PROBABILISTIC MODELING OF TEXTURE TRANSITION FOR FAST TRACKING AND DELINEATION

THÈSE N^{O} 3377 (2005)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut des systèmes informatiques et multimédias

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Seyed Ali SHAHROKNI

M.Sc. in Electrical Engineering, University of Tehran, Iran
et de nationalité iranienne

acceptée sur proposition du jury:

Prof. P. Fua, directeur de thèse
Dr T. Drummond, rapporteur
Prof. N. Paragios, rapporteur
Prof. S. Süsstrunk, rapporteur

Lausanne, EPFL
2005

To my loving family

Acknowledgements

Accomplishment of this thesis would have never been such a memorable and rewarding experience without interaction with many people.

First of all, I am thankful to the thesis jury members: Prof. Daniel Thalmann, Dr. Tom Drummond, Prof. Nikos Paragios and Dr. Sabine Süsstrunk for accepting to evaluate this work.

A huge investment in my thesis originates from the useful discussions with Dr. Vincent Lepetit that I would like to thank profoundly for his patience and valuable hints and ideas. It was also a great advantage for me to be exposed to the mathematical and machine learning knowledge of Dr. François Fleuret as well as his "Parisian" sense of humor. I am indebted to Dr. Tom Drummond for his key ideas that bestowed richness to this thesis and also for the great time that I spent during my visit to his laboratory at University of Cambridge. I would like to express immense gratitude to my thesis advisor, Prof. Pascal Fua, for all the knowledge that he transmitted to me and for forging a "researcher" out of me!

Last but not least, I should thank my dear colleagues at CVLAB and VRLAB, EPFL, my family and relatives in Iran for their support as well as my friends who were by my side in my fascinating adventures in the land of Helvètes!

Abstract

In this thesis a probabilistic approach to texture boundary detection for tracking applications is presented. We have developed a novel fast algorithm for Bayesian estimation of texture transition locations from a short sequence of pixels on a *scanline* that combines the desirable speed of edge-based line search and the sophistication of Bayesian texture analysis given a small set of observations. For the cases where the given observations are too few for reliable Bayesian estimation of probability of texture change we propose an innovative machine learning technique to generate a probabilistic texture transition model. This is achieved by considering a training dataset containing small patches of blending textures. By encompassing in the training set enough examples to accurately model texture transitions of interest we can construct a predictor that can be used for object boundary tracking that can deal with few observations and demanding cases of tracking of arbitrary textured objects against cluttered background.

Object outlines are then obtained by combining the texture crossing probabilities across a set of scanlines. We show that a rigid geometric model of the object to be tracked or smoothness constraints in the absence of such a model can be used to coalesce the scanline texture crossing probabilities obtained using the methods mentioned above. We propose a Hidden Markov Model to aggregate robustly the sparse transition probabilities of scanlines sampled along the projected hypothesis model contour. As a result continuous object contours can be extracted using a posteriori maximization of texture transition probabilities. On the other hand, stronger geometric constraints such as available rigid models of the target are directly enforced by robust stochastic optimization.

In addition to being fast, the allure of the proposed probabilistic framework is that it accommodates a unique infrastructure for tracking of heterogeneous objects which utilizes the machine learning-based predictor as well as the Bayesian estimator interchangeably in conjunction with robust optimization to extract object contours robustly. We apply the developed methods to tracking of textured and non textured rigid objects as well as deformable body outlines and monocular articulated human motion in challenging conditions. Finally, because it is fast, our method can also serve as an interactive texture segmentation tool.

Résumé

Cette thèse présente une approche probabiliste de la détection de changement de texture appliquée au suivi d'objets texturés. Nous avons développé un nouvel algorithme pour estimer la position d'un changement de texture à partir d'une courte séquence de pixels. Cet algorithme cumule la rapidité des méthodes basées sur le gradient et la sophistication de l'analyse Bayésienne de textures étant donné un ensemble limité d'observations. Lorsque les observations données ne sont pas suffisantes pour une estimation Bayésienne fiable de la probabilité du changement de texture, nous proposons une technique novatrice d'apprentissage pour construire un modèle probabiliste de transition de textures. Ceci est obtenu en considérant un ensemble d'apprentissage formé de petites images composées de deux moitiés de textures différentes. Ayant suffisamment d'exemples dans la base d'apprentissage pour modeler exactement des transitions de texture d'intérêt, nous pouvons construire un prédicteur qui peut être employé pour le suivi de frontière d'objets.

Les probabilités de transitions de texture ainsi obtenues sur plusieurs *scanlines* peuvent être combinées pour obtenir des contours d'objet. Nous démontrons qu'un modèle géométrique rigide de l'objet à suivre ou des contraintes de connectivité en l'absence d'un tel modèle peuvent être employés pour fusionner les probabilités de transition de textures de *scanlines* obtenues en utilisant les méthodes mentionnées ci-dessus. Nous proposons un Modèle de Markov Caché pour combiner les probabilités de transition de scanlines de manière robuste. En conséquence, des bords d'objet continus peuvent être extraits en maximisant des probabilités postérieures de transitions de textures. D'autre part, des contraintes géométriques plus fortes telles que les modèles rigides de la cible sont directement imposées par optimisation stochastique robuste.

En plus d'être rapide, le cadre proposé fournit une infrastructure unique pour le suivi d'objets hétérogènes qui utilise aussi bien un prédicteur basé sur des techniques d'apprentissage statistique qu'un estimateur Bayésien, combiné avec une optimisation robuste pour extraire des bords d'objet. Nous appliquons les méthodes développées au suivi, dans les conditions difficiles, d'objets rigides texturés ou non texturés ainsi que des contours déformables de corps et mouvement humain articulé monoculaire. Grâce à sa rapidité, nous avons pu utiliser notre méthode pour un outil interactif de segmentation de texture.

Contents

List of Figures

Chapter 1

Introduction

Outlines play a fundamental role in computerized methods of object recognition, tracking and segmentation. In this thesis, we investigate ways to automatically detect object outlines for tracking purposes. Unlike earlier approaches that relied on gradient-based detection, we focus here on analysis of texture boundaries and transitions. Outline and silhouettes are particularly helpful in our perception of the surroundings. As discussed in Appendix A, texture interpretation plays an important role in helping the visual system locate object boundaries.

Traditionally, edge-based tracking methods tend to rely on processing image gradient information. While gradient-based tracking algorithms are computationally fast and therefore favorable, they can easily fail due to loss of intensity information. Due to this fact and considering the guidelines from the study of the human visual system given in Appendix A, in a system dedicated to fast boundary extraction, the traditional gradient-based method should be extended to handle texture models. Unfortunately, available techniques for texture analysis are not adapted to fast contour extraction for tracking in that respect. The main distinction between the concept of texture segmentation and texture-based tracking is the fact that for tracking a known object we are mainly interested in the ways its texture and boundary appear with respect to the background. Therefore it can be stated that for tracking applications the "relative texture" of the target and background and transitions between texture processes are the main issue, while in texture segmentation problem we are interested in criteria

that allow partitioning of an image into separate regions and therefore the emphasis is more on regional properties rather than local interaction of textures. Moreover, the tracking problem is associated with a set of assumptions regarding the object's appearance and motion which can be exploited to restrict the search to a limited search space and thus reduce the computations. These are the issues that need to be addressed in order to apply texture analysis approaches to the tracking problem.

In this thesis, we explore ways to identify texture boundaries can be identified for tracking and interactive segmentation purposes. These investigations result in several texture-based approaches to finding the projected contours of 3–D objects while retaining the speed of standard edge-based techniques. More specifically, we explore silhouette tracking techniques that can deal with the following kinds of models.

1.1 Rigid Models

Detection of outlines of objects with predefined geometric shapes, such as those shown in Fig. 1.1 is of broad interest. In this case geometrical constraints imposed by the object shape, in conjunction with probabilistic sampling methods, serve as a reliable mechanism for outline extraction. Here the challenge is to handle complex geometries in a manageable manner and at a low computational cost.

1.2 Deformable Models

Next we consider the outlines of deformable and non-regular shapes such as human body with textured or non-textured clothing and arbitrary background such as the one shown in Fig. 1.2 without prior shape models. These silhouettes are characterized by their continuity and often separate locally coherent patterns. The texture on both sides of these outlines can be of arbitrary complexity or lack any distinctive pattern. We show that Hidden Markov Models provide appropriate tools for handling such cases. Interactive object selection tools and blob tracking applications can benefit from this method.

Figure 1.1: Our method has been used for 3–D edge tracking using texture boundary detection for geometrically well-defined objects.

1.3 Articulated Model

3–D tracking of model-based articulated human motion is key to a large number of activities and applications such as security, character animation, virtual reality, human-machine interfaces, biomechanics studies, traffic and customer monitoring. To date, a number of promising silhouette-based approaches to body tracking and human pose estimation have been proposed (Agarwal & Triggs, 2004a; Mittal *et al.*, 2003; Sminchisescu & Triggs, 2003). However most of them rely on the fact that silhouettes can be extracted using relatively simple algorithms such as background subtraction (Davis & Bobick, 1998) or standard edge- and gradient-based techniques (Agarwal & Triggs, 2004a; Athitsos & Sclaroff, 2003).

However in practice, and as in the other cases discussed above, this assumption rarely holds and these silhouette extraction methods can be very brittle. They

Figure 1.2: Interactive silhouette detection. Our method can extract a smooth outline of the subject's dress (thick red curves) given an initial guess (thin green curve).

tend to fail in the presence of highly textured objects and clutter, which produce too many irrelevant edges. In such situations, it would be advantageous to detect texture boundaries instead. However, because texture segmentation techniques require computing statistics over image patches, they tend to be computationally intensive and have therefore not been felt to be suitable for such purposes. Furthermore, all of the commercially available techniques for motion capture require either employing dedicated human operators or using ad-hoc sensors. This tends to make them:

- Cumbersome. The user needs to wear markers or other ad-hoc equipment which may be impractical, uncomfortable, constrain the user to a limited work space, be difficult to transport.

- Expensive. They require both hardware and skilled human operators.

- Slow. The data only becomes available after a lag required to process batches of images using manual techniques.

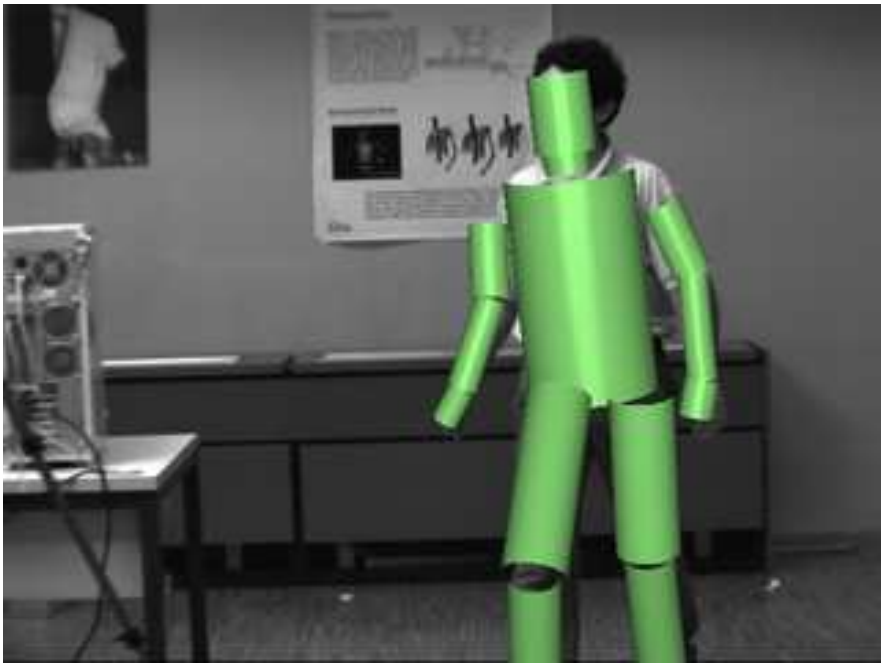


Figure 1.3: 3–D model-based articulated tracking using silhouettes.

If motion capture could become both automated and non-invasive, these limitations would disappear and many more applications would become practical. Multi-camera approaches have the potential to achieve this goal. However, single camera solutions would be even more widely applicable. This is a challenging problem because it involves tackling such difficult issues as ambiguities associated with articulated motion seen from a single camera, very high dimensional search spaces, self-occlusions, and poor quality of image features in the absence of markers. Therefore in addition to body outlines we employ constraints such as 2–D image point matching across frames and impose joint limits for the articulated structure to ensure realistic poses and avoid ambiguities to some extend. Fig. 1.3 depicts the articulated model we use for tracking of human motion.

So far we discussed three main challenging areas in computer vision where silhouettes can be used and studied the problems associated with them. In the next section we go briefly through the contributions of this thesis in each of those areas.

1.4 Achievements

In this thesis a probabilistic approach to texture boundary detection for tracking is presented. We have developed a fast and efficient algorithm for Bayesian estimation of texture transition from a small sequence of pixels on a *scanline*. This is very efficient when enough observations are available for reliable estimation of the texture models. In the absence of this condition, we use machine learning techniques to efficiently generate a probabilistic texture transition model which can be used for tracking of arbitrary textured objects against cluttered background. We then combine the texture crossing probabilities across a set of scanlines using robust probabilistic contour extraction.

The allure of this probabilistic framework is that it accommodates a unique infrastructure for using the machine learning-based predictor as well as the Bayesian estimator interchangeably in conjunction with robust optimization to extract object contours robustly. We apply the developed methods to tracking of textured and non textured rigid objects as well as deformable body outlines and monocular articulated human motion in challenging conditions in addition to interactive texture segmentation.

In the remainder of this section we briefly outline these methods.

1.4.1 Fast Bayesian Boundary Estimation

We propose a texture-based approach to finding the projected contours of objects while retaining the speed of standard edge-based techniques. Our technique is inspired by work of Drummond & Cipolla (2001) on edge-based tracking that starts from the estimated projection of a 3–D object model and performs a line search in the direction perpendicular to the projected edges to find the most probable boundary location. This is challenging because speed requirements compel us to restrict ourselves to computing statistics along a line, and therefore a fairly limited number of pixels. We calculate the probability of texture crossing by integration of likelihood probabilities using a uniform prior on the statistical model of the texture processes. We demonstrate that the exact Bayesian estimation of the texture transition probability can be calculated in closed form using a Markov model of texture process.

We will show that this rigorous and formal statistical treatment has allowed us to reach our goal under demanding circumstances. Our proposed algorithm results in a near real-time 3–D tracker that uses only a fraction of the computational resources of a modern PC, thus opening the possibility to simultaneously track many objects on ordinary hardware. Furthermore, this opens the door for other applications that require near real-time performance, such as the interactive drawing of image boundaries that are becoming increasingly popular in image-processing systems such as Photoshop.

However, the approach as described above suffers from a couple of drawbacks. The first issue is that for complex texture transitions we need more observations to build an accurate texture model. This calls for longer scanlines which is not always possible. We also need to come up with a way to combine several scanline results to extract object boundaries. Moreover, in case of using a prior texture model of the object to be tracked, we need to establish a way to update that texture model in the course of tracking as the appearance of the object changes. We overcome these problems by incorporating the following tools:

- *Scanstripes.* We extend scanlines to scanstripes which help the texture models converge faster as described in Section 4.4.

- HMM. A Hidden Markov Model is defined in Chapter 6 to bind and exploit local texture information in form of texture crossing probabilities.

- Stochastic optimization. When available geometric constraints are imposed by robust stochastic model fitting to texture crossing probabilities. These ideas are explained in Chapter 6.

- Kullback-Leibler Divergence (KLD): The relative entropy between the actual target texture and the prior target texture model. It is measured in the course of tracking and used to dynamically update the texture model.

We will show that this yields more robust tracking behavior, for example under lighting changes.

1.4.2 Machine Learning Approach

As mentioned in Section 1.4.1, the Bayesian estimation of texture cut requires relatively long scanlines to reliably estimate models for complex textures and to calculate the probability of the boundary locations between them. An alternative to using scanstripes is to train a boundary predictor to obtain a probabilistic measure of texture transition using a small image patch. We investigate the use of machine learning techniques to detect boundaries between potentially textured regions.

We demonstrate the effectiveness of this approach in the context of tracking of rigid or deformable objects. More specifically, we show how a small set of weak learners can be used to estimate the conditional probability of texture discontinuity, given a small pixel neighborhood. Weighted sum of weak learners can be seen as an approximation of a log-likelihood which, similar to the Bayesian estimator, can be combined with robust techniques to enforce shape constraints. These constraints together with trained weak learner responses provide a natural way to detect contours of objects which are not attainable through conventional methods.

