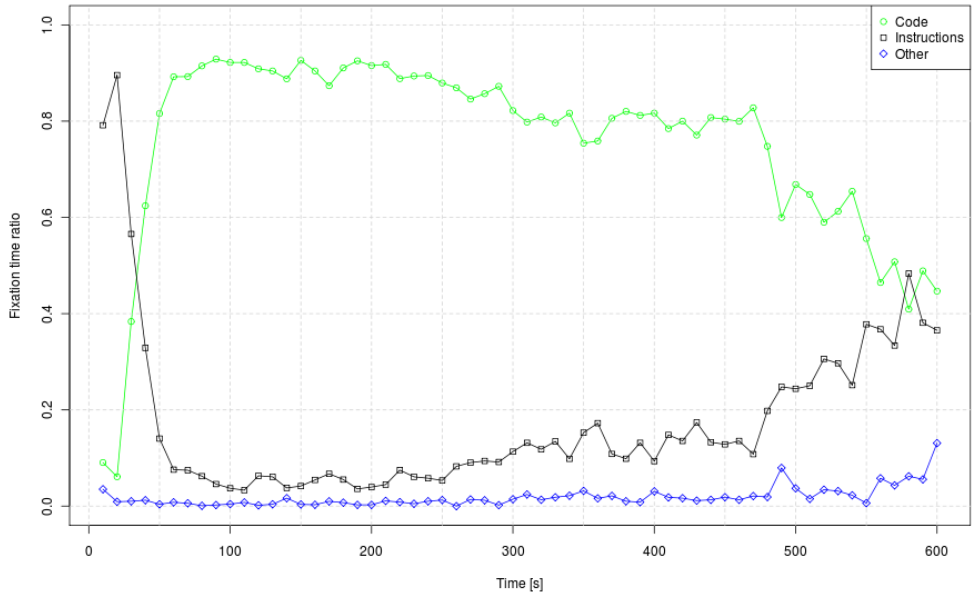


Figure 6.4: Evolution through time of fixation time ratio on the zones of the interface for the *model building* question.

We analyze the evolution of the fixation time ratios on the interface zones for the different types of questions. For the two main questions (figures 6.4 and 6.5), the general pattern is roughly the same and we can identify three main phases. At the beginning, subjects spend a lot of time on the instructions (more than 80%) and progressively look more and more at the code area. This is an obvious instructions reading phase required to do the question. After around 50 s, gazes in the code area reach a maximum with around 90% of the fixation time. This represents a sort of code discovery phase in which subjects look almost only at the code in order to understand it. After around half of the time of the question (around 300 s for the *build* question and around 150 s for the *update* question), the ratio of time in the code area

decreases to around 80%, which is explained partly by an increase of time on the instructions but also by a slight increase of time spent in the other area, i.e. time countdown and answer buttons. This is certainly an answering or answer preparation phase for which subjects get back to the instructions to check the question but also check the remaining time and possibly look at the answer buttons. In the *build* question, we can see a fourth phase at around 500 s in which the time spent on code decreases continuously up to end of the question to reach a level of only 50%. This could be explained by the fact that this question had several sub-questions which forced subjects to give detailed explanations of what the code does. Hence, this could explain the huge increase of gazes on instructions at the end as they have to check and answer all the sub-questions.

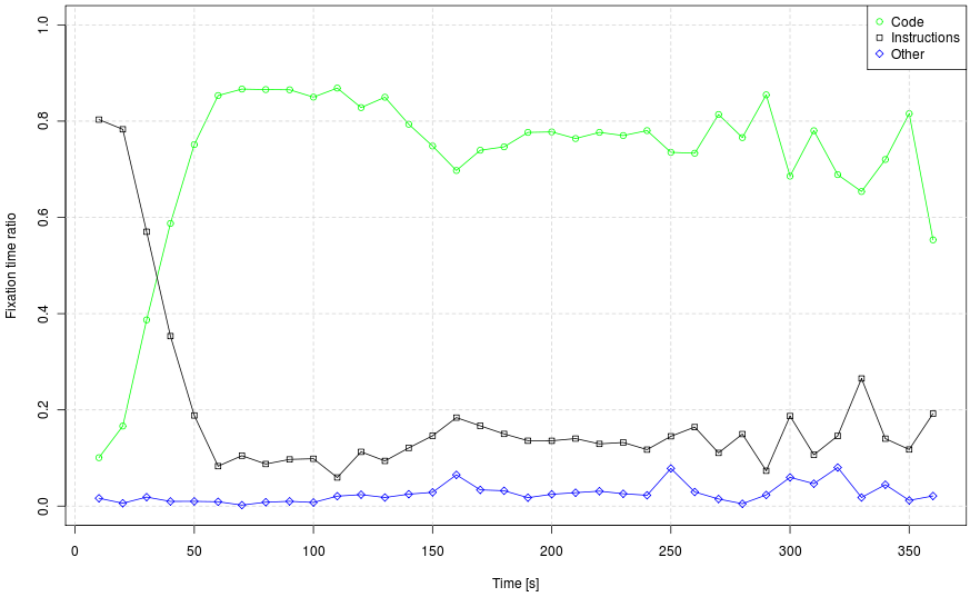


Figure 6.5: Evolution through time of fixation time ratio on the zones of the interface for the *model updating* question.

For the short questions (figure 6.6), i.e. describing (*model refining*) and debugging (*model checking*) questions, the data have been merged as they depicted very similar patterns. We have the same instructions phase as for the two main questions but then, we see only an answering phase with around 70% of the time spent in the code area. This is well explained by the fact that the code is already known, so participants don't need to discover it. Also, the lower ratio on code indicates that they need to refer more often to the instructions which could be explained by the fact that often, the instructions contained some examples of program outputs that could be useful to answer the question.

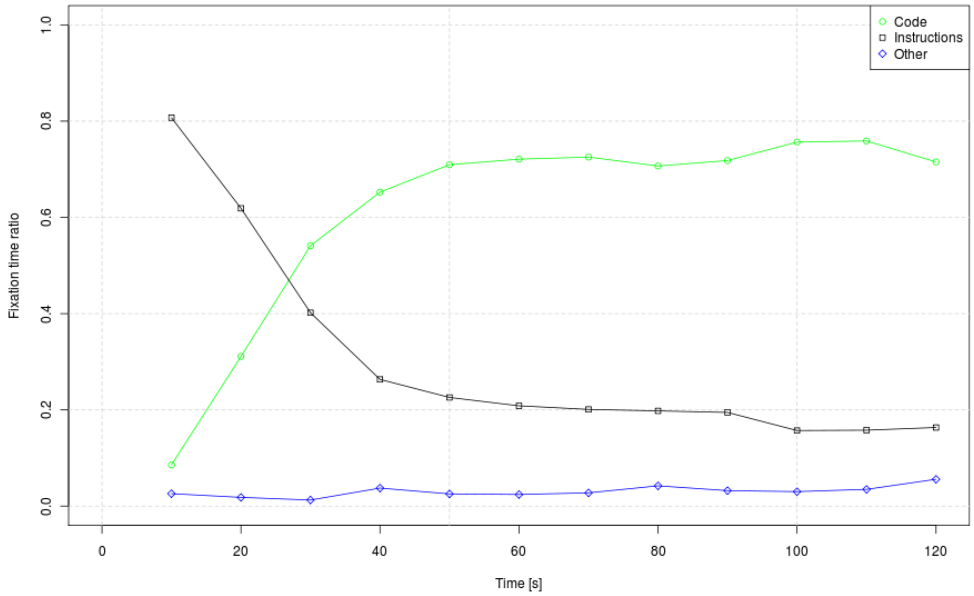
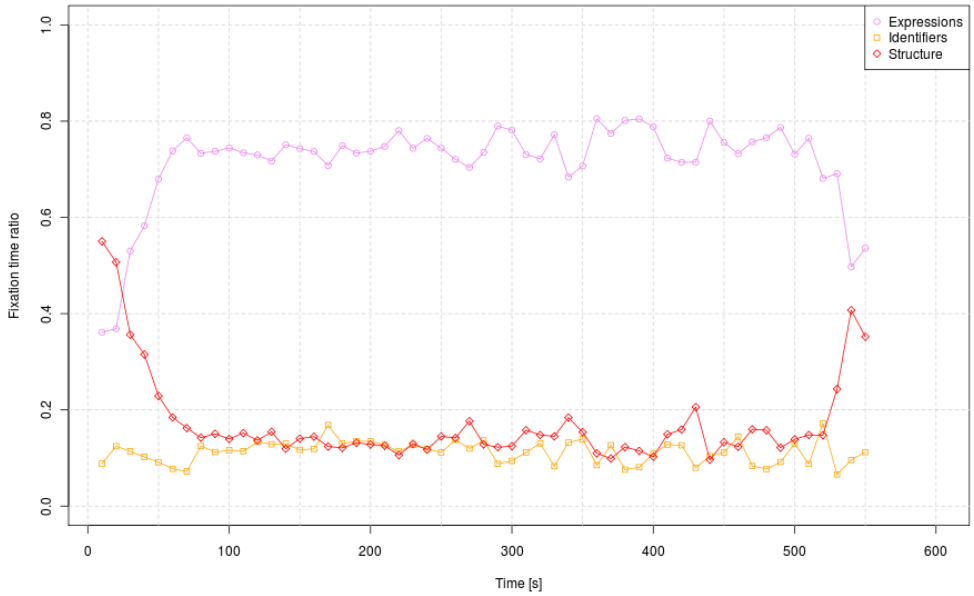


Figure 6.6: Evolution through time of fixation time ratio on the zones of the interface for the *model refining/checking* questions.

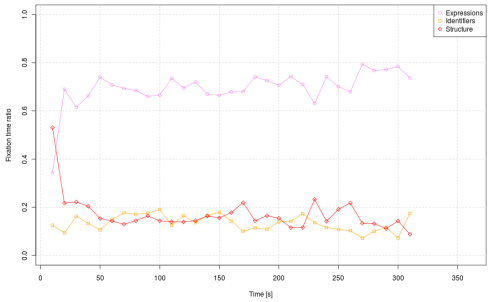
Code reading

To characterize general code reading patterns, we analyzed the ratio of fixation time spent on the different code elements before subjects start to answer (see figure 6.7). Again, we can see similar patterns in the two main questions (figures 6.7a and 6.7b) with two phases. First, we have a "structural" phase with a high amount of gazes on the structural elements which decreases rapidly during the first 50 seconds. It suggests that people first briefly scan the main structural elements before looking at the computational part of the code, i.e. the expressions. This may be a way to get an overall picture of how the code is structured. After this "structural" phase and up to the end of the question, they enter an "expressions decoding" phase and spend most of their fixation time on the expressions, which is where the code semantic is located. We can note that the "structural" phase is slightly shorter in the *model updating* question (50 s versus 70-80 s) which can be explained by the fact that the overall structure was very similar to the first version of the code that they had already seen in the first question. Also, during the "expressions" phase, the ratio on expression is slightly lower in the *model updating* question. The time spent on the identifiers remains constant over time in all questions. The *model checking/refining* questions depict the same patterns but also with a shorter "structural" phase, which is also explained by the fact that the code is already known.

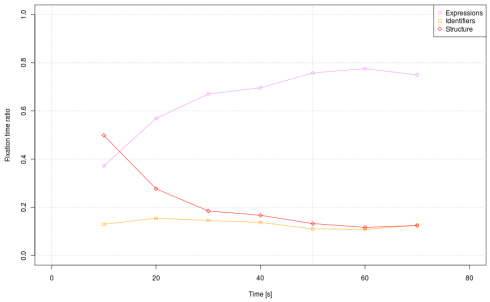
Then, we looked at the overall time ratio on the different code elements before and during the answer (figure 6.8). For all types of question, we clearly see that, during answer, there is a much higher amount of time spent on the expressions (80% instead of 70-75% before the



(a) Model building



(b) Model updating

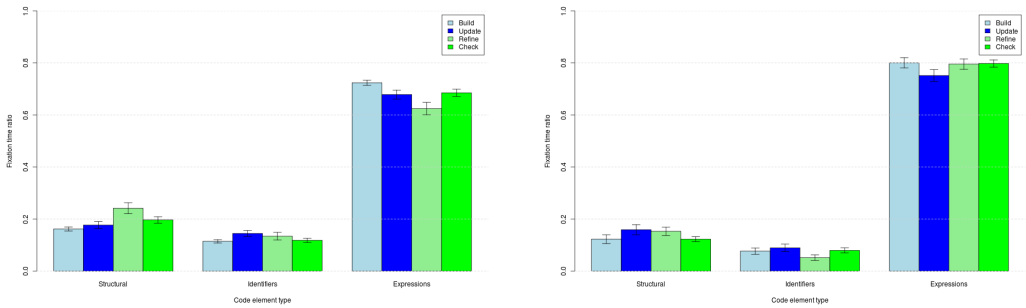


(c) Model refining/checking

Figure 6.7: Evolution through time of fixation time ratio on the code elements for the different types of questions before starting to give the answer.

answer) and a lower amount on structural elements. This indicates that subjects mainly look at expressions during the answer. This may be a manner to support their explanations by looking at the meaningful expressions while explaining the code semantic.

Finally, figure 6.9 shows the gaze semantic depth over time for the different question types. We first note that participants always start to look at low depth parts of the code and rapidly dive into deeper aspects of the code. This is consistent with the previous finding which indicates that people start by having a global look at the overall code structure before digging into the semantic parts (expressions). We also notice the difference of increase for the *model refining/checking* questions for which the semantic depth increases more rapidly and reaches



(a) Before answer

(b) During answer

Figure 6.8: Overall fixation time ratios on the code elements before and during answer for the different question types.

a higher level. An explanation is that, in those questions, subjects need to locate and examine in detail a specific part of the code. Hence, as they already know the whole code, they simply have to search for that part and then to remain focused on it, which explains both the quicker increase and the higher depth reached. On the opposite, in the two main questions, participants have to move between different regions of the code which explains a lower overall semantic depth. It is also interesting to see that for the *model building* question, the semantic depth tends to increase continuously after the rapid initial increase. This may suggest that subjects look at deeper and deeper parts of the code. On the contrary, in the *mode updating* question, the semantic depth rather seems to decrease slightly.