1.5 Summary

In this thesis we explore machine learning and statistical modeling techniques through which texture boundaries can be identified for tracking and interactive segmentation. We look for silhouette extraction methods that can achieve reliable, fast and robust pattern transition detection for geometrically well-defined as well as deformable or 3–D articulated objects. Our main contribution can be regarded as the introduction of a novel probabilistic formalism for detection of object outlines for applications that require low computational costs, which were traditionally supplied by gradient-based algorithms.

This thesis is organized as folllows. In Chapter 2 we describe the gradient-based approaches to tracking and the line search tracking scheme that is used in this thesis to apply texture models to object tracking. In Chapter 3, texture analysis and segmentation methods are described and their strength and weaknesses

for tracking applications are discussed. We then introduce a fast Markov-based line search method for texture boundary detection in Chapter 4. An alternative method based on trained classifiers is then presented in Chapter 5. These methods can be used to robustly extract object contours with geometric constraints. This issue is addressed in Chapter 6. Finally, we provide analysis and comparison of the performance of different proposed methods in Chapter 7 and show some experimental results in Chapter 8.

Chapter 2

Edge-based Tracking Using Gradient

Historically, the early approaches to tracking were all gradient-based mostly because these methods are both computationally efficient, and relatively easy to implement. They are also naturally stable to lighting changes, even for specular materials. In this chapter, we discuss these approaches which can be grouped into two categories:

- One approach is to look for strong gradients in the image around a first estimation of the object pose, without explicitly extracting the contours (Comport *et al.*, 2003; Drummond & Cipolla, 2002; Harris, 1992; M. Armstrong and A. Zisserman, 1995; Marchand *et al.*, 2001; Vacchetti *et al.*, 2004). This is fast and general.

- Another approach is to first extract image contours, such as straight line segments and to fit the model outlines to these image contours (Gennery, 1992; Koller *et al.*, 1993; Kosaka & Nakazawa, 1995; Lowe, 1992; Ruf *et al.*, 1997). In this case the loss in generality is compensated by a gain in robustness.

The first category is of special importance to this work, because we employ a similar line search approach for fast object tracking and delineation. The principal difference is that the we replace the gradient-based objective function with texture-based models of pattern transition.

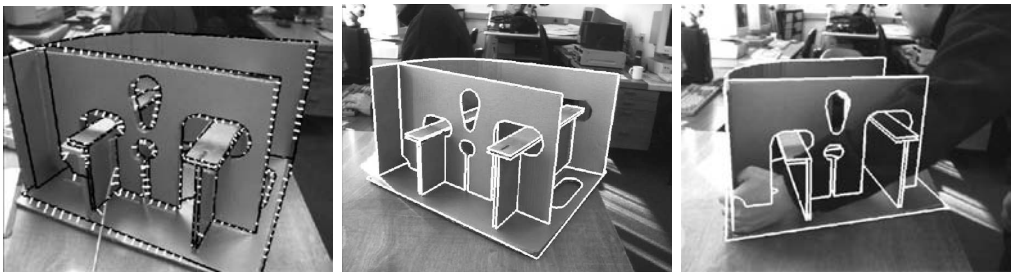


Figure 2.1: In RAPiD-like approaches, control points are sampled along the model edges. The small white segments in the left image join the control points in the previous image to their detected position in the new image. The pose can be inferred from these matches, even in presence of occlusions by introducing robust estimators. (From Drummond & Cipolla (2002), figures courtesy of T. Drummond and R. Cipolla).

2.1 RAPiD

Because of its low computational complexity, RAPiD (Harris, 1992) was one of the first 3–D trackers to successfully run in real-time. Even though many improvements have been proposed since, we describe it here in detail because many of its basic components have been retained in more recent systems. Furthermore, we use the same search and optimization framework in our texture-based 3–D tracking approach for rigid objects. The key idea is to consider a set of 3–D object points, called control points, that are most likely to project on high-contrast image edges. As shown in Fig. 2.1, the control points can be sampled along the 3–D model edges and in the areas of rapid albedo change. They can also be generated on the fly as points on occluding contours. The 3–D motion of the object between two consecutive frames can be recovered from the 2–D displacement of the control points.

Once initialized, the system performs a simple loop: For each frame, the predicted pose, which can simply be the pose estimated for the previous frame, is used to predict which control points will be visible and what their new locations should be. The control points are matched to the image contours, and the new pose is estimated from these correspondences. For each control point, the system looks for its projection $\mathbf{m}'$ in the new image around $\mathbf{m}$, its projection in

the previous frame. Because of the aperture problem, the position $\mathbf{m}'$ can not be completely determined. As depicted by Fig. 2.2 only the perpendicular distance l of $\mathbf{m}$ from the appropriate image edge is measured. Assuming that the orientations of the image edge and the model edge are nearly the same, a one-dimensional search for the image edge is conducted by looking in the direction of a vector $\vec{\mathbf{n}}$ from $\mathbf{m}$ where $\vec{\mathbf{n}}$ is a unit vector orthogonal to the projected object contour at point $\mathbf{m}$. For a fast implementation, the search is often performed in the horizontal, vertical, or diagonal direction, closest to the actual edge normal. (Comport *et al.*, 2003) uses a precomputed convolution kernel function of the contour orientation to find only edges with an orientation similar to the reprojected contour orientation, as opposed to all edges in the scan-line.

In the original RAPiD formulation, the motion is estimated in the object coordinate system whose origin is located at $\mathbf{T} = (T_x, T_y, T_z)^T$ in camera coordinates, and whose axes are aligned with the camera coordinate system. A control point $\mathbf{P} = (P_x, P_y, P_z)^T$ in object coordinates is expressed as $\mathbf{M} = \mathbf{T} + \mathbf{P} = (X, Y, Z)^T$ in the camera frame. Its projection in the image is then $\mathbf{m} = \mathbf{KM}$ with $\mathbf{K}$ being the matrix of internal camera parameters as defined in Appendix B.

After a motion $\delta\mathbf{p}$ rotating the object about the object origin by $\Delta\mathbf{R}$ and translating it by $\delta\mathbf{t}$, the control point location in camera coordinates becomes $\mathbf{M}' = \mathbf{T} + \delta\mathbf{t} + \Delta\mathbf{RP}$. RAPiD assumes that the new image is acquired after a small motion, which makes it possible to linearize the object projection with respect to motion. The rotation $\Delta\mathbf{R}$ can be approximated as $\Delta\mathbf{R} \approx \mathbf{I} + \mathbf{\Omega}$, where $\mathbf{\Omega}$ is a skew-symmetric matrix. Thus, we have $\mathbf{M}' \approx \mathbf{M} + \delta\mathbf{t} + \mathbf{\Omega P}$. The expression of the projection $\mathbf{m}'$ of $\mathbf{M}'$ can then be expanded in $\mathbf{\Omega}_x$, $\mathbf{\Omega}_y$, $\mathbf{\Omega}_z$ and $\delta\mathbf{t}$ and, by retaining only terms up to first order, becomes

$$\begin{aligned} u' &= u + \tfrac{1}{T_z+P_z}\left(\delta\mathbf{t}_x + \mathbf{\Omega}_y P_z - \mathbf{\Omega}_z P_y - u\left(\delta\mathbf{t}_z + \mathbf{\Omega}_x P_y - \mathbf{\Omega}_y P_x\right)\right) \quad , \\ v' &= v + \tfrac{1}{T_z+P_z}\left(\delta\mathbf{t}_y + \mathbf{\Omega}_z P_x - \mathbf{\Omega}_x P_z - v\left(\delta\mathbf{t}_z + \mathbf{\Omega}_x P_y - \mathbf{\Omega}_y P_x\right)\right) \quad . \end{aligned} \tag{2.1}$$

This can be written in matrix form as

$$\mathbf{m}' = \mathbf{m} + \mathbf{W}\delta\mathbf{p}\,, \tag{2.2}$$

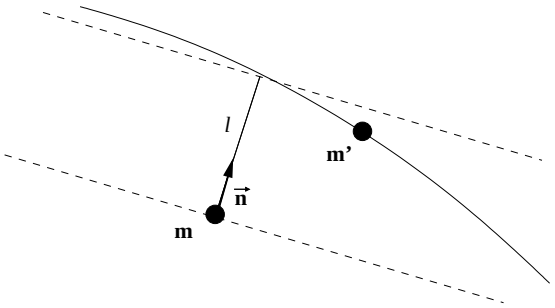


Figure 2.2: Searching for the new control point position in RAPiD-like approaches. $\mathbf{m}$ is the predicted control point position and $\mathbf{m}'$ is the actual control point position. The search is performed along the direction of vector $\vec{\mathbf{n}}$, and only the perpendicular distance l is measured.

where $\delta\mathbf{p} = (\mathbf{\Omega}_x, \mathbf{\Omega}_y, \mathbf{\Omega}_z, \delta\mathbf{t}_x, \delta\mathbf{t}_y, \delta\mathbf{t}_z)^T$ is a six-vector made of the rotation coefficients and the translation components, and $\mathbf{W}$ a 2×6 matrix that is a function of the coordinates of $\mathbf{T}$ and $\mathbf{P}$.

The distance l of Fig. 2.2 can be written as

$$l = \tilde{\mathbf{n}}^T (\mathbf{m}' - \mathbf{m}) \ . \tag{2.3}$$

From Eqs. (2.3) and (2.2), each control point $\mathbf{M}_i$ then yields one equation of the form

$$\tilde{\mathbf{n}}_i \mathbf{W}_i \delta\mathbf{p} = l_i \ .$$

Given enough control points, $\delta\mathbf{p}$ can then be computed by minimizing the sum of squares of the perpendicular distances, which we write as

$$\delta\mathbf{p} = \underset{\delta\mathbf{p}}{\operatorname{argmin}} \sum_i (\tilde{\mathbf{n}}_i \mathbf{W}_i \delta\mathbf{p} - l_i)^2 \ . \tag{2.4}$$

Therefore, it can be recovered by solving a least-squares problem of the form

$$\mathbf{l} = \mathbf{A}\delta\mathbf{p} \ ,$$

where $\mathbf{l}$ is the vector made of the distances l_i and $\mathbf{A}$ depends on the $\tilde{\mathbf{n}}_i$ and $\mathbf{W}_i$. The solution can then be found using the pseudo-inverse of matrix $\mathbf{A}$, and taking $\delta\mathbf{p}$ to be

$$\delta\mathbf{p} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}\mathbf{l}.$$

Finally the pose $\mathbf{p}$ is incremented by $\delta\mathbf{p}$, which yields $\mathbf{p}^t = \mathbf{p}^{t-1} + \delta\mathbf{p}$.

In (Harris, 1992), some enhancements to this basic approach are proposed. When the edge response at a control point becomes too weak, it is not taken into account into the motion computation, as it may subsequently incorrectly latch on to a stronger nearby edge. As we will see below, this can also be handled using a robust estimator. An additional clue that can be used to reject incorrect edges is their polarity, that is whether they correspond to a transition from dark to light or from light to dark. A way to use occluding contours of the object is also given. In (Evans, 1990), integrating a Kalman filter into RAPiD is proposed.

The control points can be defined on the fly. Harris (1992) show how profile edge points can be created along occluding contours defined by the model projection. Marchand *et al.* (2001) also discuss the discretization of the model edges visible at time t to produce the control points for the estimation of the pose at time $t + 1$.

2.2 Robust RAPiD

The main drawback of of the original RAPiD formulation is its lack of robustness. The weak contours heuristics is not enough to prevent incorrectly detected edges from disturbing the pose computation. In practice, such errors are frequent. They arise from occlusions, shadows, texture on the object itself, or background clutter.

Several methods have been proposed to make the RAPiD computation more robust. Drummond & Cipolla (2002) use a robust estimator and replaces the least-squares estimation by an iterative reweighted least-squares to solve the new problem. Marchand *et al.* (2001) use a framework similar to RAPiD to estimate a 2–D affine transformation between consecutive frames, but substitutes a robust estimator for the least-squares estimator of Eq. (2.4). The affine transformation is used to infer an approximate 3–D pose, which is then refined as will be discussed Section 2.4.

In fact, when using a more powerful minimization algorithm, linearizing the problem is not required, instead one can minimize the actual distances between the detected features and the reprojected 3–D primitives. Let the $\mathcal{M}_i$ be those

primitives, and let $\left\{\mathbf{m}'_{i,j}\right\}$ be the set of associated image features, the pose can now be estimated as:

$$\mathbf{p} = \underset{\mathbf{p}}{\operatorname{argmin}} \sum_{i,j} \rho\left(\operatorname{dist}\left(\mathbf{P_p}\mathcal{M}_i, \mathbf{m}'_{ij}\right)\right) , \tag{2.5}$$

where $\mathbf{P_p}$ is the projection defined by parameters $\mathbf{p}$ and $\mathbf{P_p}\mathcal{M}_i$ denotes the 2–D curve obtained projecting $\mathcal{M}_i$. For example, Marchand & Chaumette (2002) discuss the computation of the relevant Jacobian matrices when the 3–D primitives such as straight lines segments, circles or occluding boundaries of cylinders can be defined analytically. Simon & Berger (1998) consider free-form curves and uses an approximation of the distance.

In the approaches described above, the control points were treated individually, without taking into account that several control points are often placed on the same edge, and hence their measurements are correlated. By contrast, in (M. Armstrong and A. Zisserman, 1995; Simon & Berger, 1998) control points lying on the same object edge are grouped into primitives, and a whole primitive can be rejected from the pose estimation. In (M. Armstrong and A. Zisserman, 1995), a RANSAC methodology is used to detect outliers among the control points forming a primitive. If the number of remaining control points falls below a threshold after elimination of the outliers, the primitive is ignored in the pose update. Using RANSAC implies that the primitives have an analytic expression, and precludes tracking free-form curves. By contrast, Simon & Berger (1998) use a robust estimator to compute a local residual for each primitive. The pose estimator then takes into account all the primitives using a robust estimation on the above residuals.

When the tracker finds multiple edges within its search range, it may end-up choosing the wrong one. To overcome this problem, in (Drummond & Cipolla, 2002), the influence of a control point is inversely proportional to the number of edge strength maxima visible within the search path. Vacchetti *et al.* (2004) introduce another robust estimator to handle multiple hypotheses and retain all the maxima as possible correspondents in the pose estimation.

2.3 Explicit Edge Extraction

The previous approaches rely on matching points sampled on edges. An alternative approach is to globally match model primitives with primitives extracted from the image (Gennery, 1992; Koller *et al.*, 1993; Kosaka & Nakazawa, 1995; Lowe, 1992; Ruf *et al.*, 1997). In these specific examples, the primitives are straight line segments, but, in theory, they could be more complex parametric curves.

For each image, straight line edge segments are extracted, while the model edge segments are projected with respect to the predicted pose. The matching is based on the Mahalanobis distance of line segment attributes. For example, in Koller *et al.* (1993) segments are represented by $\mathbf{X} = (c_x, c_y, \theta, l)$ defined by the coordinates of the middle point, the orientation and the length of the segment Deriche & Faugeras (1990). Given the attribute vector $\mathbf{X}_m$ of a model segment and the attribute vector $\mathbf{X}_d$ of an extracted segment, the Mahalanobis distance between $\mathbf{X}_m$ and $\mathbf{X}_d$ can be then defined as

$$d = (\mathbf{X}_m - \mathbf{X}_d)^T (\Lambda_m + \Lambda_d)^{-1} (\mathbf{X}_m - \mathbf{X}_d) \,, \tag{2.6}$$

where Λ_d is the covariance matrix of $\mathbf{X}_d$, and depends on the extraction procedure. The covariance matrix Λ_m of a model segment depends on the covariance matrix of the predicted pose estimation. Kosaka & Nakazawa (1995) also integrate the uncertainty in the Hough transform used for segment extraction and limits its search to the uncertainty region predicted by the Λ_m matrices.

An iterative procedure is used to find the best correspondences between 3–D model edge segments M_i and 2–D image segments D_i, while estimating the pose. In Koller *et al.* (1993), a model segment M_i is matched with the closest data segment D_i according to the Mahalanobis distance of Eq. (2.6), if this distance is lower than a threshold. The pose $\mathbf{p}$ is then estimated by minimizing

$$\sum_i \left(\mathbf{X}_d^i - \mathbf{X}_m^i(\mathbf{p})\right)^T \Lambda_d^i \left(\mathbf{X}_d^i - \mathbf{X}_m^i(\mathbf{p})\right) \,, \tag{2.7}$$

with respect to $\mathbf{p}$ where $\mathbf{X}_m^i(\mathbf{p})$ is the attribute vector of the model segment M_i projected with respect to the pose $\mathbf{p}$. An additional term can be added to

the criterion of Eq. (2.7) to account for a motion model. The minimization is performed using the Levenberg-Marquardt algorithm. The process is repeated until a stable pose is found.

Such approaches, which are only adapted to polyhedral object tracking, have been applied to vehicle and robot arm tracking, but they seem to have fallen out of use and been replaced by RAPiD like algorithms. We believe this can be attributed to bottom-up nature of the edge-extraction process, which makes it unreliable. The RAPiD approach both avoids this drawback thanks to the local search around an *a priori* pose and tends to be significantly faster.

2.4 Direct Optimization on Gradients

Balcisoy *et al.* (2001); Kollnig & Nagel (1997); Marchand *et al.* (2001) propose to recover the pose by fitting the model projection directly to the image gradients. A simple approach is to maximize the gradient norm along the model reprojection but there is no guarantee that the model edges should correspond to high intensity gradient values.

It is better to take into account the expected direction of the projected contour: Marchand *et al.* (2001) propose to minimize the sum of the values $\frac{\nabla I.\vec{\mathbf{n}}}{\|\nabla I\|^2}$, where ∇I denotes the spatial gradient of the image I, and $\vec{\mathbf{n}}$ the expected direction. This measure tends to support locations where the gradient is both strong and in the expected direction. Kollnig & Nagel (1997) maximize the correlation between the predicted and the measured gradient norm plus an additional term to constrain the motion.

Such approaches require a very good initial estimate to converge to the correct pose. Therefore, they are best used as a refinement step.

2.5 Summary

Different methods of edge-based tracking using gradient information have been discussed in this chapter. While line search methods that look for strong gradients are very fast and general, they lack robustness associated with methods that explicitly extract contours and then fit the model outlines to them. Explicit

extraction of reliable gradient features on the other hand is not always possible nor efficients. To overcome these shortcomings in the next chapter we consider texture analysis tools for segmentation and tracking and then in the remainder of this thesis we explore methods to combine the two approaches of line search and texture analysis.

Chapter 3

State of the Art on Texture Analysis

Contour-based tracking finds applications in a wide variety of domains such as polyhedral object tracking (Drummond & Cipolla, 2002), human (Agarwal & Triggs, 2004a), face (Gupta *et al.*, 2004), hand tracking (Thayananthan *et al.*, 2004), road detection (Taylor *et al.*, 1996), etc. The most common way to extract contours used for these applications is either based on local gradient information (Drummond & Cipolla, 2002; Taylor *et al.*, 1996) or edge distance transform (Agarwal & Triggs, 2004b; Athitsos & Sclaroff, 2003; Gupta *et al.*, 2004; Thayananthan *et al.*, 2004) as discussed in Chapter 2. These methods are appealing due to their simplicity and speed, but their application is restricted to the cases where the contrast is sufficient. Furthermore, these methods tend to fail in the presence of highly textured objects and clutter, which produce too many irrelevant edges. This is due to their inadequacy to exploit the intensity values and structures available in the image. The main reason for not utilizing this information is the complexity of texture patterns and difficulty to adapt texture segmentation and analysis techniques to tracking applications.

However, like many other problems in vision, texture segmentation is ill posed. Texture segmentation is an old and important topic in image processing and computer vision. It aims at dividing a textured image into several regions with the same certain characteristics. An effective and efficient texture segmentation method finds application in the analysis of aerial, biomedical and seismic

images as well as the automation of industrial tasks. The segmentation of textures involves determination of suitable and discriminative features. Three major categories for texture feature extraction methods are statistical, structural and spectral features. In statistical approaches, different textures are distinguished using statistical moments of the histogram, or statistics based on cooccurrence matrix. Structural approaches use "Textons", the basic element of textures, to derive rules that are used to generate complex texture patterns. Finally, in spectral approaches , frequency domain transformation of texture is used to analyze textures (Lee & Chen, 2001).

The fact of partitioning the image into subregions of "uniform texture" raises some fundamental issues which are discussed below (Rubner & Tomasi, 1996).

- Complex procedures are required to explicitly identify uniform texture regions and often the application requires information about boundaries rather than parts inside the regions of similar texture instead. Processing regions is computationally expensive and not always helpful. This is true for tracking and interactive segmentation problems which include useful hypotheses about the nature of the problem. This information needs to be exploited to obtain better results with less effort.

- There is no concrete definition for "texture", for instance in Fig. 3.1 it is not meaningful to state that the purple fish has a uniform texture, on the other hand we expect the texture segmentation algorithm to extract the fish as a single region. The fundamental fact is that the notion of uniform texture is of dubious validity and it depends on the perception of the observer. Moreover two similar textures in the world can produce unsimilar patterns in the images due to various effects such as difference in distance, foreshortening, shading and lighting. On the other hand, a detailed and sophisticated model which can cover all these effects might lead to unrealistic and restrictive criterion.

- Due to noise and blurring introduced by the photographing device and its sensor, the exact location of texture boundaries is not well determined. These physical phenomena can cause textures to blend into each other and therefore the boundary to be blurred.

Figure 3.1: Can we find a proper definition for the texture of this purple fish?

Based on the above discussion, in the remainder of this chapter we review the existing approaches which can be used to extract object delineations. There exists various ways of categorizing these methods, here we classify them into texture segmentation using texture operators, graph cut and supervised learning-based methods which correspond approximately to their complexity and chronological order of emergence.

3.1 Texture Operators

Statistical and structural descriptors are used extensively in texture segmentation and classification approaches. Statistical operators describe the formation of texture with the statistical properties o pixel intensities and their spatial configuration. Cooccurrence statistics and histogram analysis are examples of this approach. In the structural approach however the concept of texture primitives is employed to describe a texture composition. In this section we briefly review these approaches. This survey is inspired by the work of Mäenpää (2003).

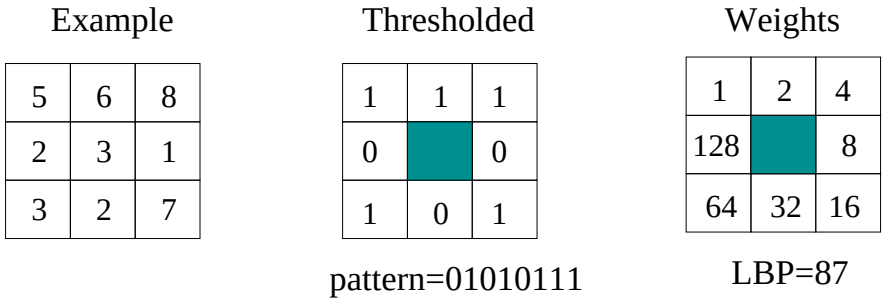


Figure 3.2: Local Binary Pattern is computed by thresholding the pixels in a window using the intensity value of the central pixel and summing up the weighted binary values.

3.1.1 Local Binary Pattern Methods

The Local Binary Pattern (LBP) operator introduced by Ojala *et al.* (1996) is a gray scale invariant descriptor which measures the contrast of textures. It performs well in terms discrimination performance and contains simultaneously structural and statistic information. Heikkila *et al.* (2004) use local binary pattern analysis to extract moving objects for blob tracking people by background subtraction assuming a static background. However, the extracted contours using such techniques are not usually detailed and accurate enough to be used for 3–D tracking.

The LBP operator assigns a binary code to a pixel based on its neighborhood as illustrated by the example shown in Fig. 3.2. The intensity value of the central pixel is used to threshold the neighboring pixels to generate a binary pattern. This binary pattern is then converted to the LBP code by summing up the weighted thresholded values. The LBP operator can also be made invariant to rotation and scale.

3.1.2 Cooccurrence Features

Statistics of gray levels of pairs of pixels can be measures by computation of a two-dimensional cooccurrence matrix. This matrix is formed by considering a position operator and counting the number of pairs of intensity values that fit

Figure 3.3: Examples of position operator used to compute cooccurrence matrices. The dots show the considered positions for pairs of gray level values.

in the spatial configuration described by the position operator. In other words, the cooccurrence matrix represents the joint probability of pairs of gray levels that occur in relative positions defined by the position operator (Tomita & Tsuji, 1990). Fig. 3.3 shows examples of position operator which compares two pixels at different positions. Cooccurrence matrices can be used to extract useful and discriminant statistical features such as energy, entropy, correlation, homogeneity and inertia (Conners & Harlow, 1980).

Other extensions of cooccurrence statistics include the gray-level difference (GLD) method (Weszka *et al.*, 1976) which is invariant to changes in global luminance of an image due to the fact of considering the difference of gray levels instead of their absolute values. Later, multidimensional cooccurrence features have also been introduced by Valkealahti & Oja (1998) for texture analysis.

3.1.3 Extensions of LBP

The LBP can be regarded as a special case of gray-level difference methods. It reduces the dimensionality by considering only the sign of the differences and therefore omits the need for vectorial quantization for dimensionality reduction. Other extensions of LBP include tri-level thresholding instead of the original bi-level case. As a result, the number of texture units increases considerably without remarkable improvement in performance (Mäenpää, 2003). N arbitrary neighbors can be used to generate the LBP code instead of the original eight-neighbors of a pixel. This is referred to as N-tuple method which is proposed by Patel &

Stonham (1992) to obtain gray-level texture cooccurrence spectrum (GLTCS) for texture image classification and segmentation.

3.1.4 Texton Statistics

In texture segmentation terminology, "textons" are the fundamental texture units that can be recognized by human vision (Julesz, 1981). Malik J (1999) refers to representative vectors of Gabor filter bank responses as textons in their model for texture segmentation. In this approach the texton vocabulary is created for a set of textures and then used for texture recognition (Leung & Malik, 2001; Varma & Zisserman, 2002).

The main distinction between Gabor filter and LBP operator is that LBP considers each pixel in the neighborhood while Gabor filter calculates the weighted mean of pixel values over a window. Therefore LBP can regarded as a micro-texton while Gabor filter represents macro-textons (Mäenpää, 2003). Studies show that the Gabor texton approach has inferior performance compared to LBP and cooccurrence methods on challenging sets of textures (Pietikäinen *et al.*, 2003; Randen & Husoy, 1999).

As an alternative to texture segmentation, Rubner & Tomasi (1996) proposed a method to coalesce Gabor descriptors of adjacent image patches with similar textural content into clusters. Significant regions are characterized as those with low texture contrast, which is defined as the maximum derivative of the texture descriptor vector. This is in principal similar to applying an edge-preserving smoothing, such as anisotropic diffusion filter (Perona & Malik, 1990), but in space of texture vectors rather than image intensities.

The compass edge detector operator (Ruzon & Tomasi, 1999a) uses the Earth Mover Distance (Rubner *et al.*, 2001) to measure the distance between color distributions in CIE-Lab color space on two sides of a potential edge position. The performance of the compass operator is superior to the one of purely gradient-based techniques such as Canny (1986), due to the fact that color distributions are introduced and their distance is measured during the operation. Their proposed method is also extended to detect corners (Ruzon & Tomasi, 1999b).

The effectiveness of different multi-scale edge detection cues on black and white and color images can be computed using information theoretic measures such as *Chernoff information* (Konishi *et al.*, 1999). They provide a statistical study of information supplied by edge detection cues. They use the Sowerby image database of country side to learn the distributions of off-boundary and on-boundary edge detector responses, P_{on} and P_{off} respectively. They show that the expected error rate of classification is bound exponentially by Chernoff information between P_{on} and P_{off} (Cover & Thomas, 1991).

Therefore the higher the Chernoff information provided by P_{on} and P_{off}, the more reliable the corresponding edge detector cue would be. In their work, Konishi et. al. have compared three edge detectors, namely magnitude of the intensity gradient, the Nitzberg edge detector (Nitzberg *et al.*, 1993) which is based on eigenvalues of the second moment matrix and the Laplacian of Gaussian. Their results shows superior performance of Nitzberg operator in multi-scale form on both gray-level or color images. Moreover they establish the fact that the step edge model is by far the most valid assumption of an edge detector. This is verified through psychophysiological studies of human observers and validated through tests (Konishi *et al.*, 1999). This observation is consistent with the results that are reported in this thesis which are obtained by exhaustive search of edge classification cues, as explained in Chapter 5.

Geodesic active region framework (Paragios & Deriche, 2001) is used in a level-set optimization scheme for image segmentation and tracking by Paragios & Deriche (2005). They exploit Minimum Description Length criterion and Maximum Likelihood in order to approximate image histogram by a mixture of Gaussians. Then different region boundaries are detected using a probabilistic edge detector which determines the local discontinuities in the statistical models of neighboring regions. Their regularized optimization of the region- and boundary-based (and in case of tracking also motion-based) objective function adaptively deals with changes of topology regardless of initial conditions. The key hypothesis is that image is composed of homogeneous regions which can be modeled with a Gaussian distribution. This is also true for the work of Hanek & Beetz (2004) who propose a method for fitting parametric curve models using the EM algorithm. They use local Gaussian distributions to model stripes perpendicular to the curve.

Assuming a Gaussian model for the texture reduces the flexibility of the algorithm. Paragios & Deriche (2002) relax the segmentation hypothesis that image is composed of homogeneous regions by learning probabilistic texture descriptors for a given set of texture patterns. They use multidimensional features captured using a set of filter operators namely, isotropic, anisotropic and Gabor filters. The training filter responses are then represented as a Gaussian mixture model which is used for segmentation. The problem of supervised and unsupervised texture segmentation is then cast as a geodesic active contour model scheme similar to (Paragios & Deriche, 2001) that looks for a minimal length geodesic curve with simultaneous boundary probability maximization and optimal region grouping.

3.2 Graph Cut Image Segmentation

A wide variety of computer vision tasks can be expressed in terms of a labeling problem to pixel regions on noisy image data. Stereo reconstruction, image restoration and segmentation are among such tasks which can be regarded as an optimization problem in presence of uncertainties. For such labeling problems a specialized graph can be defined corresponding to the energy function to be minimized which fits into an elegant minimum cut optimization framework. It can be shown that we can solve the maximum cut problem without exhaustive computation using maximum flow algorithms. Next, we briefly describe the graph cut optimization concept and review some related work.

3.2.1 Graph Representation

A graph is defined on the image to be segmented by considering each pixel as a node of a weighted graph $\mathcal{G} = (E, V)$, which consists of a set of edges E that connect nodes or vertices of the graph V, the weights of the graph are associated with the edges and measure the strength of the link between two nodes as illustrated in Fig. 3.4.

In a graph, the nodes with only ingoing or outgoing edges are referred to as terminal nodes, furthermore nodes with only inward edges are called sink and those with only outgoing edges are called source nodes. Similarly, edges

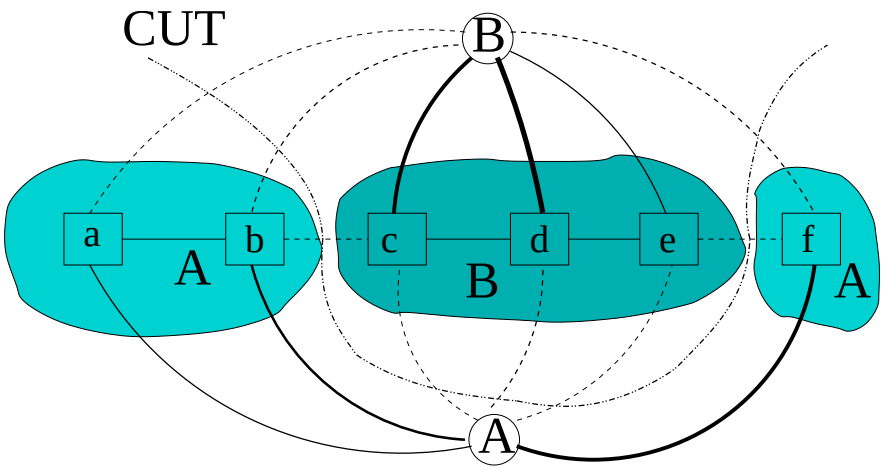


Figure 3.4: Illustration of a weighted graph $\mathcal{G}$ with two terminal nodes (labels). The weights of the edges are represented by their thickness.

connecting pixels to terminal nodes (or labels in case of segmentation) are called *t-link* and edges connecting pairs of neighboring pixels are called *n-links*. The goal is then to find a labeling for all pixels (nodes) that minimize an energy function of the form

$$E(l) \;=\; \sum_{p} D(p, l_p) \;+\; \sum_{p} \sum_{q \in V_p} V_{\{p,q\}}(l_p, l_q) \tag{3.1}$$

where $D(p, l_p)$ is the cost of assigning label l_p to pixel p which indicates how well the intensity of the pixel is defined by the given label and represents the t-link edges. $V_{\{p,q\}}(l_p, l_q)$ is the regularization term in the neighborhood of pixel p defined by V_p. It represents the n-links in the graph and is a measure of how well the pixel's label agrees with neighboring labels.