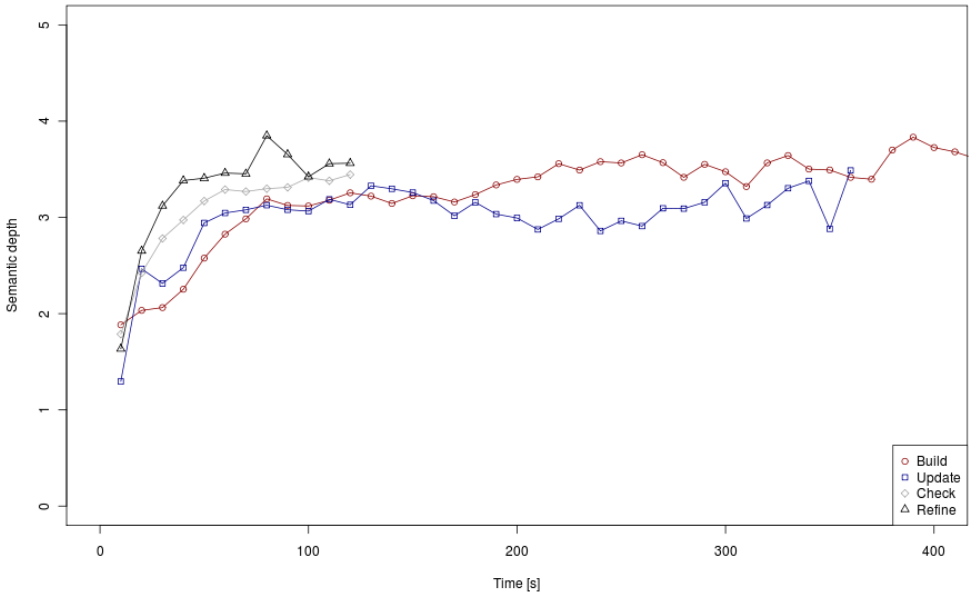


Figure 6.9: Semantic depth through time for each question type

Expressions reading

As expressions represent the most meaningful parts of the code, we analyzed more precisely the dwells on expressions. More specifically, we looked at the dwell duration according to the complexity of the expression (see figure 6.10). We see an interesting pattern with the dwell duration increasing roughly linearly with the number of operations in the expression. This indicates that we could define a sort of expression reading speed in terms of number of operations read by millisecond. Actually, the situation is a bit more complex because we have a minimum dwell duration of around 300 ms and then, for each additional operation, the dwell duration increases by some amount. We conducted a linear mixed effect analysis, using the subject as a random effect, to get numerical estimates of these values. We found an intercept of 431 ms and a significant effect of the complexity of 35 ms, which means that in average subjects have a minimum expression dwell duration of 431 ms and their dwell duration increases by 35 ms for every additional operation in the expression. (Note that the values do not correspond to the figure 6.10 because of the random effect due to the subject are taken into account in the statistical analysis.) We conducted a second linear mixed effect analysis by using also the task type as a predictor. This showed a significant effect of the task type but not of the interaction between the task type and the number of operations. This indicates that the additional dwell per operation remains constant through the task type. However, the minimum dwell duration on an expression appeared to be slightly higher in the two main questions (88 ms for *model building* and 75 ms for *model updating*) compared to the short questions.

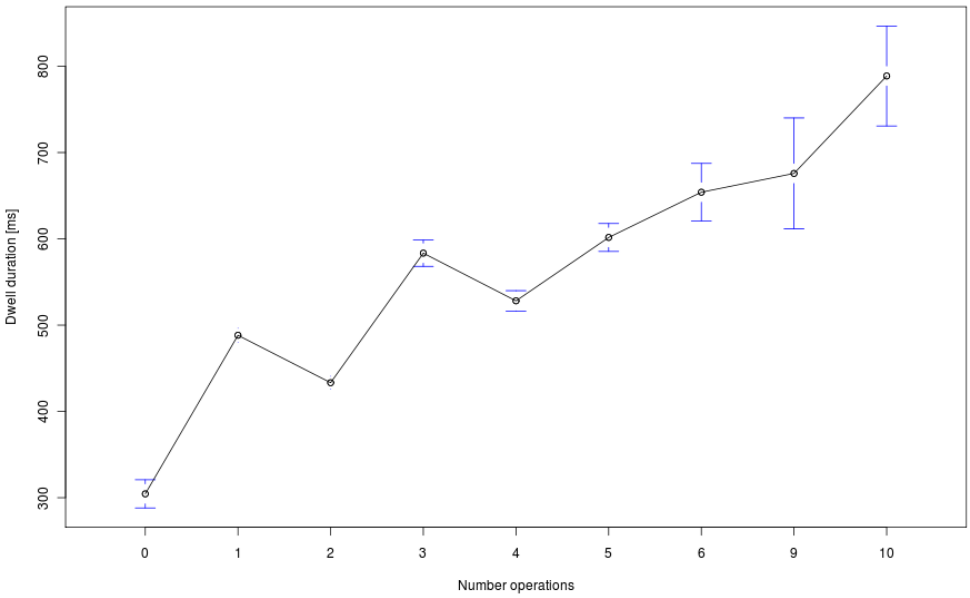


Figure 6.10: Dwell duration according to the complexity of the expression for all task types.

Program semantic reading

We explored indicators that were specifically related to the semantic of the program shown to the subjects. We plotted on figure 6.11, the ratio of code fixation time spent on the different types of methods for the two main questions. First, we note that for the *model building* question, very little time (10%) in spent outside of the methods and in the *ui* methods. This is consistent with the fact that these parts contain little information for the semantic of the code. On the opposite, a lot of time is spent on the *flow* and *rule* methods which contain the actual implementation of the game semantic. It is also very interesting to see how these ratios change in the *model updating* question. Indeed, for this latter, more fixation time is spent outside the methods and less time is spent in the *flow* methods. This is coherent if we consider first, that one of the main changes in the program was a change of the game state which is represented by a class field outside of any method, and second, that the most similar part between the two versions of the program was the game *flow* function. Hence, subjects had been able to identify where the differences were and optimized their time of analysis accordingly. Note that we did not do this analysis on the short questions because those questions were each about a specific region of the code which could be in any of the method type. Hence, any effect on the method type would be simply due to the location of the problem to solve.

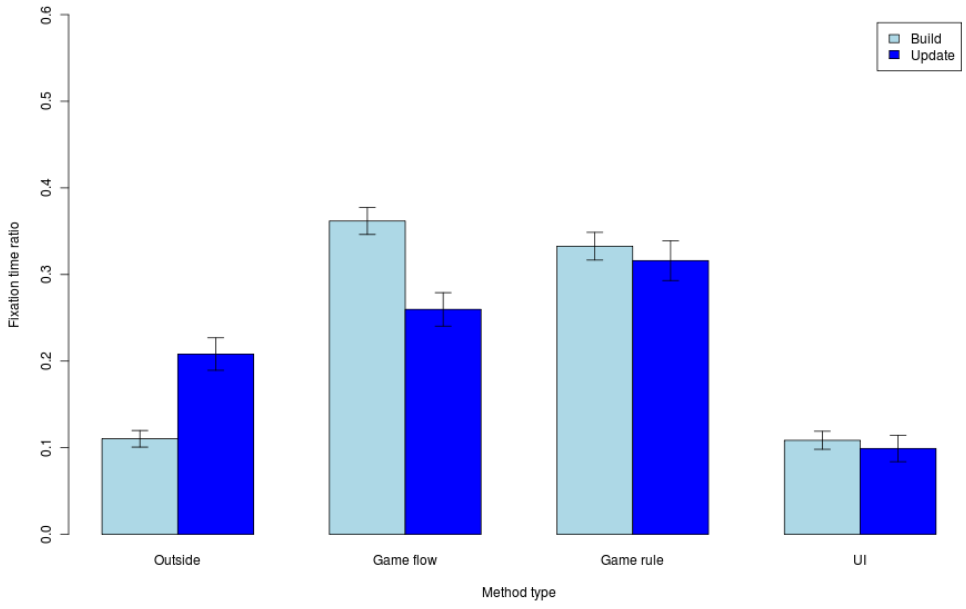
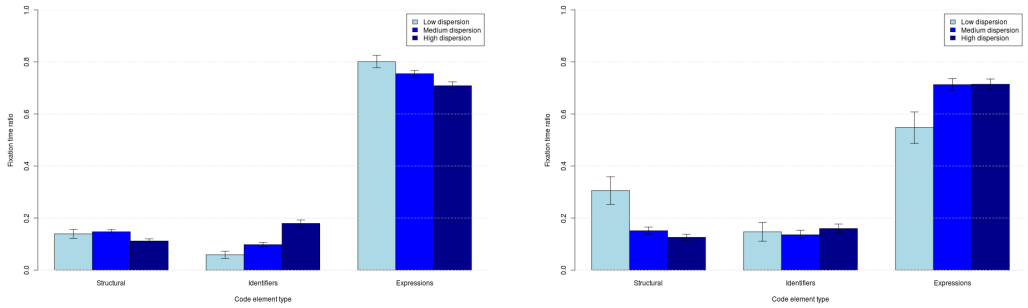


Figure 6.11: Fixation time spent on the various method types for the two main questions.

Reading episodes

We explored code reading episodes by using the gaze dispersion indicator to check whether dispersion reflects the type of cognitive processing. The hypothesis is that low dispersion

indicates a focused, deep processing and high dispersion indicates a more general overview processing. We defined levels of dispersion by using the quantiles of dispersion during the two main questions. Low level dispersion episodes have been defined as being below the first quartile and thus consist in 25% of all 10 seconds moments of all subjects while high dispersion has been defined as being above the third quartile. All other moments (50% of the moments) have been defined as medium dispersion. We plotted on figure 6.12, the fixation time ratio on the code elements for the three levels of dispersion for the two main questions. First, in the *model building* question, we see that an increase of dispersion is correlated with a decrease of fixation time on expressions, an increase on structural elements and an increase on identifiers elements. This was confirmed by a linear mixed effect analysis using the code element and the dispersion as predictors and the subject as a random effect which showed a significant effect of the code element and of the interaction between the code element and the dispersion. More specifically, it indicated a decrease of 35% of the fixation time ratio on expression for the maximum increase of dispersion. This is consistent with our hypothesis of a deeper processing during low dispersion episodes as expressions represent the complex and specific parts of the code



(a) Model building

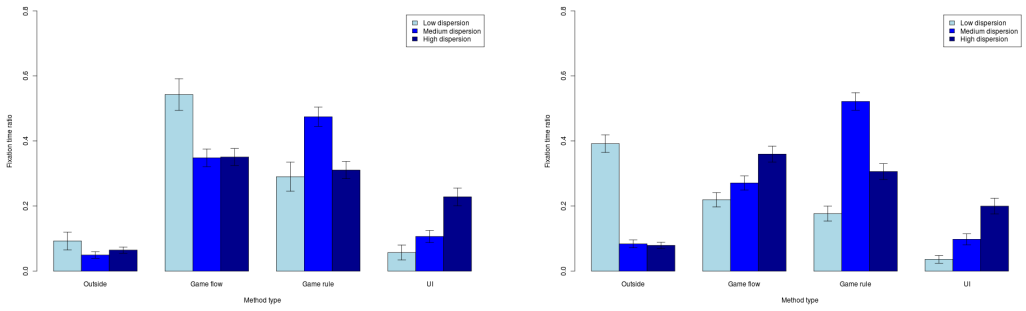
(b) Model updating

Figure 6.12: Fixation time ratio on code elements for different levels of dispersion.

We also looked at the fixation time ratio on the method types for the different levels of dispersion (see figure 6.13). In this case, the effect of dispersion is not linear but we can note several interesting differences. We first conducted a linear mixed-effect analysis for the *model building* question using the method type and the dispersion level as two categorical predictors and the subject as a random effect. Both the method type and the interaction between the method type and the dispersion level appeared to be significant. We discuss hereafter the differences that are significant. We can notice that the amount of fixation time on *UI* methods increases with the level of dispersion (+ 5% for medium dispersion and + 17% for high dispersion). This is also consistent with our hypothesis as we can expect that people look at *UI* functions mainly when they are getting an overall picture of the code and that they don't inspect in details these functions. We also see that the amount of fixation time on *flow* methods is much higher (54%) during low-dispersion episodes while it keeps the same level (35%) during medium and high-dispersion episodes. On the opposite, the amount of fixation time on *rule* methods

is higher (47%) during medium-dispersion episodes and has the same level (31%) during low and high episodes. This suggests that *flow* and *rule* methods involve different kinds of processing. This seems to not agree very well with our hypothesis as we would have expected a high ratio of fixation time on *rule* methods (the most complex part) during low dispersion episodes. A good explanation for this effect is that when analyzing *rule* methods, subjects had to trace the calls of these functions, which were done in the *flow* function, as well as the variables affected and thus they had to explore different parts of the program which explains a medium level of dispersion. In any case, this remains globally consistent with our hypothesis as, both ratios on *rule* and *flow* methods are smaller during high-dispersion episodes than during low or medium-dispersion episodes.

We conducted a similar linear mixed-effect analysis for the ratios during the *model updating* question. We see that the effect of dispersion are very different than for the first main question, except for *UI* methods which are affected in a similar way (+ 6% from to low to medium and + 16% from low to high). In this case, the fixation time on *flow* methods increases as the dispersion increases (+ 7% for medium dispersion and + 14% for high-dispersion). This is explained by the fact that they already know this part quite well as it didn't change much and thus, they just look at it as a reminder to get an overall picture of the code. The effect of dispersion on *rule* methods is similar that in the *model building* question with a higher level during medium-dispersion episodes. This is coherent as this is one part that changed the most. Hence, it requires the same kind of processing than during the *model building* task. Finally, there is huge effect on the fixation time outside method for low dispersion episodes (+ 31%) compared to mid or high dispersion episodes. It corresponds to the time spent looking at the fields declaration which are, along with the *rule* methods, the other main change of the program. Thus, this explains why they have to spend so much time on it.



(a) Model building

(b) Model updating

Figure 6.13: Fixation time ratio on method types for different level of dispersion.

6.4.2 Expertise and collaboration

Having identified general patterns of code reading for individual programmers, we now move to the analysis of programmers collaborating to solve the same problems. Our goal is to identify

the effect of collaboration on eye-movement patterns. At the same time, we are interested in analyzing the effect of expertise on these patterns. We tackle these two questions at the same time because the proportion of experts is not the same in the solo and duo experiments. Hence, if we analyze the effect of collaboration alone, we could possibly see a combination of both effects and make wrong interpretations.

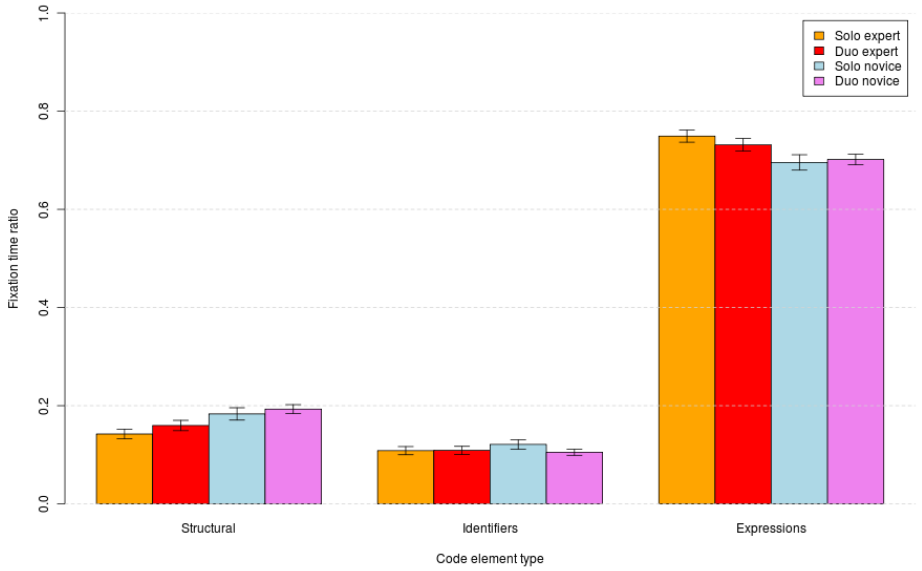


Figure 6.14: Fixation time spent on the code elements in the *model building* task compared between expertise levels and conditions.

We analyzed the ratio on the code elements for the two levels of expertise and for solo and duo condition (see figure 6.14) and we conducted a linear mixed-effect analysis by using the interaction of both factors with the code element as predictors and the pair and the subject as random factors (see table 6.2). As we can see, both the expertise and the collaboration affect significantly the ratio on code elements while the interaction between the two is not significant. More specifically, it appeared that expertise increases the fixation time on expressions (+5.5% for experts) and decreases the fixation time on structural elements (-4% for experts). This is an interesting effect which shows that experts spend more time on the semantically more meaningful parts of the code while novices require more time to identify the overall structure of the code. The effect of collaboration is quite different and concerns essentially the fixation time on identifiers (12% in solo versus 10.4% in duo). Considering the low value of this ratio, the effect is not negligible. This means that collaborators need less to look at method or parameters names. A possible explanation would be that by discussing of them, they remember them better and need less to look at them. However, this could also mean that, because of the collaboration, they spend more time on expressions and structure in general.