The smallest set of edges to cut that will disconnect a graph can be efficiently found using flow methods by defining a capacity on each edge which is equal to its weight and searching the maximum flow from source to sink satisfying the capacity constraints. However smallest set obtained this way favors isolated clusters and therefore normalized cuts must be used which take into account size of different clusters. The minimum normalized cut solution is however an NP-hard problem. the minimum cut is shown in Fig. 3.4 where edge weights are represented by their thickness. Shi & Malik (2000) use normalized cut approach

to locate clusters. Their method performs recursive bisection using eigenvector of the second smallest eigenvalue of the Laplacian matrix of the graph (the degree matrix [1] minus the adjacency matrix [2])

An alternative to the normalized minimum cut approach is the concept of graph commute times which leads to clusters of nodes with increased dissociation of separate clusters and association inside each cluster. Association is a measure of total edge linkage within a graph cluster (Qiu & Hancock, 2005). The commute time between a pair of nodes in a graph is related to the closeness of the two nodes, the value of the sum of weights of the path relating the two nodes, and the number of paths between them. Spectral graph theory and analysis of how information flows with time across the graph is used to model commute time.

The spectral analysis of graph is characterized by the heat equation and the Laplacian eigensystem. This formalism can be used to compute random walk on the graph for image segmentation, (Meila & Shi, 2001), among other applications in the field of information retrieval and pattern analysis. However, most of the proposed methods are confined to improper approximations based on the major eigenvector of the Laplacian matrix.

Graph cut techniques are closely related to the theory of Markov Random Fields. The optimization of the graph cut energy function of Eq. 3.1 is commonly done using the Markov Random Field (MRF) model which is described next.

3.2.2 Markov Random Fields

The model of a Markov Random Field is defined by a set of neighborhood system, N, a set of sites (pixels), S, and a random field F representing the site labels. The MRF model states that the probability of having a configuration of labels on sites must satisfy the Markov property that each random variable depends only on the label of its neighbors. This idea is useful in the optimization of the

[1] A diagonal matrix corresponding to a graph that has the vertex degree (i.e. number of touching edges) of v_i in the i^{th} position.

[2] the adjacency matrix for a finite graph $\mathcal{G}$ with n vertices is an $n \times n$ matrix where the non diagonal entry a_{ij} is the number of edges joining vertex i and vertex j, and the diagonal entry a_{ii} is twice the number of loops at vertex i.

energy function using Bayes theorem that relates the optimization of the energy function of Eq. 3.1 with the maximum a posterior (MAP) of the Markov field.

Graph cut has been used in interactive texture segmentation problems such as work by Blake *et al.* (2004) where one starts from an initial user-provided guess about foreground, background and undetermined regions and the goal is to segment the undetermined regions to foreground and background subregions. Other proposed texture segmentation methods based on graph cuts (Boykov & Jolly, 2001; Kolmogorov & Zabih, 2004; Rivera & Gee, 2004) require computing Markov Random Field (MRF) models from a training set and require heavy computation for optimization. These features tend to make such approaches not suitable for tracking.

Rother *et al.* (2004) introduced a technique based on graph cuts which yields impressive interactive segmentation results thanks to improvements in the optimization stage and local post-processing. Their relies on post-processing to extract details and therefore not adapted for tracking using 2–D contours.

Another prototype texture-based tracker is the set forth by Ozyildiz *et al.* (2002) in which a formulation for fusing texture and color is presented in a manner that makes the segmentation reliable while keeping the computational cost low. The texture is modeled by an auto binomial Gibbs Markov Random Field (GMRF) while a 2–D Gaussian distribution is used for modeling the color. However since the MRF model is used to learn the global characteristics of target texture, it can not extract fine contours of the object as is required by 3–D pose tracking algorithms. Instead, this method is used to track the object as a whole in the image. Moreover training MRF's would lead to difficulties in tracking objects with different local textures and patterns.

Another issue with Gibbs potential field is the difficulty to obtain good priors by sampling the potential field. Kumar *et al.* (2005) introduce a Bayesian framework to insert priors in form of pictorial structures in MRF's. Their method is used to detect and segment instances of a particular object category within an image. The pictorial structures form a set of 2–D probabilistic patterns which contain information on shape, appearance and spatial configuration of patches representing object parts which ensure a global shape prior across the image plane.

MRF-based optimization has also been used by Paragios & Ramesh (2001) to change detection and crowding/congestion density estimation for subway monitoring. Their method consists of change detection using a discontinuity preserving MRF formalism that combines different sources such as color and temporal information with additional constraints to provide a detection map. The final objective function is minimized in a multi-scale fashion in order to fulfill real-time constraints and results in a geometric measure of occupancy rather than precise detection and tracking of each individual object.

3.3 Learning-based Texture Segmentation

Machine learning methods have been used extensively for object detection (Fleuret & Geman, 2002; Viola & Jones, 2001) and recognition, character recognition and speech processing. However, we are interested in learning low level generic discriminant characteristics of textures which can be used in finding texture discontinuities (cuts) in a narrow image band. The main motivation behind this idea is that such a tool can prove to be useful in a wide variety of applications that require reliable and fast boundary detection.

Figure 3.5: Image patches used to learn texture boundaries in natural images. Images courtesy of Martin *et al.* (2004).

An interesting work on learning texture boundaries in natural images is put forth by Martin *et al.* (2004). Their goal is to use features such as brightness, color, and texture measures to estimate the posterior probability of a boundary passing through the center point of an image patch in form of a disc as shown in Fig. 3.5. Examples of their reported results are shown in Fig. 3.6. They use a large database of manually segmented natural images as the training set to

model the probability of a pixel being on or off-boundary conditioned on some set of local image features. These image features consist of two brightness features (oriented energy and brightness gradient), one color feature (color gradient) and one texture feature (texture gradient). The parameters of each of these features are trained based on the training database. The gradient-based features involve obtaining histograms of intensity, chrominance or banks of texture filter responses on two sides of oriented discs and then calculating the χ^2 distance between the histograms on both sides.

The above cues are then combined into a single function that gives the posterior probability of a boundary at each pixel and orientation. The task of combination is treated as a supervised learning problem where the combination rule is learned from a hand-marked dataset. Different types of classifiers reported by Martin *et al.* (2004) such as SVM, classification trees, density estimation, logistic regression, boosted regression, quadratic logistics and hierarchical mixture have equal performances with different measures of stability, bias and variance. The performance is measured in term of F-measure and precision-recall curves. F-measure is a metric that is widely used in information retrieval and machine learning algorithms. The two components of F-measure are recall and precision. Recall measures how well a search method pinpoints the desired outputs and precision measures how well it weeds out undesired results. Mathematically, the recall is the ratio of the number of positive examples correctly recognized to the total number of all positive examples in the database. The precision is the ratio of the number of true positive responses to the total number of positive detections. F-measure is then defined below.

$$F - measure = \frac{recall \times precision}{\alpha \times recall + (1 - \alpha) \times precision}$$

Texture can also be defined by statistical properties of the regions. For instance Will *et al.* (2000) define the texture as the statistical distribution of Gabor filter responses and come up with a tunable criterion which is optimized for simultaneous detection reliability and localization accuracy as in Canny's edge detector (Canny, 1986). Their method detects edges by calculating the a posteriori probability of a pixel or site belonging to a class. Assuming that the

Figure 3.6: Results obtained by Martin *et al.* (2004) using the texture and brightness gradient cues (second row) of images shown in the top row. Human extracted edges are shown in the third row. Images courtesy of Martin *et al.* (2004).

Gabor-responses are statistically independent, their methods amounts to finding the pixels with equal probability for both textures.

3.4 Summary

An overview of different approaches to boundary extraction is given in this chapter. Several categories have been presented which include texture operators, graph cuts and machine learning methods. While gradient-based algorithms are the computationally favorable, they suffer from vulnerabilities due to loss of intensity information. Texture segmentation techniques and graph cuts using Markov random fields on the other hand are computationally expensive and not adapted to tracking. Machine learning can be used to learn transition between textures and model arbitrary texture boundaries, however, to date, it has been applied only to texture segmentation and thus lacks specific requirements of tracking applications such as speed and integration of the geometrical constraints of the tracking target.

Another important issue in tracking is the evolution of the target model. Updating the prior model is of particular importance in tracking since the initial model of the target changes in the course of tracking due to vicissitudes in lighting conditions and object views. In the texture-based tracking scheme proposed in Ozyildiz *et al.* (2002) a statistical model for the adaptation over time of the mean and covariance vectors is proposed which uses L previous mean and covariance estimates. This causes rapid accumulation of the drift error and gives undesirable tracking results in a long sequence. We show that the Kullback-Leibler Divergence can be exploited to update the target model in an adaptive way.

Chapter 4

Line Search for Texture Boundary

In this chapter, we will first recall how contour tracking can be performed by line search through sample points. These principles apply to 2–D and 3–D tracking of rigid, deformable and articulated objects. We will then introduce a method to search for texture boundary based on Bayesian estimation of texture crossing probabilities.

In a standard tracking paradigm, successive pose parameters are estimated by minimizing the observed distances from the rendered model to the measured observations. The correspondence and the distance between model and measured silhouettes can be defined in two ways as discussed in Chapter 2.

One method is to fit the projected edges directly to the measured silhouettes. While robust this method requires an exhaustive nonlinear search. However as explained in detail in Section 2.1 and Drummond & Cipolla (2002) an approximate yet both sound and more practical alternative is to use RAPiD-like iterative closest point (ICP) algorithm for distance minimization between model and observation points. In this case we need to establish direct point correspondences on the model and the measured silhouettes. This is not particularly precise due to the approximate search directions associated with the model sampled points and the aperture problem which implies that the motion of a homogeneous contour is locally ambiguous. However,given the fact that usually the rendered model is

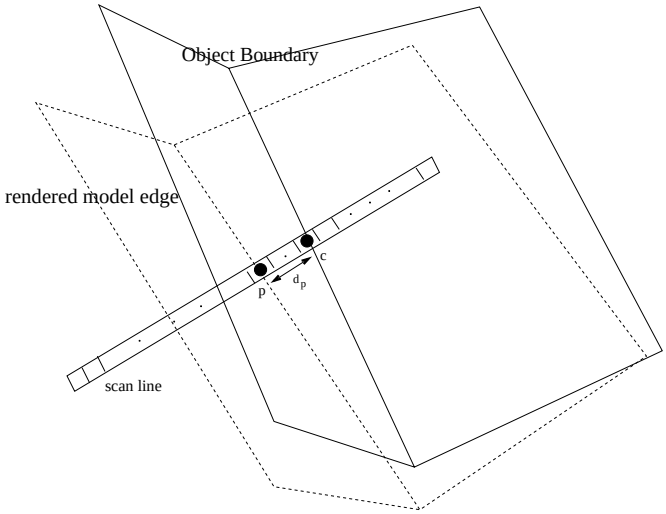


Figure 4.1: Contour-based 3-D tracking. Search for a real contour in the direction normal to projected edge. A scanline, centered on a hypothesis model sample p, is used to search for a texture crossing position c on the actual boundary of the object. The model pose is obtained by minimizing the distances d_p for all sample points.

close enough to the real silhouettes the approximate edge normals and the formulation given in Section 2.1 can be used for tracking without loss of precision.

As illustrated in Fig. 4.1 the search line directions are chosen to be orthogonal to projection of model silhouettes (dotted line) at the position of the model sample point p. The search line associated to a sample point on the model is referred to as *s*canline and used to locate the corresponding texture crossing point c, that is the point where the underlying statistics change. In the remainder of this chapter, we formalize the Bayesian search criteria and derive the algorithms we use to extract object boundaries using scanlines.

4.1 Scanlines

A texture is modeled as a statistical process which generates a sequence of pixels. The problem is then cast as follows: A sequence of n pixel intensities, $S_1^n = (s_1, s_2, ...s_n)$, is assumed to have been generated by two distinct texture

processes each operating on either side of an unknown change point, as shown in Fig. 4.1. Thus the observed data is considered to have been produced by the following process: First a changepoint c is selected uniformly at random from the range $[1 - n]$. Then the pixels to the left of the changepoint (the sequence S_1^c) are produced by a texture process T_1 and the pixels to the right (S_{c+1}^n) are produced by process T_2. The task is then to recover c from S_1^n. If both T_1 and T_2 are known then this corresponds to finding the c that maximizes:

$$P(\text{changepoint at } c|S_1^n, T_1, T_2) = KP(S_1^c|T_1)P(S_{c+1}^n|T_2) \ . \tag{4.1}$$

where K is a normalization constant. If one of the textures, for example, T_1 is unknown, then the term $P(S_1^c|T_1)$ must be replaced by the integral over all possible texture processes:

$$P(S_1^c) = \int P(S_1^c|T)P(T) \ \mathrm{d}T \ . \tag{4.2}$$

While it may be tempting to approximate this by considering only the most probable T to have generated S_1^c, this yields a poor approximation for small data sets, such as are exhibited in this problem. We show how the integral can be solved in closed form for reasonable choices of the prior $P(T)$ (e.g. uniform).

We consider two kinds of texture processes: first, one in which the pixel intensities are independently drawn from a probability distribution and second, one in which they are generated by a 1^{st} order Markov process, which means that the probability of selecting a given pixel intensity depends (only) on the intensity of the preceding pixel. We refer to these two processes as 0^{th} and 1^{st} order models.

4.1.1 Solving for the 0^{th} Order Model

The 0^{th} order model states that the pixel intensities are drawn independently from a probability distribution over I intensities ($T = \{p_i\}; i = 1..I$). If such a texture is known *a priori* then $P(S_1^c|T) = \prod_i p_{s_i}$. If the texture is unknown then:

$$P(S_1^c) = \int P(S_1^c|T)P(T) \ \mathrm{d}T = \int P(s_c|T)P(S_1^{c-1}|T)P(T) \ \mathrm{d}T \tag{4.3}$$

$$= P(S_1^{c-1}) \int p_{s_c} P(T|S_1^{c-1}) \ \mathrm{d}T \tag{4.4}$$

The integral in (4.4) is $E(p_{s_c}|S_1^{c-1})$: i.e. the expected value of the probability p_{s_c} in the texture given the observed sequence S_1^{c-1}. If we assume a uniform prior for T over the I-1 simplex of probability distributions, then this integral becomes:

$$E(p_{s_c}|S_1^{c-1}) = \frac{\int_0^1 \int_0^{1-p_1} \cdots \int_0^{1-\sum_{i=1}^{I-2} p_i} \; p_{s_c} \prod_{j=1}^{I} p_j^{o_j} \quad \mathrm{d}p_{I-1} \cdots \mathrm{d}p_2 \mathrm{d}p_1}{\int_0^1 \int_0^{1-p_1} \cdots \int_0^{1-\sum_{i=1}^{I-2} p_i} \prod_{j=1}^{I} p_j^{o_j} \; \mathrm{d}p_{I-1} \cdots \mathrm{d}p_2 \mathrm{d}p_1} \tag{4.5}$$

where there are o_j occurrences of symbol j in the sequence S_1^{c-1}. Note that both of these integrals have the same form, since the additional p_{s_c} in the numerator can be absorbed into the product by adding one to o_{s_c}. Substituting $p_I = 1 - \sum_{i=1}^{I-1} p_i$ and repeatedly integrating by parts yields:

$$E(p_{s_c}|S_1^{c-1}) = \frac{o_{s_c} + 1}{c + I - 1} \tag{4.6}$$

This result states that if an unknown probability distribution is selected uniformly at random and a set of samples are drawn from this distribution, then the expected value of the distribution is the distribution obtained by adding one to the number of instances of each value observed in the sample set.

For example, if a coin is selected with a probability of flipping heads randomly drawn from the uniform distribution over [0,1], and it is flipped 8 times, giving 3 heads and 5 tails, then the probability that the next flip will be heads is (3+1)/(8+2) = 0.4.

This result can be applied recursively to the whole sequence to give Algorithm 1.

4.1.2 Solving for the 1^{st} Order Model

This idea can be immediately extended to a 1^{st} order Markov process in which the intensities are drawn from a distribution which depends on the intensity of the preceding pixel ($T = \{p_{i|j}\}; i, j = 1..I$, where $p_{i|j}$ is the probability of observing intensity i given that the previous pixel had intensity j). These $p_{i|j}$ can be considered as a transition matrix (row i, column j). Again, the probability of a sequence given a known texture is easy to compute:

$$P(S_1^c|T) = P(s_1|T) \prod_{i=2}^{c} p_{s_i|s_{i-1}} \text{ where } P(s_1|T) \text{ is using the } 0^{th} \text{ order model.} \tag{4.7}$$

Algorithm 1 Rapid 0^{th} order computation of $\int P(S_1^c|T)P(T)\ \mathrm{d}T$.

```
sequence_probability (S[], c)
  dim Observations[NUM_CLASSES]
  // seed Observations[] with 1 sample per bin
  for i=1..NUM_CLASSES do
    Observations[i]=1
  end for
  Probability=1
  for i=1..c do
    Probability = Probability * Observations[S[i]] / ∑ Observations[]
    Observations[S[i]] = Observations[S[i]]+1
  end for
  return Probability
```

For a first order Markov process, the 0^{th} order statistics of the samples must be an eigenvector of $p_{i|j}$ with eigenvalue 1. Unfortunately, this means that a uniform prior for T over $p_{i|j}$ is inconsistent with the uniform prior used in the 0^{th} order case. To re-establish the consistency, it is necessary to choose a 1^{st} order prior such that the expected value of a column of the transition matrix $p_{i|j}$ is obtained by adding $1/I$ rather than 1 to the number of observations in that column of the co-occurrence matrix before normalizing the column to sum to 1. This means that the transition matrix is

$$E(p_{i|j}|S_1^c) = \frac{C_{ij} + 1/I}{1 + \sum_i C_{ij}} = \frac{C_{ij} + 1/I}{1 + o_j} \quad , \tag{4.8}$$

where C_{ij} is the number of times that intensity i follows intensity j in the sequence S_1^c. And hence the expected 0^{th} order distribution (which is the vector $\frac{(o_j+1)}{(c+I)}$) has the desired properties since

$$\sum_j E(p_{i|j}|S_1^c)\frac{(o_j+1)}{(c+I)} = \frac{\sum_j C_{ij} + 1/I}{c+I} = \frac{o_i+1}{c+I} \quad . \tag{4.9}$$

This modification is equivalent to imposing a prior over $p_{i|j}$ that favors structure in the Markov process and is proportional to $\prod_{ij} p_{i|j}^{(1/I-1)}$. This gives Algorithm 2.

Algorithm 2 Rapid 1^{st} order computation of $\int P(S_1^c|T)P(T)\ \mathrm{d}T$.

```
sequence_probability (S[], c)
  dim CoOccurrence[NUM_CLASSES][NUM_CLASSES]
  // seed CoOccurrence[][] with 1/NUM_CLASSES samples per bin
  for r=1..NUM_CLASSES do
    for c=1..NUM_CLASSES do
      CoOccurrence[r][c]=1/NUM_CLASSES
    end for
  end for
  Probability=1/NUM_CLASSES    // probability of the first symbol
  for i=2..c do
    Probability = Probability * CoOccurrence[S[i]][S[i-1]] / ∑
    CoOccurrence[][S[i-1]]
    CoOccurrence[S[i]][S[i-1]] = CoOccurrence[S[i]][S[i-1]]+1
  end for
  return Probability
```

4.2 The Binary Case

To fully comprehend the relationship between our developed algorithm and the Bayesian estimation of texture transition probability, let us consider analysis of the simplified case where we have only binary symbols 0 or 1. We denote by N the number of pixels in the line, by Θ_1 and Θ_2 two random variables uniform on $[0, 1]$ and by C the cut position, a random variable uniform on $\{1, \ldots, N\}$. The pixels are binary and independent and for the two texture processes separated by the cut at C we have

$$\forall i, 1 \leq i \leq C,\ P(X_i = 1) = \Theta_1$$

and

$$\forall i, C + 1 \leq i \leq N,\ P(X_i = 1) = \Theta_2\ .$$

4.2.1 Exact Conditional Probability

From this model we can compute the exact value of

$$P(C = c \,|\, S_1^N) \tag{4.10}$$

with $S_a^b = X_a, \ldots, X_b$. We denote the above term ρ_c for simplicity. It can be rewritten as follows:

$$\begin{aligned}
P(C = c \,|\, S_1^N) &= \int_{\theta_1} \int_{\theta_2} P(C = c, \Theta_1 = \theta_1, \Theta_2 = \theta_2 \,|\, S_1^N) \\
&= \tfrac{1}{P(S_1^N)} \int_{\theta_1} \int_{\theta_2} P(S_1^N \,|\, C = c, \Theta_1 = \theta_1, \Theta_2 = \theta_2) \\
&\qquad P(C = c, \Theta_1 = \theta_1, \Theta_2 = \theta_2) \mathrm{d}\theta_2 \mathrm{d}\theta_1 \\
&= \tfrac{1}{P(S_1^N)} P(C = c) \int_{\theta_1} \int_{\theta_2} P(S_1^c \,|\, \Theta_1 = \theta_1) P(\Theta_1 = \theta_1) \\
&\qquad P(S_{c+1}^N \,|\, \Theta_2 = \theta_2) P(\Theta_2 = \theta_2) \; \mathrm{d}\theta_2 \mathrm{d}\theta_1 \\
&= K \int_{\theta_1} P(S_1^c \,|\, \Theta_1 = \theta_1) \mathrm{d}\theta_1 \int_{\theta_2} P(S_{c+1}^N \,|\, \Theta_2 = \theta_2) \mathrm{d}\theta_2 \\
&= K P(S_1^c) P(S_{c+1}^N)
\end{aligned} \tag{4.11}$$

where K is a constant for normalization, and:

$$\begin{aligned}
P(S_1^c) &= \int_0^1 P(S_1^c \,|\, \Theta_1 = \theta) P(\Theta_1 = \theta) \mathrm{d}\theta \\
&= \int_0^1 \theta^{A^c} (1 - \theta)^{B^c} \mathrm{d}\theta \; = \; \gamma(A^c, B^c)
\end{aligned} \tag{4.12}$$

with $A^c = \sum_{i=1}^{c} X_i$ and $B = c - A^c$, being the number of occurrences the binary pixel value of 1 and 0 has appeared in the sequence S_1^c respectively. The integral $\gamma(A^c, B^c)$ can be computed by parts:

$$\begin{aligned}
\gamma(a, b) &= \int_0^1 \theta^a (1 - \theta)^b \mathrm{d}\theta \\
&= \tfrac{b}{a+1} \gamma(a + 1, b - 1)
\end{aligned} \tag{4.13}$$

with $\gamma(a, 0) = \frac{1}{a+1}$. Therefore we have:

$$\gamma(A^c, B^c) = \frac{A^c! B^c!}{(A^c + B^c + 1)!} \tag{4.14}$$

Therefore the probability 4.11 can be expressed by:

$$\begin{aligned} P(C = c \,|\, S_1^N) &= K P(S_1^c) P(S_{c+1}^N) \\ &= K \frac{A^c! B^c!}{(A^c + B^c + 1)!} \times \frac{D^c! E^c!}{(D^c + E^c + 1)!} \end{aligned} \tag{4.15}$$

similar to A^c and B^c defined above, $D^c = \sum_{i=c+1}^{N} X_i$ and $E = (N - c) - D^c$ are the number of occurrences of the binary pixel value of 1 and 0 in the sequence S_{c+1}^N respectively.

4.2.2 Numerical Simulations

For a given value of sequence length N, say 10, we can check the expression of the exact value by computing numerically a randomly generated reference sequence. Given the reference sequence $(x_1, \ldots, x_N)$, we generate enough random draws of $(x_1^{(m)}, \ldots, x_N^{(m)})$ for $1 \leq m \leq M$ so that we have $M \approx 10,000$ samples with exactly the same $(x_1, \ldots, x_N)$ and estimate empirically $\hat{P}(C = c \,|\, X_1 = x_1, \ldots, X_N = x_n) = \hat{\rho}_c$.

We then repeat the process $L \approx 1000$ times to obtain the distribution of the $\hat{\rho}_c^l, l = 1, \ldots, L$.

Fig. 4.2 shows the histogram of the difference between the probability values $\hat{\rho}_c$ computed above with the analytical values of ρ_c calculated using Eq. 4.15. It can be seen that it has a normal distribution with the mean equal to $\mu = 1.66e^{-10} \approx 0$ and the standard deviation of $\sigma = 0.0032$. The probability that error falls within distance of σ from the mean value, μ is ≈ 0.25.

4.2.3 Theoretical Analysis

The statistical behavior of the empirical estimation of the cut probability is related to its analytical solution through the law of large numbers. If we treat each of the possible cut positions $\gamma, 1 \leq \gamma \leq N$, as a random binary variable U_γ, which is 1 if the cut is at position γ and 0 otherwise, then the empirical estimation of the cut probability at position γ is equal to $\hat{P}(C = \gamma \,|\, X_1 = x_1, \ldots, X_N = x_n) =$

$\frac{1}{M}\sum_{i=1}^{M} U_i = \hat{P}_\gamma$. For U_i we have:

$$\begin{cases} P(U_i = 1) = P(C = \gamma \,|\, X_1 = x_1, \ldots, X_N = x_n) = P_\gamma \\ \\ \mathrm{Var}(U_i) = (1 - P_\gamma)P_\gamma \end{cases} \tag{4.16}$$

Therefore, given the fact that the U_i's are *i.i.d.* variables, the law of the large numbers states that $\mathrm{E}(\hat{P}\,(\gamma)) = \mathrm{E}(U_i)$ and $\mathrm{Var}(\hat{P}_\gamma) = \frac{1}{M}\mathrm{Var}(U_i)$. Moreover if we repeat the experiment L times these values do not change due to the *i.i.d.* assumption for the variables. This is in compliance with the observations where the order of magnitude for the standard deviation of the empirical estimation $\sigma = 0.0032$ is the same as its estimated value using Eq. 4.16,

$$\sigma_{\hat{P}_\gamma} = \sqrt{\frac{1}{M}(1 - P_\gamma)P_\gamma} \approx 0.004$$

for $M = 10,000$ and a randomly selected value of $P_\gamma = 0.170$.

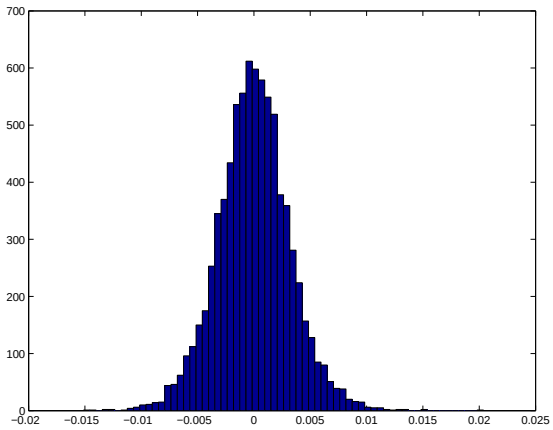


Figure 4.2: Histogram of the values $\rho_i - \hat{\rho}_i^l$ for $i = 1, \ldots, N$ and $l = 1, \ldots, L$.

4.2.4 Performance of the Optimal Estimator

We can estimate empirically the probability to catch the cut by taking the optimal cut ($\arg\max_c P(C = c \,|\, S_1^N)$) and the actual one, or the distribution of the distance between this optimal cut and the true one. Fig. 4.3 shows this distribution for $1,000,000$ tries of generating a random sequence of 10 pixels with a

random cut position uniformly distributed over the sequence and measuring the distance between the true cut and the one given $\arg\max_c P(C = c \,|\, S_1^N)$, where $P(C = c \,|\, S_1^N)$ is calculated using Eq. 4.15.

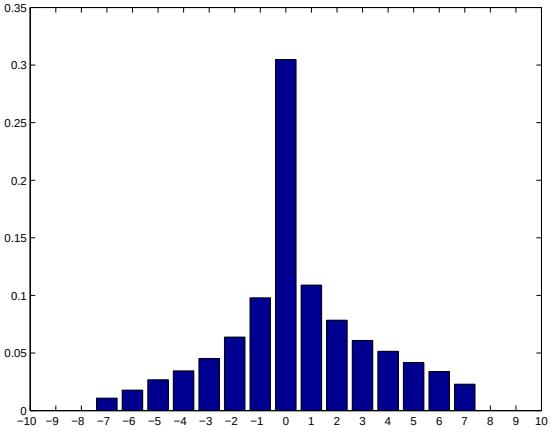


Figure 4.3: Distribution of the distance (in pixels) between the optimal cut and the true one for a sequence of binary bits of length 10.

4.2.5 Fast Computation of the Posterior Cut Probability

We have shown previously in Section 4.1.1 that the probability given by Eq. 4.15 can be calculated rapidly using a recursive formula. Here we restate the formula for the case of the problem stated above. We can write the integral I in Eq. 4.12 as:

$$\begin{aligned} I &= \frac{P(S_1^{c-1})}{P(S_1^{c-1})} \times \int_0^1 P(S_1^c \,|\, \Theta_1 = \theta) P(\Theta_1 = \theta) \mathrm{d}\theta \\ &= P(S_1^{c-1}) \times \frac{\int_0^1 P(S_1^c \,|\, \Theta_1{=}\theta) P(\Theta_1{=}\theta) \mathrm{d}\theta}{\int_0^1 P(S_1^{c-1} \,|\, \Theta_1{=}\theta) P(\Theta_1{=}\theta) \mathrm{d}\theta} \\ &= P(S_1^{c-1}) \times \frac{\int_0^1 P(S_1^c \,|\, \Theta_1{=}\theta) \mathrm{d}\theta}{\int_0^1 P(S_1^{c-1} \,|\, \Theta_1{=}\theta) \mathrm{d}\theta} \\ &= P(S_1^{c-1}) \times \frac{(O_{s_c}{+}1)!(c{-}O_{s_c}{-}1)!/(c{+}1)!}{(O_{s_c})!(c{-}1{-}O_{s_c})!/c!} \\ &= P(S_1^{c-1}) \times \frac{O_{s_c}{+}1}{c{+}1} \end{aligned} \tag{4.17}$$

where O_{s_c} is the number of occurrences of the binary symbol s_c of position c in the sequence S_1^{c-1}. Eq. 4.17 is equivalent to Eq. 4.6 and provides us with a recursive formula which can be implemented easily using Algorithm 1.

4.3 Transition Between Brodatz Textures

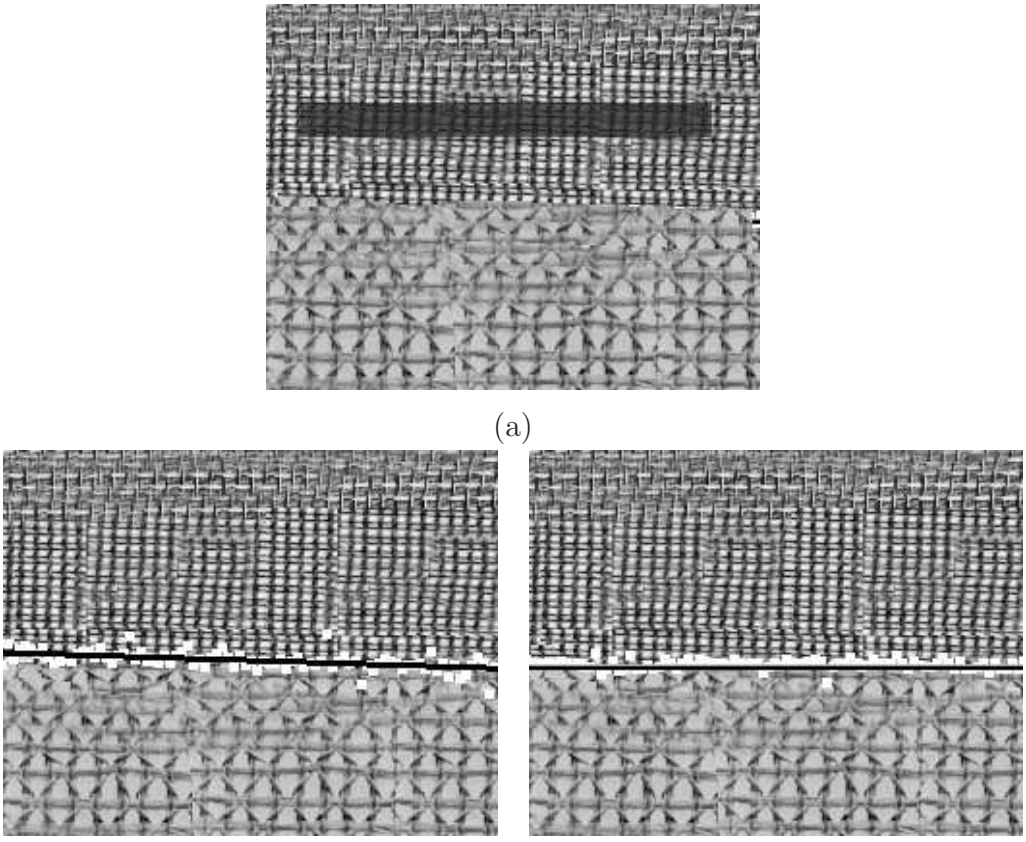
We illustrate these ideas by considering the problem of locating the boundary between two Brodatz textures.

Fig. 4.4 shows detection results in a case where the texture is different at the top and bottom of the image. Marked by a dark stripe in Fig. 4.4-(a) is the region used to generate model statistics for the upper texture. In (b) the boundary between the upper and lower textures is correctly found using 0^{th} order Markov model for both sides. (c) shows the boundary found using 1^{st} order Markov model for both sides. While the model for the lower points is built in an autoregressive manner, the model of the upper points is the one created during an initialization phase (a). White dots show the exact detected points on the boundary along vertical scanlines, to which we robustly fit a line shown in black.