We conducted similar analyses for the *model updating* question (see figure 6.15). The effects appeared to be very different from the first main question with no effect of expertise alone

	numDF	denDF	F-value	p-value
(Intercept)	1	15775.00	45716.55	0.00
code element	2	15775.00	15237.75	0.00
code element:collaboration	3	15775.00	5.95	0.00
code element:expertise	3	15775.00	30.49	0.00
code element:collaboration:expertise	3	15775.00	2.27	0.08

Table 6.2: Results of the linear mixed-effect analysis on the ratio of fixation time spent on the code elements in the *model building* task

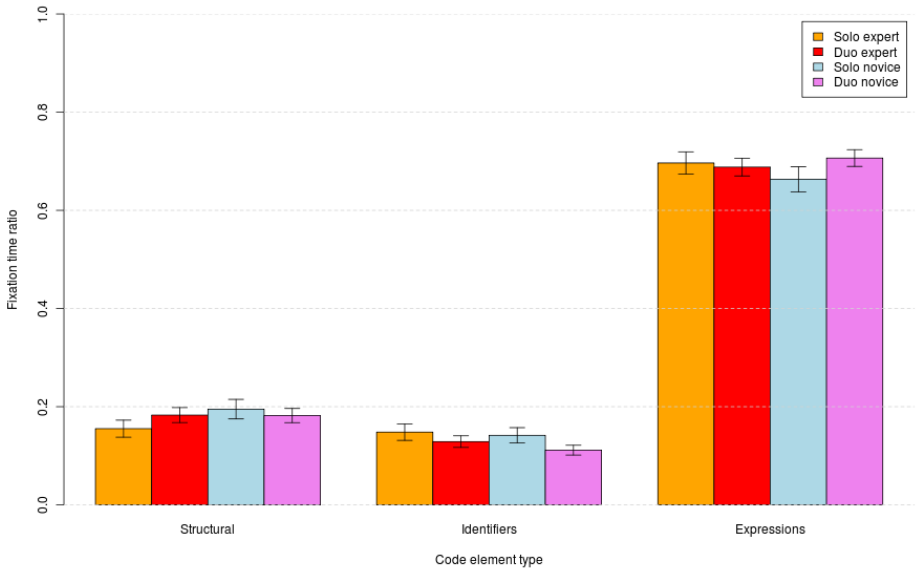


Figure 6.15: Fixation time spent on the code elements for the *model updating* task

but instead an effect of the interaction between expertise and collaboration. The effect of collaboration is similar to the first main question with a difference 3% more fixation time on identifiers for duo condition. The interaction effect concerns the structural elements ratio which is lower for experts alone compared to the three other categories. This latter effect could mean that collaboration is detrimental to experts as their gazes in a collaborative situation tend to be more like novices' ones.

The semantic depth during the *model building* task appeared to vary with expertise levels and condition (See figure 6.16). It appeared that both collaboration and expertise affect the semantic depth but there was no interaction between the two factors. Actually, the expertise increases a bit the average semantic depth and the collaboration decreases it a bit. While we could expect such an effect of expertise, the effect of collaboration is more surprising. It is interesting to note that collaboration seems to have an effect opposite to expertise. This may also indicate that collaboration may have a detrimental effect as it makes people to look

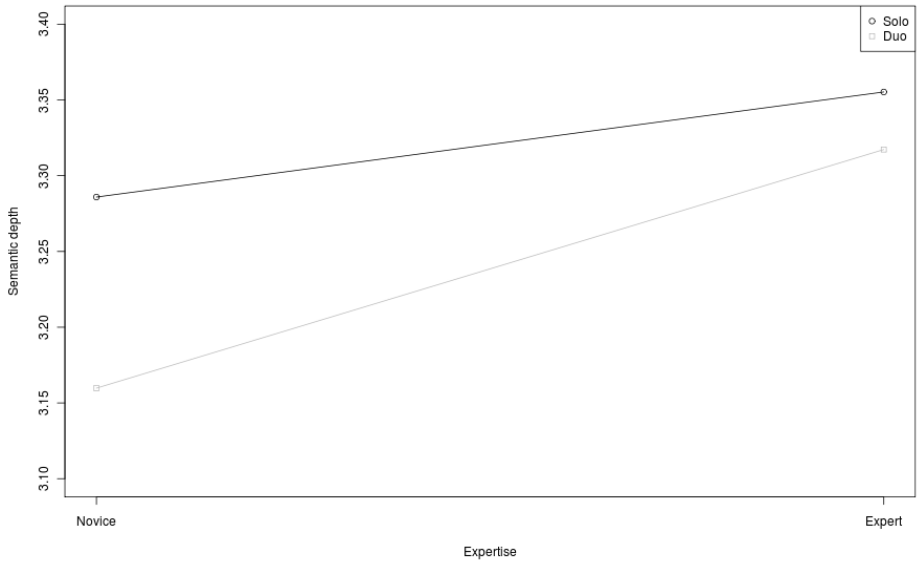


Figure 6.16: Semantic depth in the *model building* task versus expertise and condition

more like novices. There were however no such effects, neither for the expertise, nor for the collaboration, in the *model updating* task.

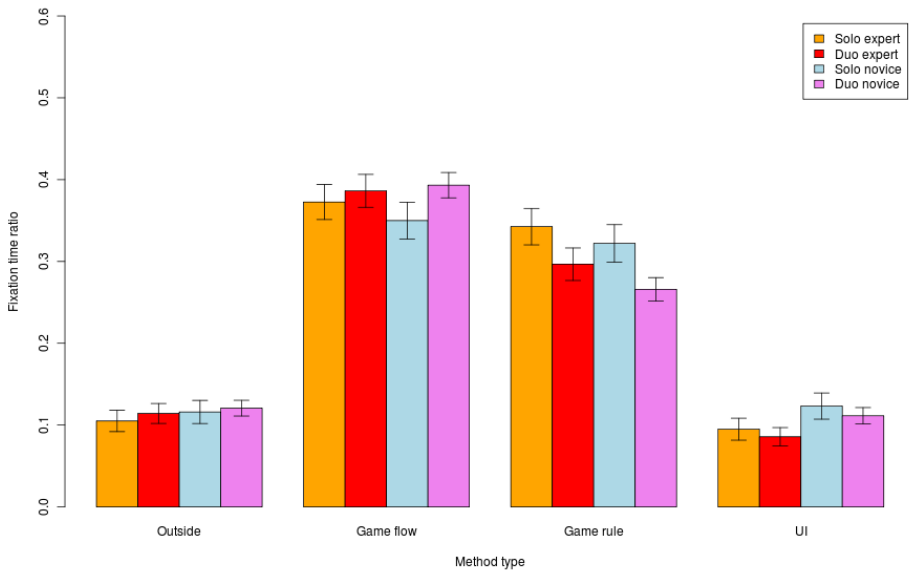


Figure 6.17: Fixation time ratio on the method types in the *model building* task versus expertise and condition

Finally, we explored the fixation time ratio on the method types for the expertise and collaboration factors (see figure 6.17). Results from a mixed effect analysis show that expertise and

collaboration affect these ratios independently (see table 6.3). The main effect of expertise concerns the amount of fixation time on the *UI* methods which is higher for novices (12.3%) than for experts (9.5%). This suggests that novices lose time at analyzing these functions while experts optimize their time to look more at the semantic parts. Collaboration increases the fixation time on *flow* methods (+4.3%) and decreases it on *rule* methods (-5.6%). This tends to show that collaborators spend more time on the global aspects of the program than on the specific details, which again can be seen as a detrimental effect of collaboration on gaze patterns.

	numDF	denDF	F-value	p-value
(Intercept)	1	27603.00	12156.20	0.00
Method type	3	27603.00	1123.43	0.00
Method type:collaboration	4	27603.00	14.20	0.00
Method type:expertise	4	27603.00	5.43	0.00
Method type:collaboration:expertise	4	27603.00	0.85	0.49

Table 6.3: Results of the linear mixed-effect analysis on the fixation time ratio on the method types according to expertise and condition in the *model building* task

We performed similar analyses for the *model updating* task (see figure 6.18 and table 6.4). However, the results appeared to be very different with effects of the expertise, the collaboration but also of the interaction between the expertise and collaboration. First, collaboration increases the amount of fixation time on *UI* methods (+5.5%) and decreases the same amount of fixation time outside any method. Second, expertise decreases the amount on *flow* methods (- 10.0 %) and increases the amount on *UI* methods (+5.0%) and on *rule* methods (+3.6%). Finally, the combined effect of expertise and collaboration decreases the amount on *UI* methods and increases the amount on *flow* methods. Expertise effect on *flow* methods is consistent with the fact that, in this question, the *flow* methods are less important as they are already known. However, it is difficult to explain why experts spend so much time on *UI* methods. In this respect, collaboration seems to cancel out this effect as experts in collaboration look very little at the *UI* methods.

	numDF	denDF	F-value	p-value
(Intercept)	1	13683.00	5865.67	0.00
Method type	3	13683.00	308.75	0.00
Method type:collaboration	4	13683.00	3.65	0.01
Method type:expertise	4	13683.00	19.65	0.00
Method type:collaboration:expertise	4	13683.00	11.31	0.00

Table 6.4: Results of the linear mixed-effect analysis on the fixation time ratio on the method types according to expertise and condition in the *model updating* task.

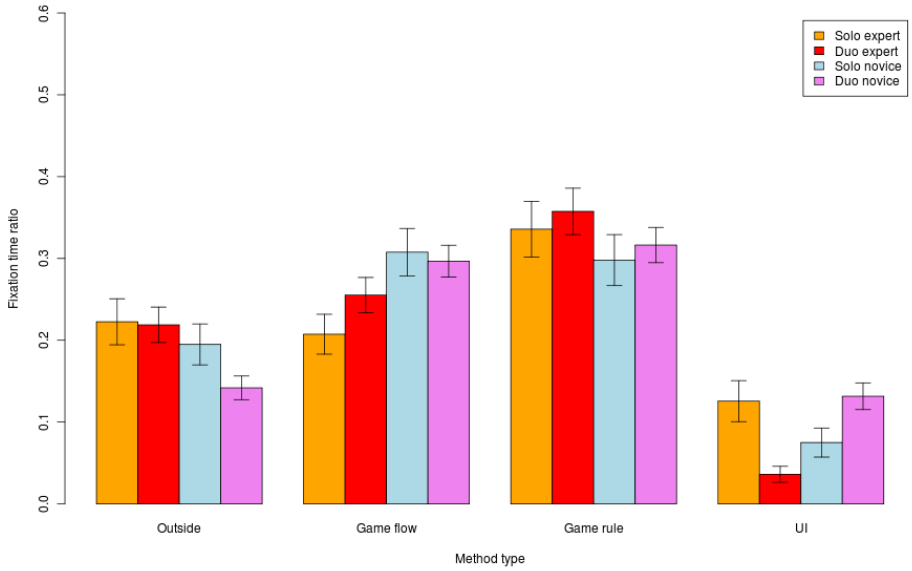


Figure 6.18: Fixation time ratio on the method types in the *model updating* task versus expertise and condition

6.4.3 Collaboration and dual-gaze patterns

Finally, we explore dual-gaze patterns and their relationships to the collaborative process. First, we analyze the process in detail by comparing gaze patterns during episodes of low gaze divergence versus episodes of high gaze divergence, which brings some insights about the type of cognitive processing that is performed jointly by the collaborators. Second, we are interested in global features that reflect the quality of the collaboration as it is measured by our collaboration ratings. In this respect, we concentrate our attention on both the divergence feature and also on the cross-recurrence feature.

Divergence episodes

As for the dispersion feature, we used the gaze divergence between collaborators to define episodes by using the quartiles of the divergence distribution. Thus, we defined low-divergence episodes as all 10s time-windows having divergence values lower than the first quartile (25% of all episodes), high-divergence episodes as all moments with divergence higher than the third quartile and all other episodes as being medium-divergence episodes (50% of the cases). First, we analyzed which types of code elements are looked at during episodes of different levels of divergence for the *model building* and *model updating* questions (see figure 6.19). There was a significant effect of the divergence level on the fixation time on code elements. Interestingly, we see that the amount of fixation time on expressions decreases with the divergence while the amount of fixation on structural elements and on identifiers increases. This means that collaborators tend to do deeper processing when they are working

together, i.e. when they have a low-divergence, than when they are working alone.

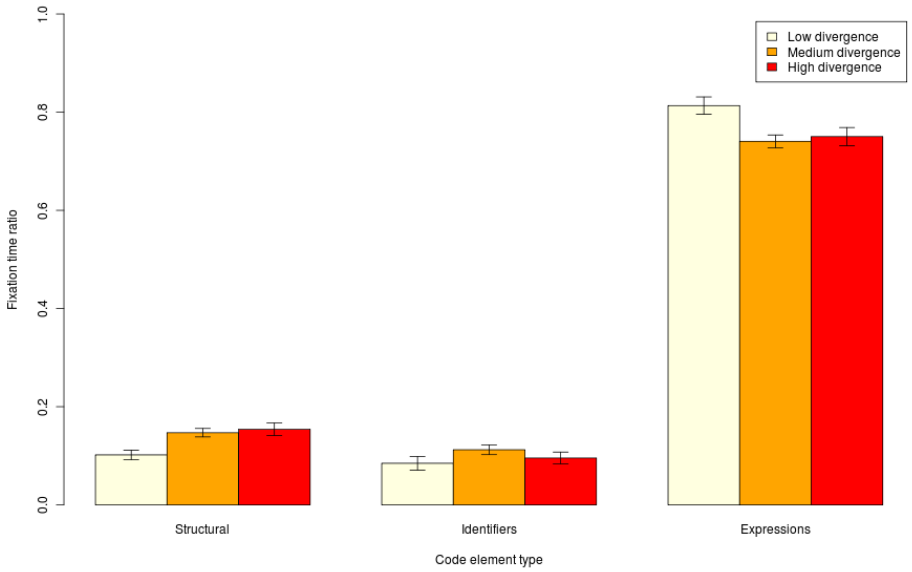


Figure 6.19: Fixation time ratio on code elements for different level of divergence for the *model building* and *model updating* questions.

We also looked at the average gaze dispersion of the collaborators for the different levels of gaze divergence (see figure 6.20). The results show that the dispersion level is lower during low-divergence episodes than during mid or high-divergence episodes. This confirms the previous results of a deeper processing during low-divergence episodes as we have seen that low-dispersion is associated to processing of specific elements (see section 6.4.1). Moreover, we also looked at the absolute difference of gaze dispersion between the two collaborators (see figure 6.21). We see that the difference of dispersion also tends to be lower during low-divergence episodes, which means that they both tend to be focused during those episodes while during, mid or high divergence episodes, there are more chances than one is highly dispersed while the other is lowly dispersed. These two results show that joint attention is mainly associated with deep processing and that general overview processing is done mainly alone.