The distribution of the distances from white points to the black line for results of Fig. 4.4 are shown in Fig. 4.5-(d) and (e) for the 0^{th} order and 1^{st} order respectively. As can be noticed the distribution peak is less sharp for the 0^{th} order model than the 1^{st} order. This can hamper the detection of boundaries when the distributions are similar as shown in Fig. 4.6.

An example of texture segmentation for a piece of texture for which we have a geometric model is shown in Fig. 4.7. This is a especially difficult case due to the neighboring texture mixture. Nevertheless, both 0^{th} and 1^{st} order models detect the boundary of the target object (black lines) accurately by robust fitting of the polygonal model to the detected changepoints (white dots). However it was observed that 0^{th} order model is more sensitive to initial conditions and converges more slowly, which can be explained by the fact that the 0^{th} order observations carry less information than the 1^{st} order ones and essentially ignore spatial structure of the texture.

The final example on textures involves the case where there is no a priori model for the texture on either side and is shown in Fig. 4.8. As can be seen

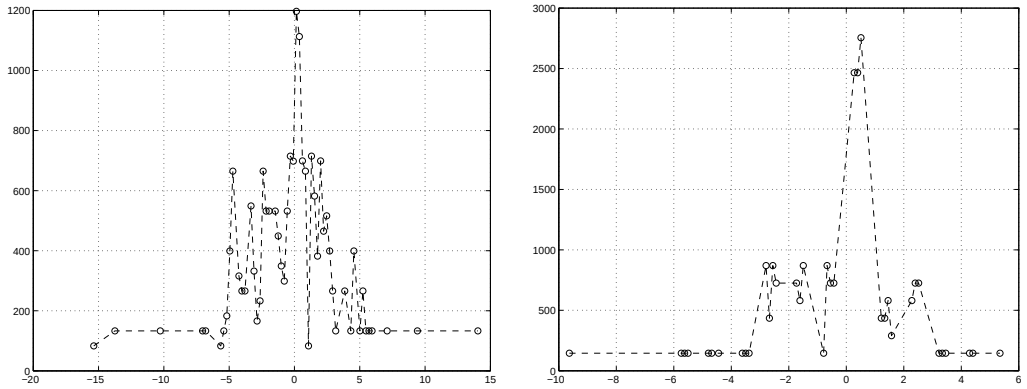


(a)

(b) (c)

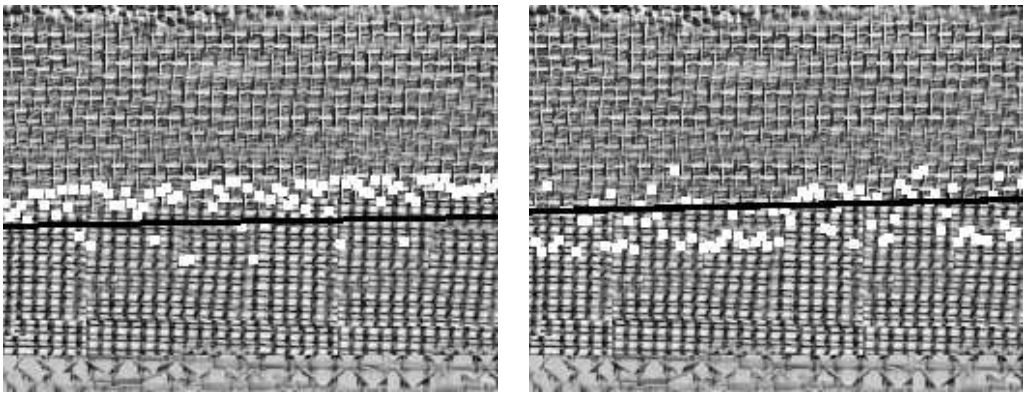
Figure 4.4: Brodatz textures results. (a) texture patch used to learn the target texture model (the dark stripe). This model is used to detect the boundary of the target texture with another texture. (b) and (c) detected boundary using 0^{th} and 1^{st} order model respectively. White dots are the detected change point and the black line is the fitted texture boundary.

the texture boundary can be found without prior model of either of the present textures successfully using the Markov-based scanline search. The performance of this technique is also evaluated quantitatively in Chapter 7.



(a) (b)

Figure 4.5: Analysis of error for results shown in Fig. 4.4: (a) distances from edge in 0^{th} order model and (b) distances from edge in 1^{st} order model.

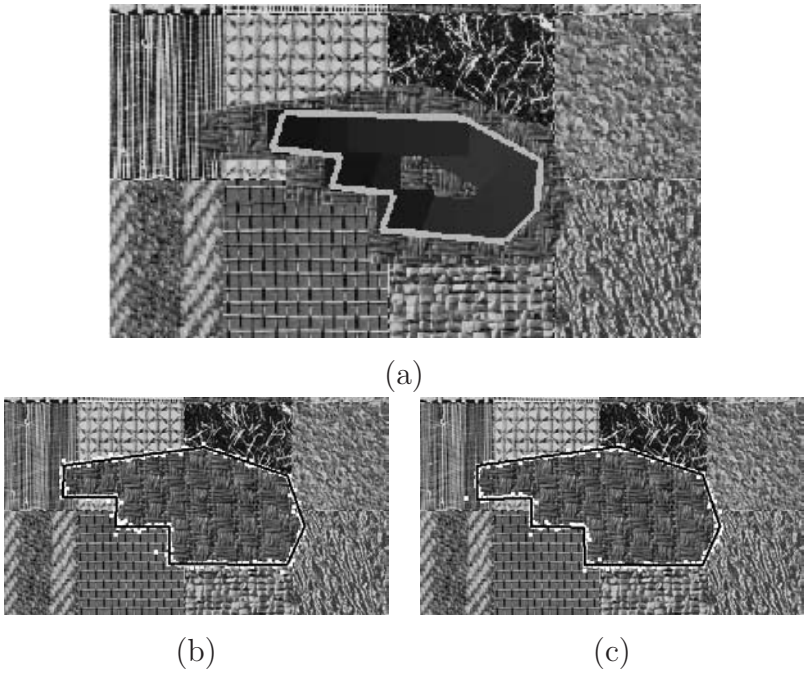


(a) (b)

Figure 4.6: In this challenging case of boundary detection between two similar textures, the results obtained by 0^{th} order model (a) are less precise than the 1^{st} order one (b). As before, white dots are the detected change point and the black line is the fitted texture boundary.

4.4 Scanstripes

In this section we investigate aggregation of the information from multiple scanlines to improve the probability distribution of texture boundaries. We use a bundle of neighboring scanlines, that we refer to as *scanstripes*, in order to better



(a)

(b) (c)

Figure 4.7: Segmentation of a polygonal patch. (a) Initialization: texture patch used to learn the target texture model (the dark region). This model is used to detect the boundary of the target texture with another texture. (b) and (c) show the boundaries detected using 0^{th} and 1^{st} Markov model respectively. 0^{th} order model is more sensitive to the initial conditions. As before, white dots are the detected change point and the black line is the fitted texture boundary.

estimate the transition matrices.

Scanstripes are a set of parallel scanlines that are scanned simultaneously to count the number of pixel occurrences of a substring S_{i-1}^{i} in the scanstripe in two perpendicular directions. This improves the estimation of scanline probabilities by basing the estimation on more observations using the neighboring lines around each scanline. In general this results in earlier convergence of the transition matrix.

One might be tempted to use two transition matrices to represent the pixel joint probabilities in two orthogonal directions, one parallel to the scanline, and one normal to it. Although this might seem to be the right choice in the case of non-isotropic textures, it has the disadvantage of doubling the number of param-

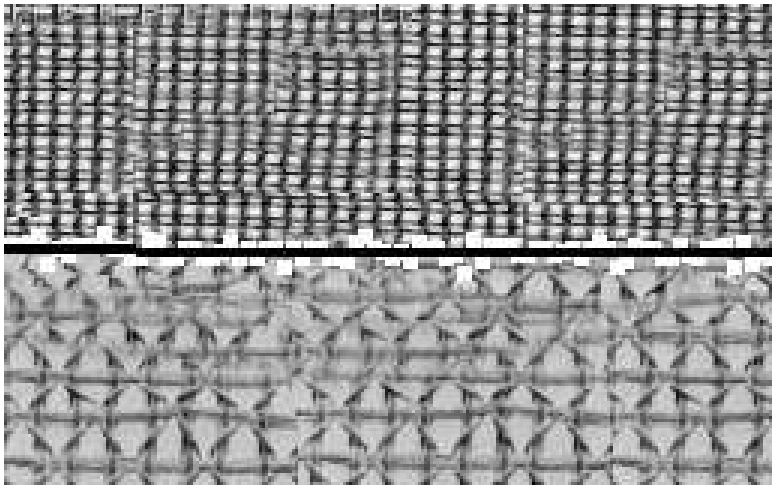


Figure 4.8: Texture boundary detection with no a priori model assumption.

eters used to model the texture. Moreover, in the case for tracking, the initial guess allows rectification of the texture orientation with respect to scanlines and therefore the error in the transition matrix due to anisotropy of texture and changes in its orientation is often negligible. Therefore in practice we use only one transition matrix to keep the number of degrees of freedom small.

To illustrate the effects of scanstripes we have conducted several tests. Fig. 4.9-(b) shows the individual scanlines texture crossing log probabilities on the test image shown in Fig. 4.9-(a) as intensities values where white corresponds to the maximum probability on each scanline and black corresponds to the minimum probability. This depicts the case where we use only one observation at a time to update the transition matrix as suggested in Section 4.1. Fig. 4.9-(c) shows the case where we apply a transition matrix corresponding to the direction parallel to the scanstripe as well as another one for the perpendicular direction on a scanstripe made of five scanlines. Fig. 4.9-(d) shows the case where we use the parallel and normal transitions to update one single transition matrix on a scanstripe made of five scanlines. The superiority of scanstripes over scanlines can be seen in this example where the textures on both sides are quite similar. The result obtained using scanlines are relatively noisier and it is hardly possible to distinguish a texture boundary from the probability image whereas

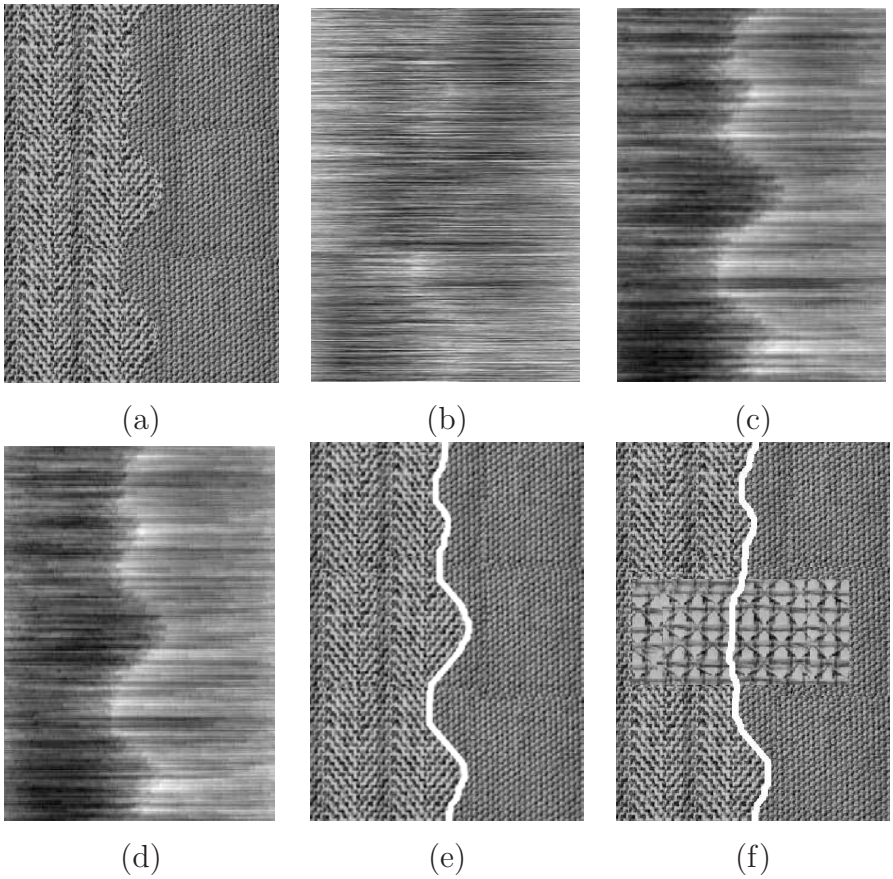


Figure 4.9: Scanstripe vs. scanline log probabilities for the texture image shown in (a). Individual scanlines and a single transition matrix (b). Parallel and vertical transition matrices on scanstripes made of 5 scanlines (c). One single transition matrix containing both parallel and vertical transitions on scanstripes made of 5 scanlines (d). Finally (e) and (f) show examples the extracted boundaries obtained using the HMM approach discussed in Section 6.2 on the test image and in presence of occlusion.

using scanstripes reduces the noise and yields a visible bright boundary (high probability) in the probability image. The rate of convergence of the transition matrix is proportional to the minimum scanline (scanstripe) length required for correct texture boundary estimation. In the above example the convergence of the transition matrix happens 3 times faster when using 5 scanlines to update the transition matrix instead of one scanline.

4.5 Learning a Target Texture Model

It is sometimes helpful to learn the transition matrix of a known target a priori in order to detect its boundary with an arbitrary background. This is particularly useful in tracking or segmentation of complex texture compositions. Unfortunately this is not a trivial issue because the initial values of the transition matrices on both sides of the scanstripes (corresponding to the exterior and interior textures) would no longer be equal. The final changepoint probability in this case will be biased. If the effects of this bias are too large, they can obscure the true peaks in the scanline probability curves and therefore push the detected boundary towards the inside of the learnt target. Moreover, it is useful to measure the consistency of the learnt texture model and the actual object texture during tracking. This measure can be used to update the model as the object's texture and the scene's ambiance are evolving. In the following we propose a method to detect and correct a bad prior model and adaptively update the texture model.

4.5.1 Log Probability and the Entropy of Texture

The log probability curves of a scanstripe have an important mathematical interpretation. We show how this curve relates to the entropy of the texture. This is an important issue since it can be exploited to distinguish the prior models which are not in compliance with the current texture and moreover it gives us a distance measure which can be directly used to update them as discussed in the next section. First we derive the relationship between the log probability curves and entropy for the 0^{th} (histogram) and 1^{st} order (transition matrix) texture model, using the estimated probability of texture symbols using Algorithms 1 and 2.

4.5.1.1 0^{th} Order Model

Imagine that we have C classes in our texture model (bins of intensity histogram for example). The uniform texture distribution prior implies that the probability of a given sequence of n pixels drawn from an unknown texture process T is given

by:

$$\begin{aligned} P(S_1^n) &= \frac{O_{s_1}}{(1+C-1)} \times \frac{O_{s_2}}{(2+C-1)} \times \cdots \times \frac{O_{s_n}}{(n+C-1)} \\ &= \frac{\prod_{i=1}^{C} O_{s_i}!}{\frac{(n+C-1)!}{(C-1)!}} \end{aligned} \tag{4.18}$$

where O_{s_i} is the number of times a class, to which pixel S_i belongs, has appeared in the sequence S_1^i. If n is large enough we have $O_{s_i} = np_i$, with p_i being the probability of class i.

The log probability of this term is therefore:

$$\ln P(S_1^n) = \sum_{i=1}^{C} \ln(np_i)! - \ln(n+C-1)! + \ln(C-1)!$$

Using the Stirling's formula, $\ln x! \approx x \ln x - x + 1$, we get:

$$\begin{aligned} \ln P(S_1^n) \;=\; & \textstyle\sum_{i=1}^{C}(np_i \ln(np_i) - np_i + 1) \\ & -(n+C-1)\ln(n+C-1) \\ & +n \;+\; (C-1)\ln(C-1)\,. \end{aligned}$$

Simplifications give:

$$\begin{aligned} \ln P(S_1^n) \;=\; & -nH - \\ & (n\ln(\tfrac{n+C-1}{n}) + (C-1)\ln(\tfrac{n+C-1}{C-1}) - C) \\ & =\; -nH \;-\; \hat{H}(n)\,. \end{aligned}$$

Thus:

$$H = -(\ln P(S_1^n) + \hat{H}(n))/n \tag{4.19}$$

with H being the entropy of the texture. $\hat{H}(n)$ determines the amount of drift from the real entropy in the sequence as n changes. The above equation suggests that the entropy can be derived from the probability of the sequence. Fig. 4.10 shows the deviation of the sequence log probability $\ln P(S_1^n)$ from the entropy, i.e. $\hat{H}(n)/n$ vs. n. As can be seen this deviation approaches zero for large n. While for small sequence lengths it should be taken into consideration.