Finally, we analyzed the fixation time on methods for the two main questions during those episodes. First, for the *model building* question (see figure 6.22), we see that it is mainly the amount of fixation time on *flow* and *rule* methods that that varies with the divergence level. Indeed, during low-divergence episodes, the amount of fixation time on *flow* methods increase (55% instead of 45% for high-divergence and 40% for medium-divergence) while the amount of fixation time on *rule* methods decreases (27% instead of 32% for high-divergence and 36% for medium-divergence). This indicates that during joint attention episodes, collaborators work more on the flow aspects of the game. This may seem inconsistent with the previous results which suggest that low-divergence is associated with deeper processing. However, if

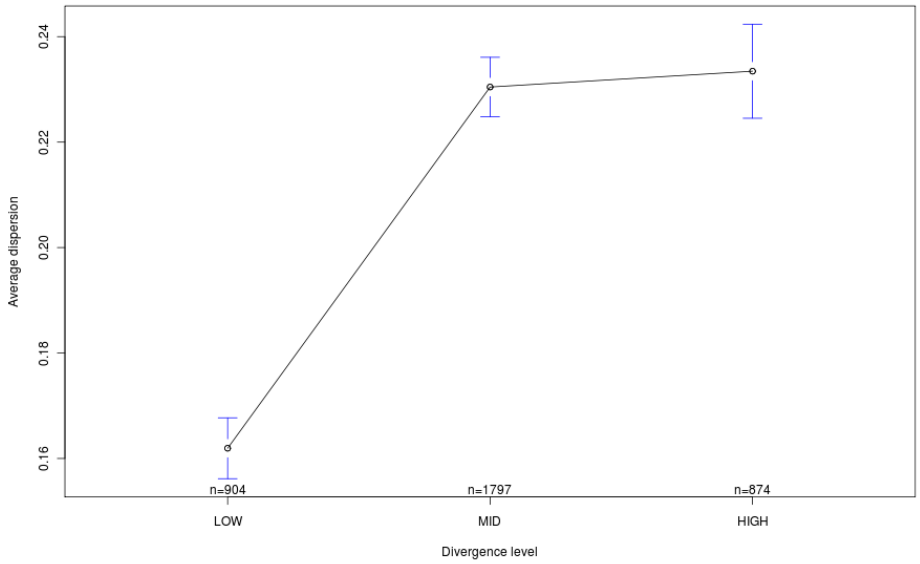


Figure 6.20: Average dispersion for different levels of divergence

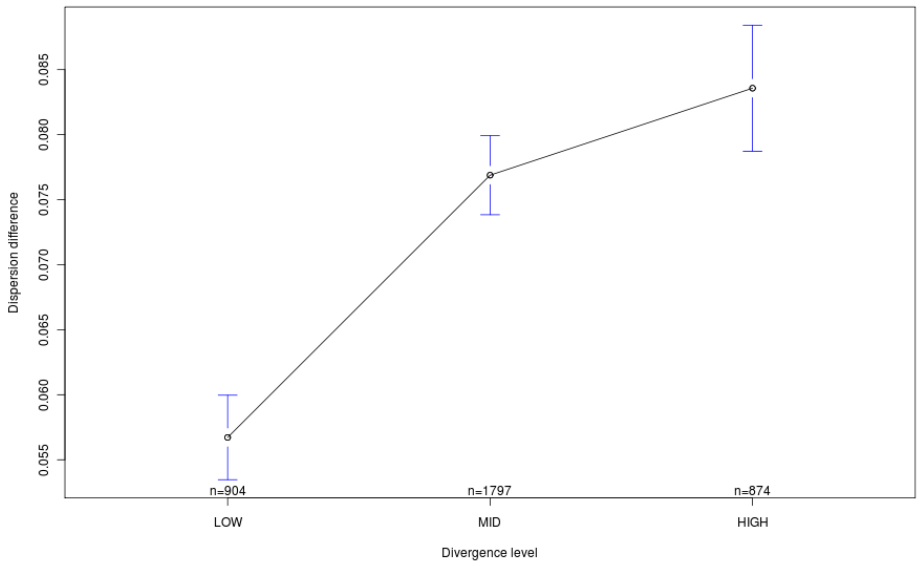


Figure 6.21: Dispersion difference between subjects for different level of divergence

we consider the results of section 6.4.1 which show that low-dispersion is associated with *flow* methods processing and considering that low-divergence is associated with low-dispersion, then it seems normal that low-divergence is also associated with *flow* methods processing.

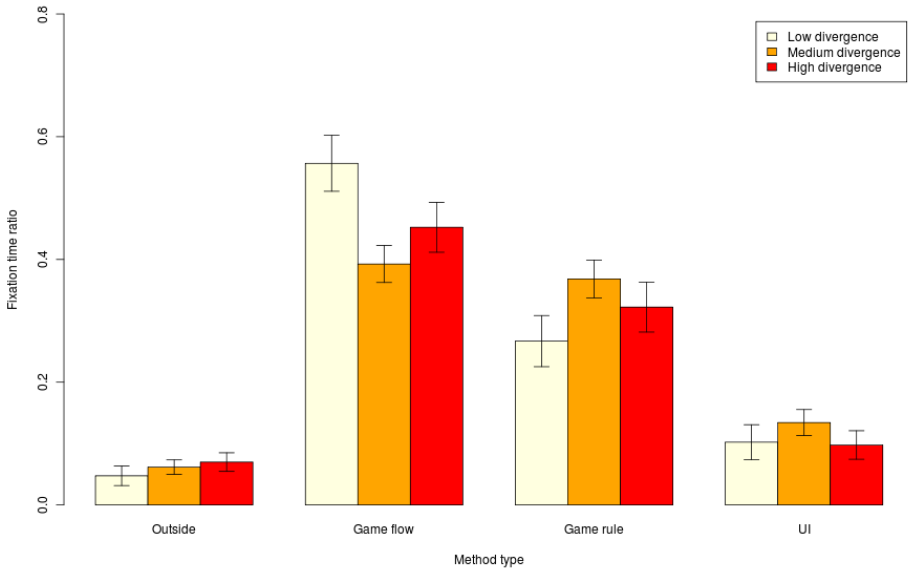


Figure 6.22: Fixation time ratio on method categories for different level of divergence in the *model building* question

For the *model updating* question (see figure 6.23), results are quite different and low-divergence episodes affect essentially fixation time on rule methods and fixation time outside methods. This is again consistent with the fact that similar effects were found for low-dispersion episodes (see section 6.4.1) and that low-divergence is associated with low-dispersion.

Collaboration quality

We analyzed the relationships between dual gaze features and collaboration ratings for the *model building* question. First, on figure 6.24, we can see the average divergence level according to the collaboration flow quality. This clearly shows that pairs with better collaboration flow had lower gaze divergence. A mixed-effect analysis confirmed this fact. Hence, a better collaboration flow implies less divergent gazes.

We also analyzed the relationship between the cross-recurrence, or more specifically the gaze coupling, and the quality of the collaboration flow (see figure 6.25). We can see that the gaze coupling level is lower for pair with a bad collaboration flow. We conducted a linear mixed-effect analysis which shows that this difference is significant. However, we should note that only pairs having a bad collaboration flow have a lower gaze coupling while pairs with medium or high collaboration flow have a similar level of coupling. We also looked at the gaze coupling for different levels of noise, i.e. in the presence of speech or not, by computing it only during

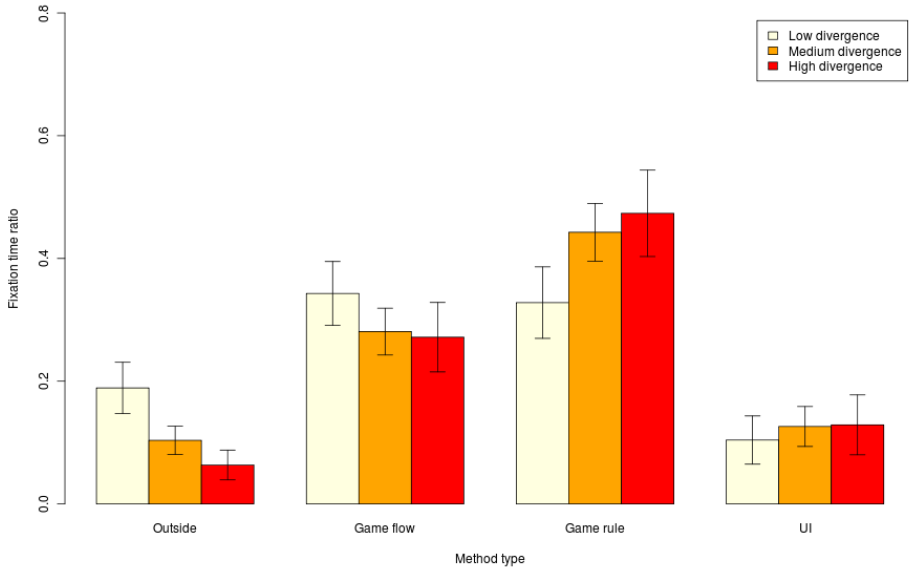


Figure 6.23: Fixation time ratio on method categories for different levels of divergence in the *model updating* question

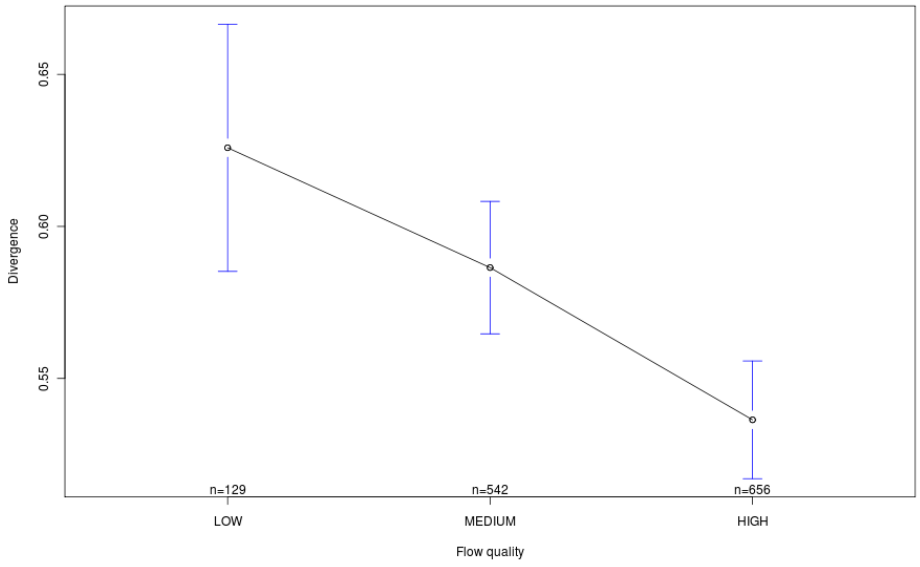


Figure 6.24: Divergence by collaboration flow quality

moments of noise and separately during moments of silence (see figure 6.26). This shows an important relationship with almost twice gaze coupling during speech. This is consistent with classical results about cross-recurrence which shows that dialogue produces gaze coupling. This could also explain why gaze coupling is not well correlated with the collaboration flow because there could bad collaborating pairs that speak more than well collaborating ones, thus having a higher recurrence level.

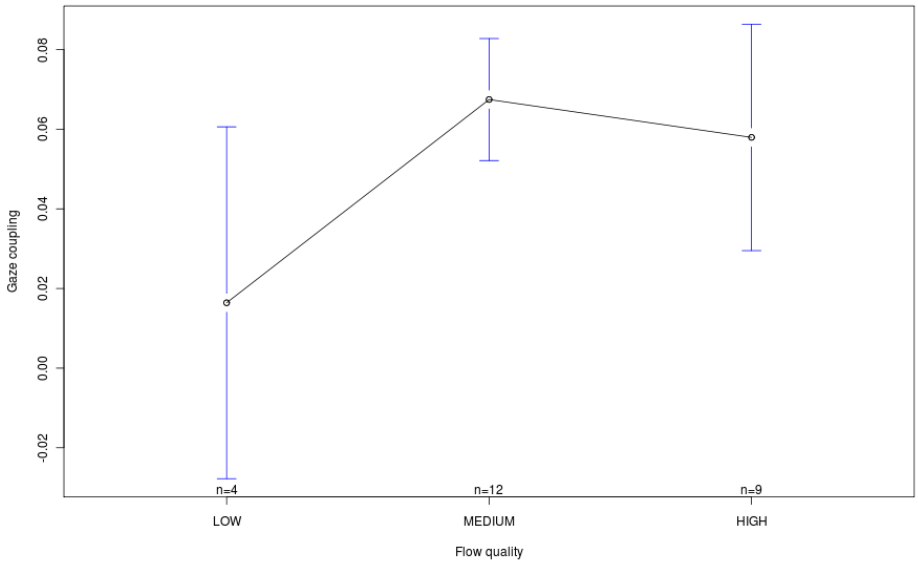


Figure 6.25: Gaze coupling by collaboration flow quality

Levels of understanding

We looked at the fixation time ratio on method types for different levels of answer correctness (see figure 6.27). We notice some interesting differences between the different levels of understanding. Moreover, these differences seem to vary linearly with the level of understanding. First, the amount of time on *UI* methods decreases with the level of understanding (- 2.6% by level) which is consistent with the other findings about expertise and with what we could expect as *UI* methods are the less important part of the code. Second, the time spent outside methods also decreases with the understanding (-3.7% by level). Again, this is expected as the important information outside methods is the fields declaration and it doesn't require much attention to be captured. Finally, the most interesting result concerns the amount of fixation time on *rule* and *flow* methods. Indeed, pairs who understood well have roughly the same amount of time between these two types of methods while pairs who understood badly spent much more time on *flow* methods than on *rule* methods. Either they were not able to understand the *flow* function and thus, could not really look at the *rule* functions, or they do not spent enough time on *rule* function to understand their semantic. In either case, it shows the importance of reading the *rule* methods in order to get a good understanding.

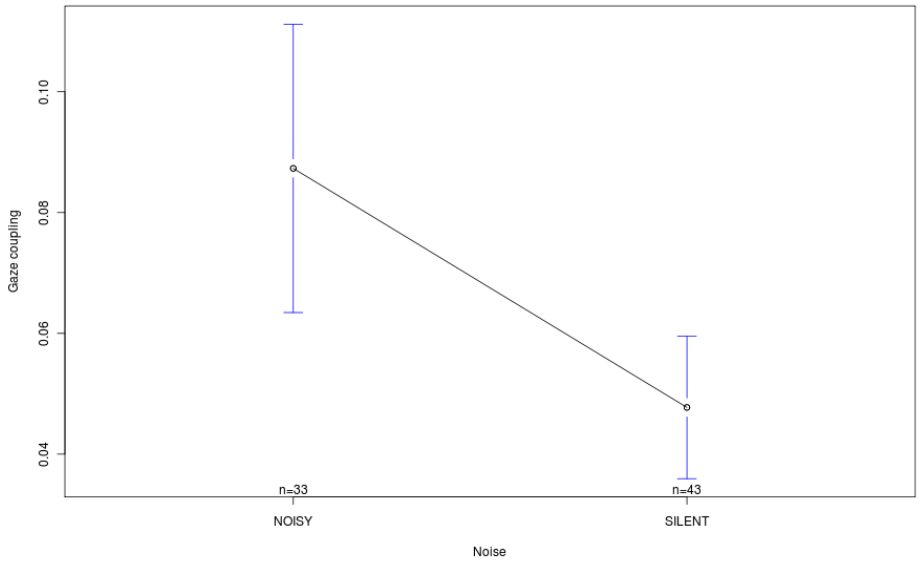


Figure 6.26: Gaze coupling by noise level

We should also note that the understanding level did not appear to be related to dual-gaze indicators, such as gaze coupling or gaze divergence. This can be explained as the best predictors for the understanding level is the expertise, while dual-gaze features predict more the collaboration flow quality which is not, in this case, a predictor of the understanding level. Indeed, we can have two novices who collaborate very well but who are not able to grasp the concepts of the game presented.