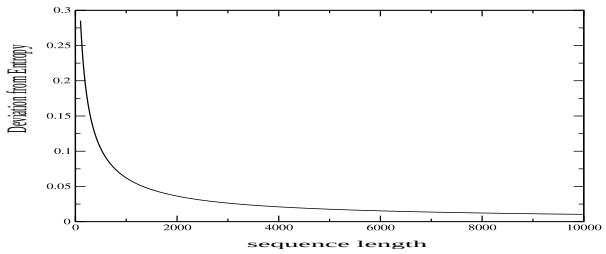


Figure 4.10: $\hat{H}(n)/n$ or the deviation of the sequence log probability $\ln P(S_1^n)$ from the entropy for $C = 16$.

4.5.1.2 1^{st} Order Distribution

The probability of a given sequence drawn from a 1^{st} order Markov texture process T governed by a transition matrix instead of a histogram can be obtained by employing Eq. 4.9 and expanding the sequence probability term. This gives:

$$P(S_1^n) = \frac{\prod_{j=1}^{C}\prod_{i=1}^{C}(1/C + O_{ij} - 1)!}{\prod_{j=1}^{C} O_j!}$$

The log probability of this probability term is therefore:

$$\ln P(S_1^n) = \sum_{j=1}^{C}\sum_{i=1}^{C}\ln(1/C + O_{ij} - 1)! - \sum_{j=1}^{C}\ln O_j!$$

where O_{ij} is the number of times symbol j is followed by symbol i in the sequence of n pixels S_1^n drawn from a texture. Unlike the 0^{th} order it is not straight forward to derive a direct formula for the entropy in this case. However, letting $Oij = np_jp_{ij}$ and $1/C + O_{ij} - 1 \approx O_{ij}$ would make appear the entropy term. These assumptions are not accurate for small n. Nevertheless, they allow us to estimate the relationship between the log probability curve and the 1^{st} order entropy of the texture:

$$H = -(\ln P(S_1^n) + \hat{H})/n$$

with

$$\hat{H} = C^2 - C\,. \tag{4.20}$$

We can see that unlike the 0^{th} order model, the deviation from the entropy, $\hat{H}$, does not increase with n which indicates the advantage of using a transition matrix instead of a histogram.

However, as mentioned earlier the approximation $1/C + O_{ij} - 1 \approx O_{ij}$ is not a good one for small n. Instead, our experiments show that we can approximate the entropy using the equation:

$$H \approx -\ln P(S_1^n)/n - 1 \tag{4.21}$$

for small n and common choices of number of pixel intensity classes $C < 20$.

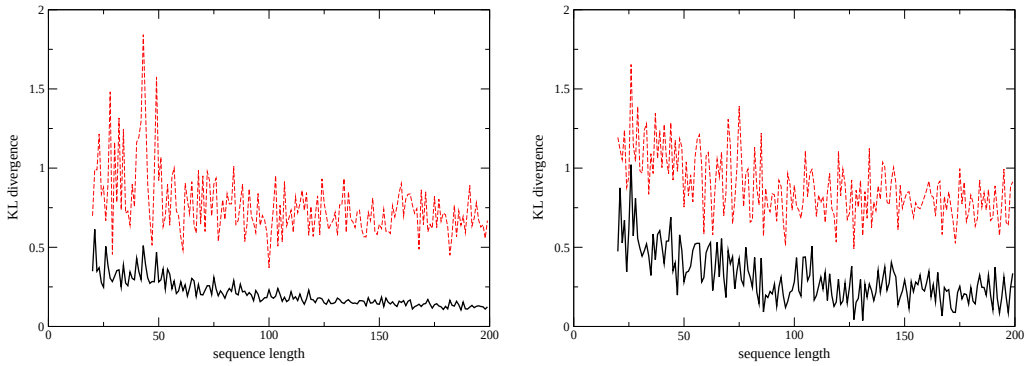
4.5.2 Updating the Texture Model

As the tracking goes on the appearance of the learnt texture of the target changes due to various lighting conditions. Measuring these changes and updating the learnt model is indispensable for a successful tracking. The predicted position given by the previous tracking stage allows us to calculate the entropy of the current target texture as discussed above from the log probability of a sequence of pixels. A second entropy $\tilde{H}$ can be computed from the sequence by using the prior texture model, T_1: $\tilde{H} = -\ln P(S_1^n|T_1)/n$. The difference $\tilde{H} - H$ is the same as Kullback-Leibeler divergence of Eq. C.5 and is always a positive value. The KL Divergence gives a clear measure of how different our calculated model is from the actual texture process. We can use the KL divergence to mix the current texture and the learnt model in order to update the model. In that case we use a filter with a parameter

$$\alpha = 1 - \exp(-\frac{\tilde{H} - H - B}{\tau_c}) \tag{4.22}$$

which depends on the KLD measure. τ_c is the user-defined time constant that determines the latency of the filter. In practice we use a small constant B to compensate the effects of approximations and it is value is determined manually. We set it to be $B \approx 0.4$. The prior model T_1 is thus updated with

$$T_1' = \alpha\, T_2 + (1 - \alpha)\, T_1 \tag{4.23}$$



(a) (b)

Figure 4.11: $\tilde{H} - H$ for the true and a lousy learnt model. (a) 0^{th} order model using Eq. 4.19 to calculate H, (b) 1^{st} order model using Eq. 4.21 to calculate H. Red dashed lines are KL divergence of the poor model and the thick solid line is the KL divergence of the true model.

where T_2 is the current texture model. Fig. 4.11 shows the curves of $\tilde{H} - H$ for the true and a poor learnt model for $20 < n < 200$ derived using Eqs. 4.19 and 4.21 for the 0^{th} and 1^{st} order model respectively. We show in Section 8.2.1 that this model update scheme can be used to improve tracking results of objects with changing appearance due to lighting.

4.6 Summary

Scanline, scanstripes and a novel Markov texture representation are introduces in this chapter which are used to model the exact conditional probability of texture transition given a sequence of pixels. Extensive numerical and qualitative analysis show the effectiveness of this approach in finding the boundaries between complex patterns and textures. The theoretical validity of the algorithm is demonstrated using the Bayesian framework and verified through computer simulations for detection of transitions between binary sequence generator processes. As a result, an algorithm is proposed which can estimate texture transition probabilities across scanlines with low computational cost that is adapted for real-time application. Furthermore, the relationship between the texture entropy and the

conditional texture transition probability is derived and exploited as a tool for updating the dynamic prior texture model during tracking to cope with changes in illumination or object's appearance adaptively.

Chapter 5

Contour Point Classification

The Bayesian formulation presented in Chapter 4 is a novel technique that combines the desirable speed of edge-based line search and the sophistication of Bayes formulation given a small set of observations. However the tracking paradigm involves situations where the given observations are too few for reliable estimation of probability of texture change. An example of such a case for human visual system is illustrated in Fig. A.1 of Appendix A. In this chapter we explore ways to ensure correct decision making about texture boundaries based on very Small images patches. This can be achieved by considering a training dataset containing small patches of blending textures. If the training set contains enough examples to accurately model texture transitions of interest we can construct a predictor that can be used for object boundary tracking. The predictor can then be used on longer scanlines by sliding a small classification window and calculating the score of how well the image area under the classification window corresponds to a patch with a texture transition in the middle as illustrated in Fig. 5.1. Furthermore, the training labels and their associated scores are used to obtain a model of conditional probability of texture transition given the score. The probabilistic notion allows coalescence for scanline information rigorously to extract object contours (white curve in Fig. 5.1).

Given the above incentives, we present a novel approach to texture boundary detection based on supervised learning of texture cuts. We show how trained classifiers can be used to provide a likelihood measure of texture cut for the points in the vicinity of a given contour point. Here, the input to each classifier is

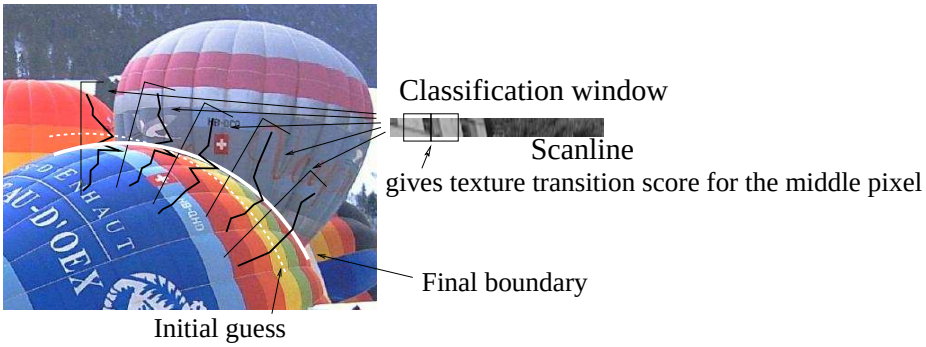


Figure 5.1: Contour extraction by classification of texture transition given an initial curve (dotted curve). Classification score is computed for pixels on scanlines (black lines) by sliding a small classification window along them. The score is then transformed to a probability measure of texture transition that can be coalesced rigorously to extract object contours (white curve).

a narrow image band and the output of each classifier is a binary decision which is '1' if it determines that there is a texture cut in the middle of the band and '0' if otherwise.

We use a set of simple trained classifiers as opposed to a single sophisticated one due to the fact that complex predictors tend to overfit to training data and are also hard to compute. Using a set of classifiers is a popular machine learning technique referred to as *ensemble learning*. In this chapter we first briefly introduce the ensemble learning principles and two major techniques in this category namely, Bagging and Boosting. We choose Boosting technique for training of our classifiers. Each of the final trained classifiers obtained by Boosting method are associated with a weight. The weight of each such classifier is related to its total classification error on the training set. In the remainder of this chapter the details of training dataset and classification features are described and the final classifier is presented and evaluated.

5.1 Ensemble Learning

The ensemble learning is an alternative method to training a single predictor from a hypothesis classifier space, in which several simpler classifiers are trained to make the decision in an aggregated manner. Given the training data set $\mathcal{L} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y} = \{0, 1\}$ a predictor $\varphi(x, \mathcal{L})$ is trained to predict the label y_i of the input vector x_i. The optimum predictor is equal to the unknown function $\mathcal{F}$ which relates input vectors and output labels which are drawn from a (unknown) distribution $\mathcal{D}$. The idea behind learning is to come up with a predictor that is as close as possible to $\mathcal{F}$. The predictor is chosen from a hypothesis classifier space H.

The motivation for ensemble learning is two-fold. The first reason is that, due to the independence of the predictors, decisions based on results of several predictors are obviously more reliable and moreover and more importantly, the hypothesis classifier space does not often contain the function $\mathcal{F}$ therefore a single predictor trained on the training dataset $\mathcal{L}$ could yield unexpected results on the test datasets due to overfitting.

There are different widely used techniques to train several sufficiently diverse predictors and select an aggregation scheme to produce a valid decision as an approximation of $\mathcal{F}$. *Bagging* or Bootstrap AggregatING and its extension *Arcing* or Adaptive Reweighting and CombinING have been introduced by Breiman (Breiman, 1996, 1998). In the rest of this section we briefly introduce Bagging and Boosting methods. We use the latter to train classifiers for silhouette detection.

5.1.1 Bagging

The combination of classifiers is particularly useful in case of instability of the procedure of the hypothesis selection (Breiman, 1996), or the diversity of the classifiers. The idea of the Bagging method is to ensure diversity by using different subsets of the training set for each classifier. Using different features and decorrelation of classifiers during training can also help achieve robust and simple classifiers. At each iteration of the Bagging algorithm, m training samples are selected at random with replacement. The learning algorithm is then trained on

selected examples to generate hypothesis h_t. This process is repeated T times and the final hypothesis is a simple majority vote (denoted by MAJ):

$$H(x) = \text{MAJ}(h_1(x), \ldots, h_T(x)) \ .$$

In summary, Bagging reduces classifier variance by improving unstable classifiers for which small changes in training data leads to significantly different classifiers and large changes in accuracy. On the other hand the drawback of this method is that it does not reduce the classifier bias.

5.1.2 Boosting

The Boosting approach has two major differences with respect to Bagging. Firstly, instead of a random sample of the training data, a weighted sample set is used to focus learning of most difficult examples and secondly, a weighted voting scheme is used instead of simple majority vote used in Bagging algorithm. Here we consider only boosting in the form of AdaBoost which was invented by Freund & Schapire (1996).

Adaboost offers many practical advantages, namely simple and easy to implement with almost no parameters to tune, its theoretically proved efficiency in case of consistency of existence of rules of thumb and finally possibility to use and combine many classifiers (or features). Some of the features used in the literature for training using Adaboost include stumps, decision trees, multi-layer perceptrons, radial basis function etc. Moreover Adaboost can be regularized to reduce the effect of outliers. This is done by introducing a notion of memory in the training so that the penalty terms (weights) for the misclassified samples do not increase unboundedly (Rätsch *et al.*, 1998).

Based on the above discussion, we choose the Adaboost algorithm for training of the weak learners for texture boundary classification together with different features such as mean energy of Fourier coefficients or intensity values as well as cooccurrence features. The Adaboost algorithm integrates these features into a single decision making scheme according to their classification performance.

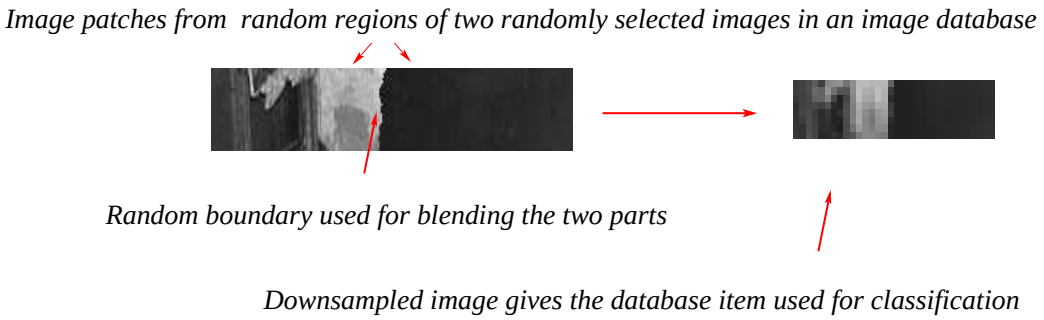


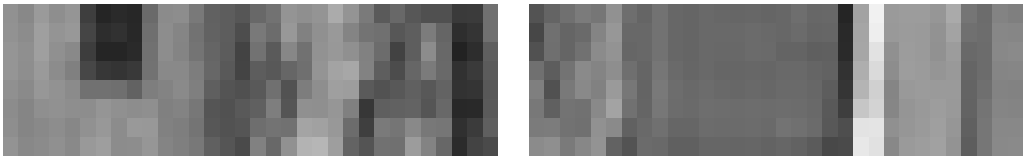
Figure 5.2: The database samples with a texture transition in the middle are made using blending of random image regions and down sampling.

5.2 Database

We build the training database so that it can detect texture transitions even in small image patches. This would lead to a probabilistic model that does not need a large convergence margin as will be discussed later in Chapter 8. Moreover the processing time of each sample remains considerably small. Our set of training examples consists of images of size 32×8 pixels. The positive examples are composed of two randomly selected patches of size 128×32 from random images collected from the web. These patches are concatenated to each other to form an image of size 256×32 with a texture transition in the middle. To produce more realistic texture transitions, the concatenation is done by stochastic blending of the connecting ends of the two patches as illustrated in Fig. 5.2. Finally the results are downsampled to give 32×8 images with a smooth texture transition in the middle. The negative samples contain only one downsampled randomly selected image region. Examples of both positive and negative samples are shown in Fig. 5.3.

5.3 Classifiers

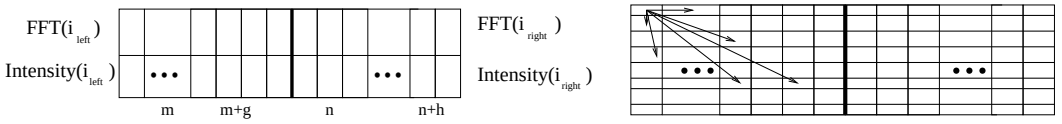
As mentioned earlier the boosting technique has the advantage of offering the possibility to use and combine many features to find weak learners. Three types of features are discussed and used in our weak learners. Namely, we compare mean



(a) positive sample (b) negative sample

Figure 5.3: Examples of (a) positive, and (b) negative training samples used to classify texture cut in the middle of the test image. The database items are 32×8 pixels long.

intensity or frequency coefficients at different size-varying regions or frequency bands on two sides on an image in order to decide whether there is a texture transition in the middle or not. Another type of feature used to distinguish images with varying patterns is comparison of cooccurrence frequencies of pixel intensities in two side. These features are described below.