6.5 Discussion

We have presented the GREPP experiment which is a study about program reading and understanding both in solo and in duo. We aimed at studying eye-movements patterns during program understanding.

We first analyzed in section 6.4.1, individual eye-movements patterns during program reading. We identified several characteristics of eye-movements during code reading. First, we saw the existence of phases both in the problem solving strategy but also in the code exploration strategy. Indeed, it appears that programmers start by doing a general overview analysis of the code during which they look at shallow structural elements and then enter a semantic code analysis phase in which they analyze the deeper elements, essentially the expressions. On the code semantic side, we saw also that the type of task subjects were performing, building a mental model of the problem implemented, or updating their mental model of the program, produces very different gazes patterns on the different types of methods, thus reflecting their focus. Finally, we have seen that gaze dispersion reflected, up to some extent, the depth of the

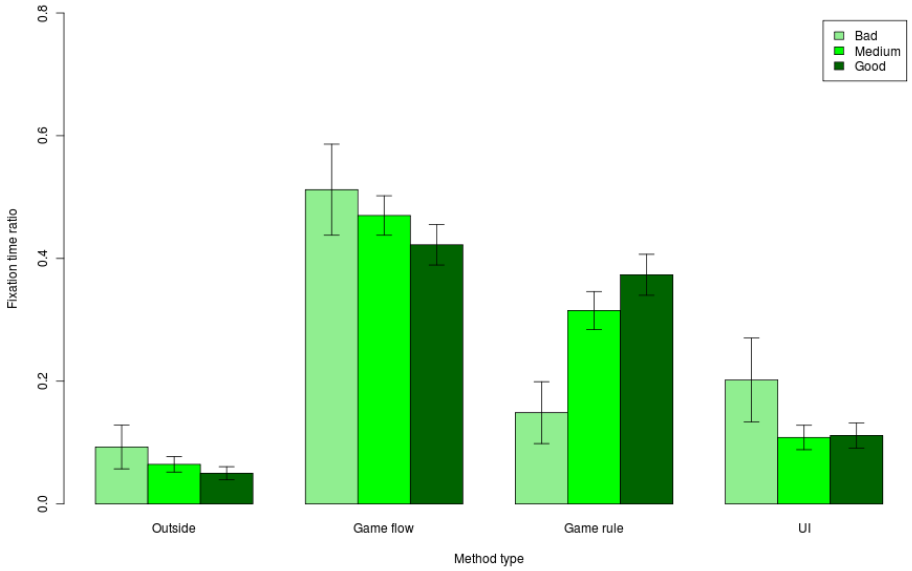


Figure 6.27: Fixation time ratio on method categories for different levels of understanding

processing done with low-dispersion being related to deep processing on complex specific parts, while high-dispersion was more related to general overview processing.

In a second step (see section 6.4.2), we looked at how these indicators varied with the levels of expertise of the subjects and with the collaboration. Interestingly, several results tend to show that collaboration has an effect on eye-movements which could be detrimental for the programmers with an effect sometimes opposed to expertise (e.g. semantic depth). Indeed, for example, programmers that collaborate look less at the semantically meaningful parts of the code, i.e. the expressions, and also less at the method of interest for the *model building* task. Interestingly, in the second main task (*model updating*), collaboration seems to affect novices negatively but experts positively. This could mean that the experts could get used to the collaboration and take advantage of it after becoming accustomed to it, while novices spend already too much effort on code comprehension to be able to benefit from collaboration. A general result is that collaboration does not affect eye-movements as much as expertise does.

Finally, in the last section 6.4.3, we analyzed first the relation between gaze divergence and the type of processing performed by the subjects. Several clues indicate that low gaze divergence, thus moments of joint attention, are related to deeper, complex processing (more time on expressions, lower gaze dispersion) while high-divergence is related to general structural processing. We have also seen that the overall gaze divergence level was strongly related to the quality of the collaboration flow. Similarly, the level of gaze coupling was also associated with the quality of the interaction flow, although in a lesser extent. However, we still found a strong relationship between the level of coupling and speech, even simply measured as the noise

volume. This is very interesting as it tends to confirm the validity of the approach used to measure the gaze coupling developed in section 4.3.4 as we expect this coupling to be higher during speech.

All in all, this study shows the usefulness of the multiple methodological developments done in this work. Indeed, thanks to these latter, we have been able to analyze at a very deep level dual gazes in a very complex visual stimulus. This could not have been done in a meaningful and reliable way without the corrections and methods developed in chapter 3 nor without the development related to dual eye-tracking methodology of chapter 4. In this respect, this study can be compared with previous eye-tracking studies about program reading such as Bednarik and Tukiainen (2006) or Bednarik (2011). Indeed, in these studies, they could not reach the same level of accuracy that we obtained. Hence, they are limited to general analysis such as the time spent in global zone (e.g code versus output). Unfortunately, it is not possible to compare their results with ours as the experimental design was different. Indeed, in their case, subjects had access not only to the code but also to the program output and to a graphical program visualization.

7 General discussion

The original motivation of this thesis was to explore the pertinence of dual eye-movement analysis to study collaborative activities. We were interested in characterizing, or predicting, the nature or the quality of the interaction between people collaborating remotely by analyzing their eye-movements. On the road to accomplish this task, we encountered several important methodological issues related to eye-tracking in general and to dual eye-tracking in particular. Hence, half of this work was dedicated to the analysis of these issues and the development of solutions. In this respect, we made several important findings and we developed useful tools and methods that could be used for eye-movement analyses.

The second part of this thesis was devoted to the analysis of experiments in which pairs of subjects had to solve tasks collaboratively on remote computers. We conducted four exploratory experiments on different types of tasks to show the validity of our approach. In a final step, we did a larger experiment which benefited from the experience gained in the previous studies as well as from all of the methodological developments that we accomplished in the first part. Instead of a depth first exploration in the field of collaborative gaze, we chose a breadth first approach to describe the variety of situations and problems that can be encountered. As a result of this endeavor, we got task-independent recommendations and methods that can be applied in the future to study. We did not discover universal relations between gaze patterns and effective collaboration, but rather showed that the task and the visual substrate for cognition is deeply influencing gaze.

7.1 Summary of main contributions

We summarize hereafter the main contributions made by this thesis. First, we present the various developments made on eye-tracking methods with, on the one hand, developments for the general methodology of eye-tracking and, on the other hand, developments related to the specific methodology of dual eye-tracking. Second, we show the contributions to the study of collaboration through dual gaze analysis with general results that illustrate the validity of such an approach and more specific results about the study of pair-programming.

7.1.1 Method

Through this work, we encountered several important issues concerning the analysis of the eye-tracking data. These problems appeared to be related to the various errors that occur when recording eye-tracking data and also, to some extent, to a gap existing in eye-tracking methodology. In chapter 3, we presented three main developments in relation to the three types of issues that appear when analyzing recorded gaze data. These developments include the following:

1. fixation identification, i.e. the detection of fixation in the raw gaze data provided by the eye-tracker
2. systematic location errors correction, i.e. the correction of systematic biases in the location of recorded data
3. hit detection, i.e. the assignation of fixations to the AOIs of the stimulus.

In chapter 4, we investigated the specific method related to dual eye-tracking analyses. We presented:

4. developments of technical aspects required for the synchronous recording of dual eye-movements
5. extension to complex collaborative situations of an existing analytical tool to measure gaze coupling between collaborators' eye-movements.

We summarize and discuss all these contributions hereafter.

Summary of results

Fixation identification is the process of detecting still phases in the raw gaze data measured by the eye-tracker. In section 3.2, we explored this question in detail. First, we showed that the choice of the parameter of the fixation detection algorithm affects the numbers of detected fixations, the average fixation duration, or the time covered with fixations. Moreover, we showed that this effect was different for each subject because the amplitude of variations of raw gaze positions during a fixation differs from one subject to another. This is explained in part by differences of subject's head distance to the eye-tracker, with a minimal dispersion for the optimal head-distance and a higher dispersion for closer or further distances. In a second step, we developed a *fixations set quality score* which assess the quality of a fixation identification given a stream of raw gaze. We used this score to run an optimization procedure of the parameter of the fixation identification algorithm to find the optimal parameter for each subject. We found that, on average, identification with optimal parameter reaches scores 50% better than with default parameters. We also showed that it is possible to estimate the optimal

threshold of a subject by simply computing the average of her raw gaze velocity. Finally, we developed an adaptive algorithm that does not require the setting of subject dependent parameters and instead estimates the noise level of the raw gaze data continuously to adapt its parameter accordingly. We optimized the algorithm by using the *fixations set quality score* and we ended up with an algorithm that reached the same performance level with a single set of parameters for all subjects as the classical algorithm with parameters adapted to each subject separately.

We discovered that the gaze data recorded by the eye-trackers was often affected by **systematic location errors**, i.e. the measured data was shifted in a specific direction compared to the actual gaze data. In section 3.3, we developed an automatic method to identify and correct the systematic errors in the gaze data positions, called gaze offset. This method relies on the fact that people look at the objects composing a stimulus most of the time rather than at the blank part. More specifically, we defined a function from the stimulus which affects to each point of the screen a fixation score that is high for points inside AOIs and low for points outside AOIs. We developed a brute-force optimization procedure using this fixation score which finds the offset so that the mean score of all fixations is maximized. The results unambiguously show the existence of an optimal gaze offset leading to an important improvement, in terms of number fixations inside AOIs, for many subjects. We also showed that there is a trend in the distribution of these systematic errors among the subjects with most subjects having an upward vertical offset. Finally, we also developed an analytical procedure to perform this correction, which proved to reach the same performance level, with less computational power.

We realized that the usual method for performing **hit detection**, i.e. the association of fixations to stimulus objects, was not adapted in the case of complex dense stimuli with numerous small AOIs. In section 3.4, we pointed out problems and biases that may appear when using the simple, closest object, method. Instead, we proposed a novel probabilistic method which aims to overcome these issues.

Dual eye-tracking involves an important technical issue, which is the synchronous recording of gaze data of two people. The difficulty lies in the use of two eye-trackers such that the gaze data recorded by both machines become comparable temporally. In section 4.2, we proposed four options to reach this goal. The first one consists of separately recording each stream of gaze and performing post-synchronization afterwards by making correspondences between events synchronized with gaze data. The second one consists of starting the recording of both eye-tracker synchronously thanks to the use of the programming API provided by the manufacturer. The third option consists of bypassing the use of the default software used to record gaze and instead using a custom software which takes care of the logging of both eye-trackers' data through the use of the low-level programming API. The fourth and last one is similar to the third one except that it makes both eye-trackers use the same timer located on a third computer so that the timestamps on the gaze data are directly comparable. This latter solution appeared to be the best we can achieve considering the hardware we were using.

The second issue of dual eye-tracking that we explored was the use of **cross-recurrence analysis** to study eye-movements during complex collaborative task solving. In section 4.3, we showed that cross-recurrence analysis, which aims to measure the gaze coupling between two individuals speaking to each other, is likely to be biased by other phenomena when used in complex tasks settings. In particular, the structure of the interaction (how much they collaborate versus cooperate or whether subjects explore the stimulus by zone) may have a big influence on the coupling measured with cross-recurrence analysis. We proposed a solution to overcome these problems, which consists of computing a zone-based cross-recurrence. The principle is to compute the cross-recurrence on each zone of the stimulus that is looked at more or less uniformly and to correct the computed cross-recurrence rate with the number of objects contained in the zone to take into account different random levels. We performed a simulation by generating fake data and we showed the effectiveness of our method compared to the normal simple cross-recurrence on this fake data.

Discussion of contributions

Through these analyses, we have identified and quantified problems that exist with the recording of eye-tracking data using the Tobii 1750. In particular, we have shown that contrary to what is claimed by the manufacturer, **subjects could not freely move their head**. Indeed, if they do so, these head movements could affect the noise of the recorded data and, in turn, subsequent analyses done on the gaze data. We do not know the reason for this phenomenon and we cannot say whether it could affect other eye-trackers or not. However, one possibility is that it is an effect of the camera lens, i.e. that changes of head distance affect the quality of the focus and in turn, decrease the precision of the eye-tracker measures. If this is the reason, it is likely that other eye-trackers of the same type would be affected by similar problems. A second problem that we identified is the **presence of systematic biases in the accuracy of the recorded gaze**. We have indeed shown that most of the time, the recorded gazes are shifted compared to their real locations. Moreover, we have seen that these shifts vary from one subject to another but that there is a general upward tendency. Again, we cannot tell what the causes of these systematic errors are and thus, whether this could affect other eye-trackers as well. However, we know that it is not due to variations of head position as we controlled for this in one experiment and the problem persisted. Also, other researchers using different apparatuses noticed to have found similar problems in their data, which tends to show that this is not specific to the eye-tracker used in this study. The quantification of these issues are important contributions both for applied eye-tracking research, as they unambiguously points out the existence of problems that may bias results of analyses, and also for the development of eye-tracking techniques as it highlights issues that may arise with video-based eye-trackers.

We have also contributed to the analysis methodology by highlighting problems and difficulties that may arise when analyzing gaze data. In particular, we have shown that **the choice of the parameters to obtain a correct fixation identification is dependent on the subject**. These analyses complement earlier works existing in the literature, such as the work of Karsh and

Breitenbach (1983) and Shic et al. (2008a) which have already shown the great importance of choosing the correct parameters of the algorithm or of Blignaut and Beelders (2009), who have shown the necessity of choosing subject specific parameters. We even made an additional step by automatically quantifying the result of fixation identification with the *fixations set quality score*. While this score would certainly require improvements to completely fulfill its goal, it is already an important contribution as it opens interesting alleys for the development of fixation identification algorithms. We have also pointed out **biases that exist in the process of hit detection** as it is usually done, i.e. by choosing the closest object as being the hit. We have shown that this simple method of uniquely associating each fixation to a single object does not account for the presence of unsystematic location errors and that it is not adapted to situations with many small AOIs.

We have also offered solutions that overcome these issues to some extent. First, by **developing an adaptive algorithm for fixation identification**, we have been able to cope with differences of dispersion that exist in the recorded gaze data between the subjects. The adaptive algorithm we have developed is similar to the work of Nyström and Holmqvist (2010). However, our algorithm brings interesting contributions for two reasons. First, because its fine tuning was done through the mean of an objective automatic quality assessment and second, because it works in real-time and does not require iteration. Second, we have developed a **general procedure to correct automatically systematic errors** present in the recorded gaze positions. This algorithm is comparable to the work of Zhang and Hornof (2011) but uses a different approach. Results show that both of these developments reach their goal and are actually able to diminish the impact of the issues they aim to correct. Moreover, these algorithms have been designed with the idea that they must be usable for real-time applications. Indeed, the analytical method we have developed for the correction of systematic errors requires little computational power to be used in real-time. This is an important point as it could allow people to build gaze-contingent applications that would benefit from these developments. In particular, one of the long-term goals of the developments done in this thesis is the construction of collaborative gaze aware applications to enhance computer-supported collaboration and in this respect, the fact that our solutions can work in real-time is very important. Finally, we have proposed a **probabilistic hit detection** method which takes into account the presence of unsystematic errors in the measured gaze locations and is better adapted to situations with numerous small AOIs. To our knowledge, this is the first attempt in this direction to cope with this procedure.

Finally, we have contributed to the method of dual eye-tracking by **offering technical solutions to synchronously record the gaze of two collaborators** with a high precision level. This is a contribution as it shows the possibility of performing analysis of eye-movements during interaction between two people. We have also **extended cross-recurrence analysis to the context of real complex collaborative problem solving**. The zone-based cross-recurrence analysis that we proposed, however, does have some limitations. Indeed, its effectiveness has only been shown with fake data which has been generated with a simplifying assumption. We considered that the subjects looked at objects in a zone uniformly, i.e. that they had the same

probability of looking at any object of a zone at any time. This assumption is certainly not true and it is more likely that collaborators look at some objects more often and with different probabilities at different time. Hence, it is likely that, while our method is theoretically valid, it is only partially in the case of real data.

7.1.2 Collaboration and eye-movements

The original focus of this work was the analysis of dual eye-movements in order to gain insight about collaborative processes. In this respect, we have been able to highlight several results through exploratory experiments presented in chapter 5. We found statistical relationships between eye-movements of collaborators and:

1. the quality of the collaboration product, i.e. the performance of the collaborators at the task
2. the level of expertise of the collaborators
3. some specific interaction episodes, i.e. episodes during which collaborators are engaged in a specific type of interaction
4. the specific type of verbal interaction constituted by explicit references, i.e. when collaborators refer to an object

In chapter 6, we presented a larger study aimed at studying specifically pair-programming from which we have found that:

5. eye-movements are less affected by collaboration as by expertise
6. collaboration seems to have a cost and affects eye-movements in a manner that could be detrimental for the subjects
7. subjects tend to collaborate more on complex specific aspects than on general structural aspects of the task
8. good collaboration is generally correlated with a higher level of similarity between eye-movements of collaborators

Summary of results

At a very general level, we have investigated the **relationship between eye-movements and the outcome of the collaboration** in two tasks. In section 5.3, we have shown that there was some correlation between the performances of the collaborators in some problem solving activity and indicators of eye-movements. In one case, the Shoutspace experiment, we have found a significant correlation between the level of cross-recurrence and the performance

score at the task. In the second case, the RavenBongard, we have been able to train machine learning algorithms to make predictions of success level with a reasonable level of accuracy.

In section 5.4, we have also seen that it could be possible to **predict the levels of collaborators' expertise from the gaze patterns**. In two different experiments, the KAT and Tetris experiments, we have trained machine learning algorithms to predict the level of expertise of the subjects. In one situation, we have also been able to predict the pair-composition, i.e. the level of one subject as well as the level of her partner.

On a lower-level, in section 5.5, we have illustrated how the content of **short interaction episodes was reflected to some extent by the gaze patterns**. In the Tetris experiment, we have shown that eye-movements were affected by conflict episodes which require a higher level of coordination between players. In the KAT experiment, we have seen that the type of dialogue between collaborators was reflected by two dual-gaze indicators, the mutual dispersion and the gaze divergence.

In section 5.6, we have replicated two important results about the **link between speech and gaze** in a complex collaborative setting. First, we have shown the existence of an eye-speech span during explicit references, i.e a time delay between fixations on the referred object and production of the verbal reference, and of a speech-eye span, i.e. a time delay between the listening of an explicit reference and the fixations of the referred object. We have also shown the existence of similar phenomena in the case of written communication, by showing how readers and writers look at the referred objects when reading or writing messages containing references. Second, we have found a cross-recurrence peak for a lag of 2 seconds by analyzing the gazes of the collaborators during explicit references. Based on these findings, we have also developed a prototype of an algorithm, REGARD, which is able to automatically detect verbal references to visual objects through the analysis of the speech and gaze streams of collaborators.

In section 6.4, we have seen that in the GREPP experiment, **collaboration does not affect eye-movement patterns as much as expertise** does. In other words, it seems that collaboration does not profoundly affect the way of looking at the stimulus, while expertise has a much greater influence. We have also shown that, in some cases, **collaboration seems to affect eye-movements in a negative way** in the sense that it tends to make collaborators look more like novices. We have found that **during episodes of collaboration, subjects tend to focus more on the complex, specific parts of the task** as opposed to making a general overview processing during more individual phases. Finally, we have shown that a **good collaboration generally means more similarity between collaborators' eye-movements**.

Discussion of contributions

One main contribution is the demonstration of **the validity of the dual eye-tracking approach to study collaboration**. Indeed, through these various results, we have been able to show that

relationships between eye-movements and collaborative processes exist. However, except for the prediction of expertise levels, most of our results are limited by the fact that they are only statistical relationships which by no means allows for predictions. Hence, we have shown the pertinence of using dual eye-tracking to study collaboration, but at the same time, we have seen its limitation for drawing strong, reliable predictions. Rather, it appears that **the most promising line of research is the study of dual eye-movements combined with other modalities such as actions or speech**. Another limitation of our results is the specificity of the task. In almost all cases, interesting results required eye-movement analysis specific to the task under concern.

One contribution of these analyses was to **delimit the sets of the tasks** that are interesting to be studied with this methodology. Indeed, we have identified two dimensions which describe our different experiments. The first dimension is the type of cognitive activities required by the task, ranging from purely conceptual tasks (such as the KAT experiment, i.e. collaborative learning from concept-mapping) to sensori-motor coordination tasks (the Tetris experiment). The second dimension concerns the type of visual content which ranges from visual content fully present since the beginning (e.g. the RavenBongard experiment) to visual content progressively built (e.g. the KAT experiment). This allowed us to outline a zone of interest for dual eye-tracking studies in this two-dimensional space of collaborative tasks. This zone is located in the middle of the first dimension, i.e. for tasks that are not too conceptual so that eye-movements have more opportunities to reflect the cognitive activities, but which still involve important conceptual aspects so that they are interesting enough to study cognitive aspects. Concerning the second dimension, the tasks of interest are situated on the extrema which include tasks with visual content present from the beginning.

Concerning the study collaboration, our main contribution was to highlight the main relationships between eye-movements and interaction. One of the main findings is that **collaboration affects eye-movement mainly through speech** and more specifically through explicit references. Indeed, the most promising results concern the analysis of gazes during references but also through the analysis of cross-recurrence, which is mainly due to explicit references. In this respect, we replicated and extended previous works, such as Richardson and Dale (2005) and Griffin and Bock (2000) who showed the close connection between gaze and speech. First, we replicated results about eye-speech span and cross-recurrence in a more ecologically valid context, i.e. in complex collaborative problem solving activities. Second, we extended these results by showing their pertinence in the context of textual communication. Moreover, this relation between gaze and speech appeared to be strong and reliable, allowing us to **develop a gaze-aware system to automatically detect references to objects**. Although it is not a fully functional system per se, as it still requires having access to a textual stream of the pronounced words with precise timing, it proved to be very effective. Thus, it shows the effectiveness of our approach for the development of gaze-aware tools that could help remote collaborative activities.

Finally, another finding is that **collaboration does not seem to affect eye-movements beyond the effect due to language**. More specifically, it does not seem to significantly affect the patterns of eye-movements. These patterns are instead much more affected by the subjects' expertise of the domain. Indeed, in terms of predictability, the strongest results concern the prediction of levels of expertise. Although it was still possible to find relationships with the performance in two tasks, we did not reach the same accuracy of predictions as for levels of expertise. Moreover, in the GREPP experiment, when we compared effects of collaboration on eye-movements versus effect of expertise, the former appeared to be negligible compared to the latter.

7.2 Implications and future works

We discuss hereafter the implications of our contributions for the research on applied eye-tracking in general and for research on collaboration.

7.2.1 Applied eye-tracking research

On the methodological side, our results have important implications for applied eye-tracking research. We have indeed shown that it is not acceptable to consider the eye-tracker and its analysis software as a black-box without taking care of the quality of the data it produces. Indeed, this would first result in very noisy analyses (because of the problems of variable noise and its effect on fixation identification) and second this would bring wrong and biased results (because of the systematic location errors and especially of the systematic upward trends in these errors). Each measuring instrument comes with an error, and better understanding the extent of the error allows researchers to better adapt tools to the research questions at hand.

This also opens several new lines of research. First, many improvements can still be made in the fixation identification algorithm. In particular, the *fixations set quality score* could be improved to better reflect the quality of the identified fixations. We could, for example, use better indicators such as the vectorial speed instead of the simple dispersion measure that we used. This score should also be validated, for example, by conducting a brute force optimization that would find the optimal fixation identification according to this score for a set of raw gaze and then qualitatively analyzing the result. This would permit a better optimization of the parameters of the adaptive algorithm. Indeed, we saw that currently no unique optimum existed for the parameters of the algorithm. So, it is likely that with a better score, we could better identify the optimal parameters. The algorithm should then be tested with other eye-tracking data of different frequencies, different units (visual angles instead of pixels), and more generally from different hardware. This would permit us to test the generalization of the algorithm.

Also, concerning the correction of systematic errors, several improvements have to be made. First, the correction methods should be more developed to take into account the variation

of offsets through time and or space. In this respect, the analytical method offers good perspectives because it can easily be generalized to more complex models in which the offset is not fixed but instead is represented by some function of time and space. The difficulty resides in the fact that, while it appears that this offset actually depends on space and possibly also on time, we do not know the form of this dependence. In particular, qualitative analyses revealed that this dependence is certainly not linear. At some point, if we allow very complex dependence, we risk "overfitting" the correction and "forcing" the data to the stimulus up to the point that they do not reflect the reality anymore. The second improvement that could be made is the generalization to situations for which we do not know the stimulus and for situations with complex stimuli such as natural images. Indeed, the current methods require a detailed knowledge of the stimulus. Hence, it would be of great interest to be able to perform the same kind of correction without this knowledge. A possible direction for this would come from image analysis. For example, we could imagine some way to detect zones in an image that are more likely to contain gazes, such as saliency maps. Finally, it would be interesting to compare the methods we developed with the method developed by Zhang and Hornof (2011).

Finally, a third direction that would require more exploration is the effect of these different pre-processing steps on subsequent statistical analyses. It would be interesting to systematically study the effect of each step on some statistical results in order to show to which extent they could be affected. In this way, we could get a quantitative account of the impact of this processing for applied eye-tracking research. Indeed, while it is qualitatively clear that, in some situations, uncorrected data (typically without systematic errors correction) is completely wrong and would result in wrong analyses, it is difficult to assess the overall effect of not applying these corrections.

7.2.2 Collaboration analysis

On the analysis side, our work shows the pertinence of the dual eye-tracking approach for the study of collaboration. We have shown that this could bring insights about the cognitive processes that are in play during collaboration. Hence, this is a valid novel methodology that is worth being taken into account for the study of collaboration. It is even truer today as eye-trackers are being developed more and more and are becoming more easily accessible to researchers outside of the main field of eye-tracking.

In particular, it appears that dual eye-movement research is likely to provide fruitful results when combined with speech analysis. Indeed, eye-movements could inform about the type and quality of speech. More specifically, eye-movements are likely to provide information about the level of understanding between collaborators so more research should be conducted in this direction. This opens interesting perspectives for the development of gaze-aware collaborative in domains where understanding between collaborators is crucial (e.g. air controllers).

For the analysis of dual eye-movements, cross-recurrence appears to be a promising and valid

method which can offer very interesting results. In this respect, more work is required in order to use it in a meaningful way in the context of complex collaborative activity. Indeed, the correction method we proposed (zone-based cross-recurrence) appears to be insufficient for fully distinguishing the short-term gaze coupling due to dialogue from other effects due to the general structure of interaction. Hence, additional work should be done in this direction to find a way to extract the actual short-term coupling when analyzing cross-recurrence during complex collaborative problem solving.

This also opens perspectives for the development of gaze-aware collaborative softwares that would take advantage of dual eye-movement patterns to provide help to collaborators. In this respect, the REGARD model is a good example of such a tool. However, it would still require additional developments to become an autonomous system that can be used in more general situations. Also, other systems of the same kind could be developed. For example, we can imagine misunderstanding detectors that would analyze the gaze matching of the collaborators during referencing to make inferences about potential misunderstandings. However, our results tend to imply that general gaze-aware collaborative systems have to be limited to the analysis of explicit references. Indeed, systems that make other types of predictions (e.g. about the performance or the expertise) should certainly be designed specifically for a given task and could difficulty be general enough to work in multiple domains. Moreover, the field of application of such systems is also limited to certain types of task.

7.3 Final words

This thesis was done with the goal of gaining more insights about the cognitive processes that happen during collaboration. In this respect, we decided to use a novel method, dual eye-tracking, in order to have closer access to processes happening in the mind. This task revealed to be far more complex than it appeared to be at beginning. Indeed, in addition to the developments required to apply such a new methodology, we faced several important problems that we could not ignore concerning the general methodology of eye-movement data analysis. Hence, we tackled these issues at the same time as our main goal. This leaded us to identify and find solutions to several important methodological issues and we managed to cope with most of them in a very acceptable way. To this end, we made several important contributions that may be of great help for the applied eye-tracking research community. Indeed, on the one hand, we have pointed out and quantified several problems that exist which have received little attention from researchers and on the other hand, we have offered solutions, which, even if they are not perfect, are a good step towards more reliable and stronger eye-tracking analyses.

Concerning our main goal, the analysis of collaboration, results are less fruitful. Indeed, we have not been able to find strong relationships between dual eye-movements and collaboration. Nevertheless, we have still highlighted different statistical results in several tasks concerning different aspects of collaboration. However, we were not able to find a strong unify-

ing result that goes beyond a single task. There is, however, one exception which is the strong link between speech and gaze through referencing. This appears to be general enough to open interesting perspectives. One reason for the partial failure of our enterprise may come from the fact that we decided to analyze realistic complex situations in which many phenomena are intricate. In any case, we have been able to show the effectiveness of our approach and, in this respect, we have made small but valuable steps for studies of collaboration.

A Appendix

Listing A.1: First version: Model building task

```
import java.io.BufferedInputStream;
import java.util.LinkedList;
import java.util.Scanner;

public class AddThemUp {

    LinkedList<Integer> availableNumbers = new LinkedList<Integer>();
    LinkedList<Integer>[] playersNumbers = new LinkedList[2];

    public void run() {
        initGame();
        int currentPlayer = 1;
        showGameStarted();
        int winner = 0;
        while ((winner = checkForWinner()) == 0 && !gameFinished()) {
            showCurrentGameState();
            int choice = getPlayerMove(currentPlayer);
            while (!checkAndSet(choice, currentPlayer)) {
                showBadMove();
                choice = getNewPlayerMove(currentPlayer);
            }
            currentPlayer = 2 - currentPlayer + 1;
        }
        showCurrentGameState();
        if (winner != 0)
            showWinner(winner);
        else
            showGameDraw();
        showGameEnded();
    }


    private void initGame() {
        playersNumbers[0] = new LinkedList<Integer>();
        playersNumbers[1] = new LinkedList<Integer>();
        for (int i = 1; i < 10; i++) {
            availableNumbers.add(i);
        }
```

```
39      }
40
41
42      private boolean checkAndSet(int choice, int player) {
43          if (!availableNumbers.contains(choice))
44              return false;
45          playersNumbers[player - 1].add(choice);
46          availableNumbers.removeFirstOccurrence(choice);
47          return true;
48      }
49
50
51      private boolean checkForSum(int p, int i1, int i2, int i3) {
52          return (playersNumbers[p].get(i1) + playersNumbers[p].get(i2) +
              playersNumbers[p].get(i3) == 15);
53      }
54
55
56      private int checkForWinner() {
57          for (int p = 0; p < 2; p++) {
58              for (int i1 = 0; i1 < playersNumbers[p].size(); i1++) {
59                  for (int i2 = i1 + 1; i2 < playersNumbers[p].size(); i2++) {
60                      for (int i3 = i2 + 1; i3 < playersNumbers[p].size(); i3++)
                            {
61                          if (checkForSum(p, i1, i2, i3))
62                              return p + 1;
63                      }
64                  }
65              }
66          }
67          return 0;
68      }
69
70
71      private boolean gameFinished() {
72          return availableNumbers.size() == 0;
73      }
74
75
76      public static void main(String[] args) {
77          new AddThemUp().run();
78      }
79
80      /** Following functions implement the user interface for input/outputs */
81
82      // useful stuff to analyze input from console
83      private Scanner scanner = new Scanner(new BufferedInputStream(System.in),
          "UTF-8");
84
85
86      public void showBadMove() {
87          System.out.println("Bad or already taken number!");
88      }
89
90
91      public void showCurrentGameState() {
92          System.out.println();
93          System.out.print("Numbers left:");
```

```
94          for (Integer v : availableNumbers) {
95              System.out.print(" " + v);
96          }
97          System.out.println();
98          for (int p = 0; p < 2; p++) {
99              System.out.print("Player " + (p + 1) + " numbers:");
100             for (Integer v : playersNumbers[p]) {
101                 System.out.print(" " + v);
102             }
103             System.out.println();
104         }
105         System.out.println();
106     }
107
108
109     public void showGameDraw() {
110         System.out.println("Draw! No one wins...");
111
112     }
113
114
115     public void showGameEnded() {
116         System.out.println("Game is finished...");
117     }
118
119
120     public void showGameStarted() {
121         System.out.println("Game has started...");
122     }
123
124
125     public int getNewPlayerMove(int p) {
126         System.out.println("Player " + p + " chooses a number again:");
127         return scanner.nextInt();
128     }
129
130
131     public int getPlayerMove(int p) {
132         System.out.println("Player " + p + " chooses a number:");
133         return scanner.nextInt();
134     }
135
136
137     public void showWinner(int p) {
138         System.out.println("Player " + p + " wins!");
139     }
140 }
```

Listing A.2: Second version: Model updating task

```
import java.io.BufferedInputStream;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

public class AddThemUp_EW {

    int[] ownerships = new int[9];


    public void run() {
        initGame();
        int currentPlayer = 1;
        showGameStarted();
        int winner = 0;
        while ((winner = checkForWinner()) == 0 && !gameFinished()) {
            showCurrentGameState();
            int choice = getPlayerMove(currentPlayer);
            while (!checkAndSet(choice, currentPlayer)) {
                showBadMove();
                choice = getNewPlayerMove(currentPlayer);
            }
            currentPlayer = 2 - currentPlayer + 1;
        }
        showCurrentGameState();
        if (winner != 0)
            showWinner(winner);
        else
            showGameDraw();
        showGameEnded();
    }


    private void initGame() {
        for (int i = 1; i < 10; i++) {
            ownerships[i - 1] = 0;
        }
    }


    private boolean checkAndSet(int choice, int player) {
        if (ownerships[choice - 1] != 0)
            return false;
        ownerships[choice - 1] = player;
        return true;
    }


    private int checkForSamePlayer(int v1, int v2, int v3) {
        int player = ownerships[v1 - 1];
        if (ownerships[v2 - 1] == player && ownerships[v3 - 1] == player)
            return player;
        return 0;
    }


```

```
57      private int checkForWinner() {
58          for (int v1 = 1; v1 < 10; v1++) {
59              for (int v2 = v1 + 1; v2 < 10; v2++) {
60                  int v3 = 15 - v1 - v2;
61                  if (v3 > v2 && v3 < 10) {
62                      int p = checkForSamePlayer(v1, v2, v3);
63                      if (p != 0)
64                          return p;
65                  }
66              }
67          }
68          return 0;
69      }
70
71
72      private boolean gameFinished() {
73          for (int i = 1; i < 10; i++) {
74              if (ownerships[i - 1] == 0)
75                  return false;
76          }
77          return true;
78      }
79
80
81      public static void main(String[] args) {
82          new AddThemUp_EW().run();
83      }
84
85      /** Following functions implement the user interface for input/outputs */
86
87      // useful stuff to analyze input from console
88      private Scanner scanner = new Scanner(new BufferedInputStream(System.in),
          "UTF-8");
89
90
91      public void showBadMove() {
92          System.out.println("Bad or already taken number!");
93      }
94
95
96      public void showCurrentGameState() {
97          LinkedList<Integer>[] numbersLists = new LinkedList[3];
98          for (int i = 0; i < 3; i++)
99              numbersLists[i] = new LinkedList<Integer>();
100
101         for (int i = 1; i < 10; i++) {
102             numbersLists[ownerships[i - 1]].add(i);
103         }
104
105         System.out.println();
106         System.out.print("Numbers left:");
107         for (Integer v : numbersLists[0]) {
108             System.out.print(" " + v);
109         }
110         System.out.println();
111         for (int p = 0; p < 2; p++) {
112             System.out.print("Player " + (p + 1) + " numbers:");
113             for (Integer v : numbersLists[p + 1]) {
```

```
114                     System.out.print(" " + v);
115                 }
116                 System.out.println();
117             }
118             System.out.println();
119         }
120 
121 
122         public void showGameDraw() {
123             System.out.println("Draw! No one wins...");
124 
125         }
126 
127 
128         public void showGameEnded() {
129             System.out.println("Game is finished...");
130         }
131 
132 
133         public void showGameStarted() {
134             System.out.println("Game has started...");
135         }
136 
137 
138         public int getNewPlayerMove(int p) {
139             System.out.println("Player " + p + " chooses a number again:");
140             return scanner.nextInt();
141         }
142 
143 
144         public int getPlayerMove(int p) {
145             System.out.println("Player " + p + " chooses a number:");
146             return scanner.nextInt();
147         }
148 
149 
150         public void showWinner(int p) {
151             System.out.println("Player " + p + " wins!");
152 
153         }
154 
155     }
```

Bibliography

P. D. Allopenna, J. S. Magnuson, and M. K. Tanenhaus. Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models, , , ,. *Journal of Memory and Language*, 38(4):419 – 439, 1998. ISSN 0749-596X. doi: DOI:10.1006/jmla.1997.2558. URL http://www.sciencedirect.com/science/article/pii/S0749596X97925584.

R. Bednarik. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies*, (0):–, 2011. ISSN 1071-5819. doi: 10.1016/j.ijhcs.2011.09.003. URL http://www.sciencedirect.com/science/article/pii/S1071581911001388.

R. Bednarik and M. Tukiainen. An eye-tracking methodology for characterizing program comprehension processes. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, ETRA '06, pages 125–132, New York, NY, USA, 2006. ACM. ISBN 1-59593-305-0. doi: http://doi.acm.org/10.1145/1117309.1117356. URL http://doi.acm.org/10.1145/1117309.1117356.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

P. Blignaut. Fixation identification: the optimum threshold for a dispersion algorithm. *Attention, Perception & Psychophysics*, 71(4):881–895, May 2009. doi: 10.3758/APP.71.4.881. URL http://dx.doi.org/10.3758/APP.71.4.881.

P. Blignaut and T. Beelders. The effect of fixational eye movements on fixation identification with a dispersion-based fixation detection algorithm. *Journal of Eye Movement Research*, 2 (5):1–14, 2009.

S. E. Brennan, X. Chen, C. A. Dickinson, M. B. Neider, and G. J. Zelinsky. Coordinating cognition: The costs and benefits of shared gaze during collaborative search. *Cognition*, 106(3):1465 – 1477, 2008. ISSN 0010-0277. doi: DOI:10.1016/j.cognition.2007.05.012. URL http://www.sciencedirect.com/science/article/B6T24-4P4FSS9-1/2/5ccdbc579c3c8e54998ace40338e3d7d.

J. C. Brown, A. Collins, and D. Newman. Situated cognition and the culture of learning. *Educational Researcher*, 18(1):32–42, 1989. URL http://www.educ.msu.edu/DWongLibrary/CEP900/Library/BrownCollinsDuguid1989.pdf.

G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio. Evaluating performances of pair designing in industry. *J. Syst. Softw.*, 80:1317–1327, August 2007. ISSN 0164-1212. doi: 10.1016/j.jss.2006.11.004. URL http://dl.acm.org/citation.cfm?id=1244477.1244776.

P. A. Carpenter, M. A. Just, and P. Shell. What one intelligence test measures: a theoretical account of the processing in the raven progressive matrices test. *Psychological review*, 97(3): 404–31, July 1990.

M. Cherubini. *Annotations of maps in collaborative work at a distance*. PhD thesis, Lausanne, 2008. URL http://library.epfl.ch/theses/?nr=4116,http://library.epfl.ch/theses/?nr=4116.

M. Cherubini and P. Dillenbourg. The effects of explicit referencing in distance problem solving over shared maps. In *Proceedings of the 2007 international ACM conference on Supporting group work*, GROUP '07, pages 331–340, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-845-9. doi: http://doi.acm.org/10.1145/1316624.1316674. URL http://doi.acm.org/10.1145/1316624.1316674.

M. Cherubini, M.-A. Nüssli, and P. Dillenbourg. Deixis and gaze in collaborative work at a distance: a computational model to detect misunderstandings. In *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research and applications*, pages 173–180, New York, NY, USA, 2008. ACM.

M. Cherubini, M.-A. Nüssli, and P. Dillenbourg. This is it ! : Indicating and looking in collaborative work at distance. *Journal of Eye-Movement Research*, 3(5):1–20, 2010. ISSN 1995-8692. doi: NA. URL http://www.jemr.org/online/3/5.

A. Clark. *Being There: Putting Brain, Body, and World Together Again*. The MIT Press, Jan. 1998. ISBN 0262531569. URL http://www.worldcat.org/isbn/0262531569.

H. H. Clark and D. Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22(1):1–39, Feb 1986.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 1999.

A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd. edition, 2007. ISBN 1852336668.

J. P. Eckmann, S. O. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *EPL (Europhys. Lett.)*, 4:973–977, 1987.

M. Frank, E. Vul, and R. Saxe. Measuring the development of social attention using free-viewing. *submitted*, 2011. URL http://langcog.stanford.edu/papers/FVS-infancy2011.pdf.

M. C. Frank and E. Vul. Being sure you know where participants are looking: Offline calibration checking. In *Special symposium Tracking Infants' Eye Gaze Patterns Using the Tobii System at the International Conference on Infant Studies*, 2010. URL http://langcog.stanford.edu/materials/icis_calib_slides.pdf.

Z. M. Griffin. Gaze durations during speech reflect word selection and phonological encoding. *Cognition*, 82(1):B1–B14, November 2001. URL http://www.sciencedirect.com/science/article/B6T24-4471K7X-5/2/4ac7f41cba3a6b8f0815acbb20a80ba6.

Z. M. Griffin and K. Bock. What the eyes say about speaking. *Psychological Science*, 11(4):274–279, July 2000. URL http://www.ncbi.nlm.nih.gov/entrez/utils/fref.fcgi?PrId=3046&itool=Citation-def&uid=11273384&db=pubmed&url=http://www.blackwell-synergy.com/openurl?genre=article&sid=nlm:pubmed&issn=0956-7976&date=2000&volume=11&issue=4&spage=274.

Z. M. Griffin and D. M. Oppenheimer. Speakers gaze at objects while preparing intentionally inaccurate labels for them. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 32(4):943–948, Jul 2006. doi: 10.1037/0278-7393.32.4.943. URL http://dx.doi.org/10.1037/0278-7393.32.4.943.

P. G. T. Healey, N. Swoboda, I. Umata, and Y. Katagiri. Graphical interaction and the emergence of abstraction. In *Proceedings of IGC2000: First International Workshop on Interactive Graphical Communication*, pages 77–84, August 2000.

D. R. Hofstadter. *Fluid Concepts and Creative Analogies*. Basic Books, 1995.

A. J. Hornof and T. Halverson. Cleaning up systematic error in eye-tracking data by using required fixation locations. *Behav Res Methods Instrum Comput*, 34(4):592–604, Nov 2002.

E. Hutchins. How a cockpit remembers its speeds. *Cognitive Science*, 19(3):265 – 288, 1995. ISSN 0364-0213. doi: DOI:10.1016/0364-0213(95)90020-9. URL http://www.sciencedirect.com/science/article/pii/0364021395900209.

A. Hyrskykari. Utilizing eye movements: Overcoming inaccuracy while tracking the focus of attention during reading. *Computers in Human Behavior*, 22(4):657 – 671, 2006. ISSN 0747-5632. doi: DOI:10.1016/j.chb.2005.12.013. URL http://www.sciencedirect.com/science/article/B6VDC-4J72YW1-2/2/8ba5466cc55abcaaf639bc20d15314c3. Attention aware systems - Special issue: Attention aware systems.

H. Ishii and M. Kobayashi. Clearboard: A seamless medium for shared drawing and conversation with eye contact. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 525–532, Monterey, CA, USA, May 3-7 1992. ACM Press. URL http://doi.acm.org/10.1145/142750.142977.

L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10-12):1489 – 1506, 2000. ISSN 0042-6989. doi: DOI:10.1016/S0042-6989(99)00163-7. URL http://www.sciencedirect.com/science/article/pii/S0042698999001637.

L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, Mar 2001. doi: 10.1038/35058500. URL http://dx.doi.org/10.1038/35058500.

P. Jermann and P. Dillenbourg. Group mirrors to support interaction regulation in collaborative problem solving. *Computers and Education*, 51:279–296, August 2008. ISSN 0360-1315. doi: 10.1016/j.compedu.2007.05.012. URL http://portal.acm.org/citation.cfm?id=1371264.1371453.

P. Jermann, M.-A. Nüssli, and W. Li. Using dual eye-tracking to unveil coordination and expertise in collaborative Tetris. In *24th ACM BCS Conference on Human Computer Interaction*, 2010.

Y. Kammerer. How to overcome the inaccuracy of fixation data – the development and evaluation of an offset correction algorithm. In *he Scandinavian Workshop on Applied Eye-Tracking (SWAET)*, 2009.

R. Karsh and F. W. Breitenbach. *Looking at looking: The amorphous fixation measure*, chapter I.5, pages 53–64. Lawrence Erlbaum Associates, 1983.

D. Kirsh and P. Maglio. On distinguishing epistemic from pragmatic action. *Cognitive Science*, 18(4):513 – 549, 1994. ISSN 0364-0213. doi: DOI:10.1016/0364-0213(94)90007-8. URL http://www.sciencedirect.com/science/article/B6W48-46H14R9-7/2/25e4fd5479799e3138a70e0270aa4317.

W. Kosnik, J. Fikre, and R. Sekuler. Visual fixation stability in older adults. *Investigative Ophthalmology & Visual Science*, 27(12):1720–1725, Dec 1986.

B. Law, M. S. Atkins, A. E. Kirkpatrick, and A. J. Lomax. Eye gaze patterns differentiate novice and experts in a virtual laparoscopic surgery training environment. In *ETRA '04: Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 41–48, New York, NY, USA, 2004. ACM. ISBN 1-58113-825-3. doi: http://doi.acm.org/10.1145/968363.968370.

Y. Liu, P.-Y. Hsueh, J. Lai, M. Sangin, M.-A. Nüssli, and P. Dillenbourg. Who is the expert? analyzing gaze data to predict expertise level in collaborative applications. In *Proceedings of the 2009 IEEE international conference on Multimedia and Expo*, ICME'09, pages 898–901, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-4290-4. URL http://dl.acm.org/citation.cfm?id=1698924.1699144.

B. R. Manor and E. Gordon. Defining the temporal threshold for ocular fixation in free-viewing visuocognitive tasks. *Journal of Neuroscience Methods*, 128(1-2):85 – 93, 2003. ISSN 0165-0270. doi: DOI:10.1016/S0165-0270(03)00151-1. URL http://www.sciencedirect.com/science/article/B6T04-495694F-1/2/51e8628f44e55ce3ce52f4bae05a72f5.

S. Martinez-Conde, S. L. Macknik, and D. H. Hubel. The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*, 5(3):229–240, March 2004. doi: 10.1038/nrn1348. URL http://dx.doi.org/10.1038/nrn1348.

A. Meier, H. Spada, and N. Rummel. A rating scheme for assessing the quality of computer-supported collaboration processes. *International Journal of Computer-Supported Collabo-*

rative Learning, 2(1):63–86, Mar. 2007. ISSN 1556-1607. doi: 10.1007/s11412-006-9005-x. URL http://dx.doi.org/10.1007/s11412-006-9005-x.

A. S. Meyer, A. M. Sleiderink, and W. J. M. Levelt. Viewing and naming objects: eye movements during noun phrase production. *Cognition*, 66(2):B25 – B33, 1998. ISSN 0010-0277. doi: DOI:10.1016/S0010-0277(98)00009-2. URL http://www.sciencedirect.com/science/article/pii/S0010027798000092.

A. F. Monk and C. Gale. A look is worth a thousand words: Full gaze awareness in video-mediated conversation. *Discourse Processes*, 33(3):257–278, 2002. URL http://www.leaonline.com/doi/abs/10.1207/S15326950DP3303_4.

V. Navalpakkam and L. Itti. Modeling the influence of task on attention. *Vision Research*, 45(2):205–231, Jan 2005. doi: 10.1016/j.visres.2004.07.042. URL http://dx.doi.org/10.1016/j.visres.2004.07.042.

A. Noë. *Out of our heads: Why you are not your brain, and other lessons from the biology of consciousness.* Hill and Wang, 2009.

M. Nyström and K. Holmqvist. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavioral Research Methods*, 42(1):188–204, Feb 2010. doi: 10.3758/BRM.42.1.188. URL http://dx.doi.org/10.3758/BRM.42.1.188.

M.-A. Nüssli, P. Jermann, M. Sangin, and P. Dillenbourg. Collaboration and abstract representations: towards predictive models based on raw speech and eye-tracking data. In *CSCL '09: Proceedings of the 2009 conference on Computer support for collaborative learning*. International Society of the Learning Sciences, 2009.

M. J. Pickering and S. Garrod. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27(2):169–90; discussion 190–226, Apr 2004.

S. Pietinen, R. Bednarik, T. Glotova, V. Tenhunen, and M. Tukiainen. A method to study visual attention aspects of collaboration: eye-tracking pair programmers simultaneously. In *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 39–42, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-982-1. doi: http://doi.acm.org/10.1145/1344471.1344480.

G. Rewari and C.-S. Poon. Saccadic eye movement detection using kalman filtering and the generalized likelihood ratio approach. In *Engineering in Medicine and Biology Society, 1993. Proceedings of the 15th Annual International Conference of the IEEE*, pages 418–419, 1993.

D. C. Richardson and R. Dale. Looking to understand: The coupling between speakers' and listeners' eye movements and its relationship to discourse comprehension. *Cognitive Science*, 29(29):1045–1060, 2005. URL http://psych.ucsc.edu/eyethink/publications_assets/RichardsonDale2005.pdf.

D. C. Richardson, R. Dale, and N. Z. Kirkham. The art of conversation is coordination: Common ground and the coupling of eye movements during dialogue. *Psychological Science*, 18(5): 407–413, June 2007. doi: 10.1111/j.1467-9280.2007.01914.x.

D. D. Salvucci and J. R. Anderson. Automated eye-movement protocol analysis. *Human-Computer Interaction*, 16(1):39–86, 2001. ISSN 0737-0024. doi: http://dx.doi.org/10.1207/S15327051HCI1601_2.

D. D. Salvucci and J. H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *ETRA '00: Proceedings of the 2000 symposium on Eye tracking research & applications*, pages 71–78, New York, NY, USA, 2000. ACM. ISBN 1-58113-280-8. doi: http://doi.acm.org/10.1145/355017.355028.

M. Sangin. *Peer Knowledge Modeling in Computer Supported Collaborative Learning*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2009.

M. Sangin, G. Molinari, M.-A. Nüssli, and P. Dillenbourg. How learners use awareness cues about their peer's knowledge: Insights from synchronized eye-tracking data. In *ICLS*, number 2, pages 287–296, 2008.

M. Sangin, G. Molinari, M.-A. Nüssli, and P. Dillenbourg. Facilitating peer knowledge modeling: Effects of a knowledge awareness tool on collaborative learning outcomes and processes. *Computers in Human Behavior*, 27(3):1059 – 1067, 2011. ISSN 0747-5632. doi: 10.1016/j.chb.2010.05.032. URL http://www.sciencedirect.com/science/article/pii/S074756321000172X. <ce:title>Group Awareness in CSCL Environments</ce:title>.

D. Sauter, B. J. Martin, N. D. Renzo, and C. Vomscheid. Analysis of eye tracking movements using innovations generated by a kalman filter. *Medical & biological engineering & computing*, 29(29):63–69, January 1991.

D. L. Schwartz. The emergence of abstract representations in dyad problem solving. *The Journal of the Learning Sciences*, 4(3):321–354, 1995. ISSN 10508406. URL http://www.jstor.org/stable/1466735.

F. Shic, K. Chawarska, and B. Scassellati. The amorphous fixation measure revisited: with applications to autism. In *Proceedings of the 30th Annual Meeting of the Cognitive Science Society*, pages 1964–1969. Cognitive Science Society, July 23-26 2008a. URL http://www.cogsci.rpi.edu/CSJarchive/proceedings/2008/pdfs/p1964.pdf.

F. Shic, B. Scassellati, and K. Chawarska. The incomplete fixation measure. In *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research and applications*, pages 111–114, New York, NY, USA, 2008b. ACM. ISBN 978-1-59593-982-1. doi: http://doi.acm.org/10.1145/1344471.1344500.

M. Tanenhaus, M. Spivey-Knowlton, K. Eberhard, and J. Sedivy. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632–1634, 1995.

doi: 10.1126/science.7777863. URL http://www.sciencemag.org/cgi/content/abstract/268/5217/1632.

G. T. Timberlake, M. A. Mainster, E. Peli, R. A. Augliere, E. A. Essock, and L. E. Arend. Reading with a macular scotoma. i. retinal location of scotoma and fixation area. *Investigative Ophthalmology & Visual Science*, 27(7):1137–1147, Jul 1986.

User's Guide to Tobii Programming Interfaces, Version 1.0. Tobii Technology AB, SE-113 59 Stockholm.

User Manual - Tobii Eye Tracker and ClearView analysis software. Tobii Technology AB, 2006.

Velichkovsky and B. M. Communicating attention: Gaze position transfer in cooperative problem solving. *Pragmatics & Cognition*, 3(2):199–223, 1995.

A. von Mayrhauser and A. M. Vans. Program comprehension during software maintenance and evolution. *Computer*, 28:44–55, August 1995. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.402076. URL http://dx.doi.org/10.1109/2.402076.

C. Weigle and D. C. Banks. Analysis of eye-tracking experiments performed on a tobii t60. In K. Börner, M. T. Gröhn, J. Park, and J. C. Roberts, editors, *Visualization and Data Analysis 2008*, volume 6809. SPIE, 2008. URL http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal&id=PSISDG006809000001680903000001&idtype=cvips&gifs=yes.

L. Williams. Integrating pair programming into a software development process. In *Software Engineering Education and Training, 2001. Proceedings. 14th Conference on*, pages 27 –36, 2001. doi: 10.1109/CSEE.2001.913816.

A. L. Yarbus. *Eye Movements and Vision*. Plenum Press, New York, 1967. Translated from Russian by Basil Haigh.

G. J. Zelinsky and G. L. Murphy. Synchronizing visual and language processing: an effect of object name length on eye movements. *Psychological Science*, 11(2):125–131, Mar 2000.

J. Zhang. The nature of external representations in problem solving. *Cognitive Science*, 21 (2):179 – 217, 1997. ISSN 0364-0213. doi: DOI:10.1016/S0364-0213(99)80022-6. URL http://www.sciencedirect.com/science/article/pii/S0364021399800226.

Y. Zhang and A. J. Hornof. Mode-of-disparities error correction of eye-tracking data (to appear). *Behavior Research Methods*, Apr 2011. doi: 10.3758/s13428-011-0073-0. URL http://dx.doi.org/10.3758/s13428-011-0073-0.

Marc-Antoine Nüssli

21 September 1981
Swiss / French
(+41) 79 247 09 59
marc-antoine.nuessli@epfl.ch

PROFILE

I work in IT domain for 15 years. Starting as an independent computer installer/repairer, I moved towards software development of various kinds. Besides, I have a great interest in science and research, which conducted me to do a PhD. I have a rigorous and logical mind and I have very good programming skills. I can be completely autonomous but I also enjoy working and sharing knowledge with others.

EDUCATION

PhD in Computer Science 2011
Ecole Polytechnique Fédérale de Lausanne (EPFL)

M.Sc. in Cognitive Sciences 2007
Université Lumière Lyon 2

M.Sc. In Computer Science (final grade 4.95/6.00, diploma 5.0/6.0) 2005
Ecole Polytechnique Fédérale de Lausanne (EPFL)

Swiss Federal Matura (major in science) 2000
Gymnase d'Yverdon (CESSNOV)

EXPERIENCE

Ecole Polytechnique Fédérale de Lausanne, PhD Student

July 2007 – November 2011
Dual Eye-Tracking Methods for the Study of Remote Collaborative Problem Solving: relate collaboration processes to gaze patterns of people working on collaborative tasks.

- Development of a gaze analysis framework (PostgreSQL, Python, Java, Scala)
- Important contributions to methodology of eye-tracking
- Development of dual eye-tracking methodology and tools (C++)
- Run and analyze several dual eye-tracking experimental studies (Matlab, R)
- Main teaching assistant for the course on computer-supported collaborative work for 4 years

Euratlas (www.euratlas.com), Software Engineer

January 2003 – present (part time job at 20-40%)
Small company specialized in historical cartography. I work as an autonomous developer in charge for all software developments in the company. This includes the followings realizations:

- Various web CGI in HTML/Perl (secure download area, mailing list, …)
- Development of a historical maps visualization software (Java)
- Development of a OGC-compliant Historical-GIS database (PostGIS, Java)
- Development of dedicated end-user software to explore the HGIS database (C++)
  (A trial version can be downloaded at http://www.euratlas.net/tele/periodis-en-demo.exe)

MeteoSwiss, Software Engineer

July 2005 – April 2006 (Internship for 6 months followed by a short-term contract)
I was in charge of the development of data visualization and analysis software

- Development of a visualization software for atmospheric sounding data (JAVA)
- Development of a data processing software for ozone measurements (LabView)

Marbrerie Pidoux SA, System and database administrator

1997 – 2005 (part time job at ~20%)

- Installation and administration of a small network on Windows (5 workstations)
- Development of a database for offers and invoices (Access, Visual Basic)

TECHNICAL

- Expert programming knowledge: Java, C/C++
- Good programming knowledge : Python, Perl, Scala, HTML, SQL, VisualBasic, Labview
- Scientific computing: Matlab, R, Weka
- Database: PostgreSQL, MySQL, Access, Oracle
- Operating systems: Linux (very good), Windows (good), Mac OS X (medium)
- Hardware: VHDL, FPGA development
- Various: GIS data processing, data visualization, machine learning, statistics, data processing

LANGUAGES

- French: Mother tongue
- English: Fluent oral and written (TOEFL iBT: 101/120)
- German: Basic oral and written knowledge

INTERESTS - ACTIVITIES

- Science in general and especially in epistemology, cognitive sciences and complex systems
- Sports: climbing, scuba-diving (PADI Rescue Diver), Water skiing, biking
- Guitar playing, reading, traveling

PUBLICATIONS (most representatives are underlined)

Journal papers

G. Molinari, M. Sangin, P. Dillenbourg, and M.-A. Nüssli. (2009) Knowledge interdependence with the partner, accuracy of mutual knowledge model and computer-supported collaborative learning. *European Journal of Psychology of Education*, Special Issue: Social Dynamics in Judgment and Performance in Academic Settings.

M. Sangin, M.-A. Nüssli, and P. Dillenbourg. (2010) Facilitating peer knowledge modeling: Effects of a knowledge awareness tool on collaborative learning outcomes and processes. *Computers in Human Behavior*, 2010.

M. Cherubini, M.-A. Nüssli, and P. Dillenbourg. (2010) This is it ! : Indicating

and looking in collaborative work at distance. *Journal of Eye-Movement Research*, 3(5):1-20, 2010.

Conference papers

M.-A. Nüssli, C. Nüssli. (2010) Euratlas : From Historical Mapping To Historical Geographical Information System. In the 35th Annual Meeting of the Social Science History Association, 2010

W. Li, M.-A. Nüssli, and P. Jermann. (2010) Gaze quality assisted automatic recognition of social contexts in collaborative Tetris. *In 12th ACM International Conference on Multimodal Interfaces (ICMI)*, 2010.

P. Jermann, M.-A. Nüssli, and W. Li. (2010) Using dual eye-tracking to unveil coordination and expertise in collaborative Tetris*. In 24th ACM BCS Conference on Human Computer Interaction, 2010.*

Y. Liu, P.-Y. Hsueh, J. Lai, M. Sangin, M.-A. Nüssli, and P. Dillenbourg. (2009) Who is the expert? analyzing gaze data to predict expertise level in collaborative applications. *In Proceeding ICME'09 Proceedings of the 2009 IEEE international conference on Multimedia and Expo*, pages 898-901. IEEE Press, 2009.

M.-A. Nüssli, P. Jermann, M. Sangin, and P. Dillenbourg. (2009) Collaboration and abstract representations: towards predictive models based on raw speech and eye-tracking data*. In CSCL '09: Proceedings of the 2009 conference on Computer support for collaborative learning. International Society of the Learning Sciences, 2009.*

M. Cherubini, M.-A. Nüssli, and P. Dillenbourg. (2008) Deixis and gaze in collaborative work at a distance (over a shared map): a computational model to detect misunderstandings. *In Proceedings of the International Symposium on Eye Tracking Research & Applications (ETRA2008)*, pages 173-180. ACM Press, 2008.

M. Sangin, G. Molinari, M.-A. Nüssli, and P. Dillenbourg. (2008) How learners use awareness cues about their peer knowledge? insights from synchronized eye-tracking data. *In International Conference of the Learning Sciences, 2008.*

G. Molinari, M. Sangin, M.-A. Nüssli, and P. Dillenbourg. (2008) Effects of informational interdependence on visual attention and action transactivity in collaborative concept mapping. *In Proceedings of the International Conference of the Learning Sciences, 2008.*

G. Molinari, M. Sangin, M.-A. Nüssli, and P. Dillenbourg. (2008) When Co-Learners Work on Complementary Texts: Effects on Outcome Convergence. *In Proceedings of the Third European Conference on Technology Enhanced Learning*, Lecture Notes in Computer Science, pages 304-311. Springer, 2008.

M. Cherubini, M.-A. Nüssli, and P. Dillenbourg. (2007) Deixis and coupling of partners'eye movements in collaborative work at distance. *In Extended Abstracts of 2007 International ACM Conference on Supporting Group Work*, pages 9-10. ACM Press, 2007.

REFERENCES

Professor Pierre Dillenbourg, CRAFT, Ecole Polytechnique Fédérale de Lausanne (EPFL), pierre.dillenbourg@epfl.ch, +41 21 693 20 71