(a) (b)

Figure 5.4: (a) Intensity and FFT feature parameters defined on the images. (b) Some position operators used to generate cooccurrence features.

5.3.1 Intensity and Frequency Mean Energy

The mean of the different bands on the left and right sides of an image in frequency or intensity domains provide intuitive basis for classification of database images. The morphology of each such weak learner using intensity of frequency features is shown in Fig. 5.4-(a). Therefore, a feature $f(s)$ corresponding to a sequence s of image pixels can be defined by 4 parameters:

$$f(s) = f_{m,g,n,h}(s) = 1/(g+1) \sum_{k=m}^{g} I_l(k) - 1/(h+1) \sum_{k=n}^{h} I_r(k) \qquad (5.1)$$

where I_l and I_r are the intensity values or the magnitude of FFT coefficients of the left and right pixel sequences of the image respectively. The total number of intensity and frequency features is $2 \times [z \times (z+1)/2]^2$ where z equals half of the image length in pixels. Thus for a 32-pixel training image database, there are total of $36,992$ such features.

5.3.2 Cooccurrence Matrix Features

Let M be a $n \times n$ matrix whose element $M[i][j]$ is the number of occurrences of intensity level i in the position specified by a position operator $P(i,j)$ relative to the gray level j. Dividing M with the total number of point pairs that satisfy the operator P yields a $n \times n$ matrix C which is called the cooccurrence matrix defined by P. For a given texture, $C[i][j]$, calculated on an image patch, provides an estimation of the joint probability that a pair of points belonging to that texture satisfying P will have intensity values i and j.

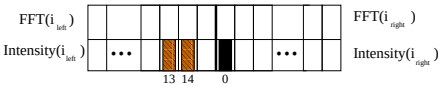
The position operator P can be defined so as to consider any kind of neighborhood. Some examples are shown in Fig. 5.4-b for a given database patch for the left image side. These include, for instance, the position where point i is above j or i is two rows below and five columns to the left of j, etc. During the training phase we consider all possible occurrences within a given neighborhood radius, CM. A neighborhood radius CM includes $CM \times (CM+1)/2$ position operators (excluding symmetrically identical positions). Moreover, each element of these $n \times n$ operators, n is the number of gray levels, can be associated to a weak learner. Consequently, similar to intensity and frequency, cooccurrence features are defined by 4 parameters, i and j gray level indices, and two more parameters to define the operator P. Cooccurrence measures provide a large source of $CM \times (2 * CM - 1) \times n^2$ elements for the training of weak learners which will later be boosted along with other features used. For example $CM = 5$ will create $11,520$ features.

5.4 Training Classifiers

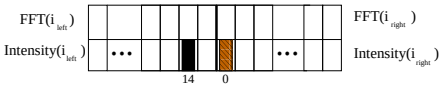
We associate a feature to each classifier and use the Adaboost Algorithm for training them. At each stage r the trained classifier h_r is given a threshold T_r, a parity P_r and a weight w_r, as explained in Section 5.1.2. The output label of such classifier is then given by

$$h_r(s) = \begin{cases} 1 & \texttt{if } P_r\, f(s) > P_r\, T_r \\ 0 & \texttt{otherwise} \end{cases} . \tag{5.2}$$

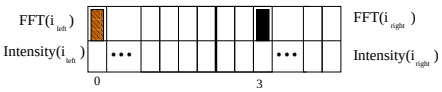
The first four trained classifier using intensity and FFT features are shown in Fig. 5.5. The bars show the indices of pixel on left and right side of a sample image and whether features are in intensity or FFT domain. The hatched bars indicate the parity of the classifier (i.e. P_r is -1 in Eq. 5.2 if the hatched bars are on the left side). We see that the first two classifiers compare intensity values of the last pixels on the left side with the first pixel on the right side with different parities. That basically means that the best way to determine whether there is texture cut in the middle of an image is simply by comparing pixels around the potential cut position. We speculate that the reason why there is one pixel space between the indices compared on both sides in the case of the first two classifiers is that in most edges the two textures tend to blend into each other. The next two classifiers are in FFT domain and compare different frequency bands of the two sides of the image.



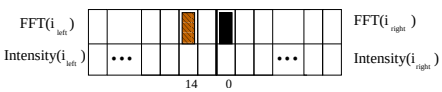
(a) 1^{st} weak learner



(b) 2^{nd} weak learner



(c) 3^{rd} weak learner

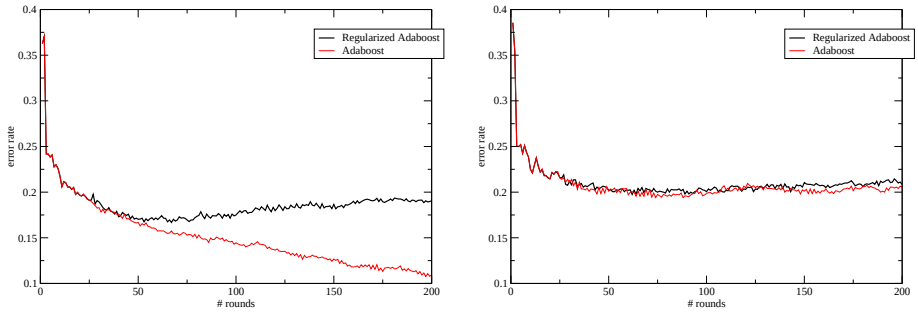


(d) 4^{th} weak learner

Figure 5.5: The first four trained classifiers. The bars show the indices of pixel on left and right side of a sample image and whether features are in intensity or FFT domain. The hatched bars indicate the parity of the classifier.

A boosted set of classifier with R weak learners then gives a score $x_i = \sum_{r=1}^{R} w_r \times h_r(i)$ to input sequence s. Once the classifiers have been trained and boosted using enough training samples, we can check the performance on the training and test databases for different numbers of weak learners. Fig. 5.6 shows the error rate of classifications on training and test sets versus the number of weak learners. The graphs shown correspond to trained classifiers using Adaboost and regularized Adaboost (Rätsch *et al.*, 1998) techniques.

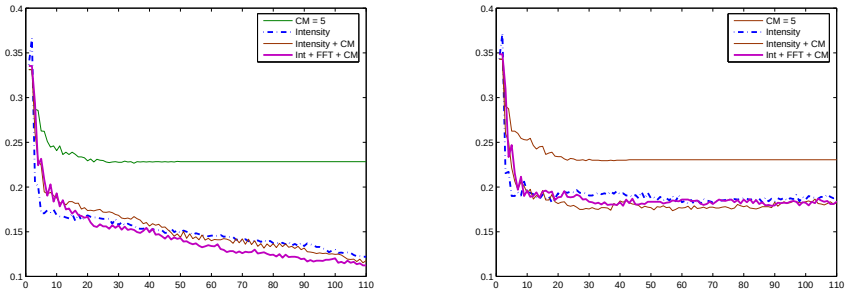
We see that Adaboost error rate decreases on the training set by employing more weak learners while it remains constant on the test dataset after about 50 weak learners due to over-fitting. Fig. 5.6 shows that although the over-fitting is reduced in the case of the training set, no significant improvement is observed on the test set in the case of regularized Adaboost. Therefore, we use non-regularized version of Adaboost algorithm to train the weak classifiers.



(a) training set (b) test set

Figure 5.6: Classification error rate vs. number of weak learners trained using 2000 positive and 2000 negative samples. Adaboost (thin curve) and regularized boosting (thick curve) are used for the training of weak learners. (a) is the error rate on the original training set and (b) is the error rate on a test set of 1000 positive and 1000 negative samples.

Similar evaluations of the performance of the classifier using different feature sets are shown in Fig. 5.7 It can be seen that the error rate versus the number of weak learners decreases almost at the same rate on both training and test sets using (*i*) only intensity, (*ii*) intensity and cooccurrence (intensity + CM), and



(a) training set (b) test set

Figure 5.7: Classification error rate vs. number of weak learners trained using different feature sets.

(*iii*) intensity, cooccurrence and FFT features (Int + CM + FFT). Although using intensity together with cooccurrence and FFT features offers slight advantage in both training and test error rates, computation of the FFT coefficients is not efficient for probabilistic measurements where the classification score is calculated under a sliding window along a scanstripe to estimate the probability of texture cut along the scanstripe as explained later in this chapter. On the other hand, the slight advantage gained by employing CM and intensity over only intensity features has no computational overload due to simplicity of calculation of the cooccurrence features (Once trained there is no need to calculate the whole cooccurrence matrix but only the elements required).

It can also be noted in Fig. 5.7 that using only cooccurrence features is not enough to learn to classify the training set thoroughly and reduce the error to zero. However, error is reduced by over 95% of its final value by using less that 20 or 10 first trained weak learners on the training and test sets respectively. This is interesting and means that we can reduce the computational load by using only a few weak learners without noticeable increase in error.

5.5 Boundary Score vs. Boundary Label

The classifier described so far gives a binary decision about existence of boundary in the middle of a small image patch. The score of the classification carries supporting information regarding this decision. More specifically, we show how to use this score to surpass the binary decision making to selecting the best boundary position in a textured image. To achieve this goal we use a sliding window over the test image and observe the classification score which corresponds to the "boundary score" of the pixel in the middle of image region under the window as it is moved over the image. The maximum score corresponds to the "best boundary" position.

To study the performance of the classifier in detecting the texture boundary in an image as explained above, the following simulation was conducted. Fig. 5.8 shows the distribution of error in pixels of the detected texture transition position from the real texture boundary in 1000 randomly generated images using different numbers of weak learners. The histograms are obtained using only intensity features, comparison between boundary histograms using different features is given in Chapter 7. The images are 256 pixels long and are made by blending two random textures in the middle, therefore we can assume a texture change in the middle of them. The sliding window is 32 pixels long which is the same size as the 4000 used as training examples. The detected boundary corresponds to the position which maximizes the weighted sum of classifiers response. We can see in Fig. 5.8 that as the number of weak learners increases, the peak around zero error gets more prominent and the distribution elsewhere becomes flat. Moreover, it can be deduced again that a small number of weak learners is enough for reliable detection.

5.6 Model of the Conditional Probability

To apply this concept to the prototype tracking framework used through out this thesis, we compute the classifier score using the sliding window along a search direction (i.e. scanline or scanstripe). For each location of such scanline, we obtain a response equal to the weighted sum of weak learners as defined in

Section 5.4. We propose to combine those responses into a probabilistic model by first converting them into conditional probabilities as follows.

At a given location, we denote by Y a random variable standing for the presence of a cut, by S the pixel intensities in the considered neighborhood on the search direction and by X the weighted sum of weak learners at that location. The posterior probability of having a texture transition at that location is thus given by:

$$P(Y = 1 \,|\, S = s) = P(Y = 1|X = x) = \frac{1}{1 + \frac{P(X=x \,|\, Y=0)}{P(X=x \,|\, Y=1)}} \ . \tag{5.3}$$

Under the assumption that $X|Y = 0$ and $X|Y = 1$ both follow normal laws of expectation μ_0 and μ_1 and of same variance σ, we obtain

$$P(Y = 1 \,|\, S = s) = \frac{1}{1+\exp\{-\alpha\,(x-\beta)\}} \tag{5.4}$$

with $\alpha = (\mu_1 - \mu_0)/\sigma^2$ and $\beta = (\mu_1 + \mu_0)/2$, and μ_0, μ_1 and σ estimated with a trivial likelihood maximization. Eq. 5.4 represents the conditional probability of texture transition as a Sigmoid function with the form shown in Fig. 5.9.

Finally, the weighted sum of weak learners X can be seen as an approximation of a log-likelihood $\log \frac{P(Y=1|X)}{P(Y=0|X)}$, similarly to the combination of weak learners in a naive Bayesian predictor (Langley *et al.*, 1992). Thus, the parameters α and β stand for a correcting factor for the dependency between weak learners and the prior log-ratio respectively.

Later we show in Chapter 6 how the object model can be used, in conjunction with the classifier responses around the object outlines, to robustly find silhouette for tracking applications.

5.7 Learning Specific Object Boundaries

Training can be concentrated on a single object when the tracking target is known a priori. In this case the database examples are made using samples of the silhouette of the target which are collected offline. In this section we describe a method for construction of the database and discuss some preliminary results.

We also compare these results with the case where we use the general database of Section 5.2 to find object boundaries.

To create the positive examples of the database, the patches from a single or multiple view of the object are collected along the object boundary. These patches are then downsampled to create multiple levels of resolution for each patch. An example of a single frame used to collect patches and rescaled instances of a single patch are shown in Figs. 5.10 and 5.11. Using multiple scale for each patch makes the training robust to changes in the scale of the object when it is seen from different points of view. The collected and rescaled patches are then concatenated against random patches from arbitrary images to be used for training. Similar to the general training, the negative training examples are random patches selected from arbitrary images.

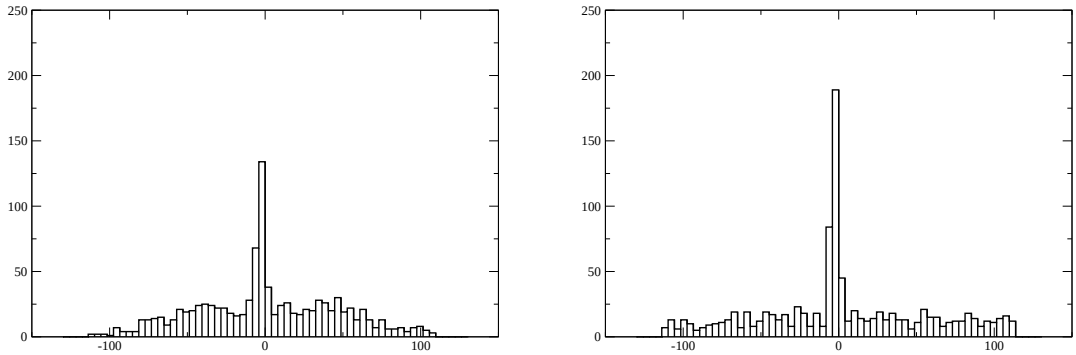
In our preliminary test we trained weak learners composed of intensity and cooccurrence matrix features on an asymmetric database composed of 3000 multi scale examples of borders of the magazine shown in Fig. 5.10 as the positive set and 1000 random patches as the negative set. The training was done using Adaboost. We then used the trained classifier to track the magazine in a sequence. The pose is estimated in 3–D using the frame work explained in Section 2.1 while observations are straight line segments returned by the modified RANSAC approach using the classifier responses as described in Section 6.1.4. The tracking results are shown using the white wire frame in Fig. 5.12 and demonstrate the success of the method even in presence of strong motion blur in the sequence.

In Fig. 5.13 we show the results on the same sequence using the classifier using intensity, Fourier coefficients and cooccurrence matrix elements trained on the general database of Section 5.2. It can be noticed that the results are less accurate with respect to the case where the training is concentrated on the borders of the magazine. In both cases we employ 10 weak learners to calculate the conditional probability of texture cut.

Finally Fig. 5.14 shows more results obtained using 10 weak learners trained on the magazine database in presence of clutter and motion blur. The reported preliminary results show that this method is promising and can be used to learn the silhouettes of specific objects.

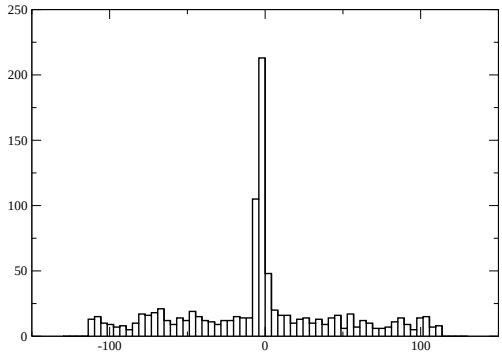
5.8 Summary

We have reviewed two ensemble learning techniques and compared their advantages and drawbacks. Boosting using Adaboost offers versatility and robustness and assures fast convergence and is therefore apt for training weak hypotheses for texture boundary estimation. Adaboost has been employed to train weak learners to classify image patches into those with a texture transition in the middle and those without one. The training can be done for general boundary detection as well as specific object boundary detection simply by changing the positive training examples to contain samples of the target boundary. The weak learners compare intensity, Fourier coefficients or cooccurrence matrix features on two sides of an image patch. Each weak learner is associated with one feature and a weight determined by Adaboost. The final classifier score is the weighted sum of the weak learners' responses. Furthermore, the classification score on the labeled training set is used to fit a Sigmoid function that measures the probability that a pixel belongs to the boundary between two regions. This probabilistic model is essential in robust object outline extraction as discussed in Chapter 6.

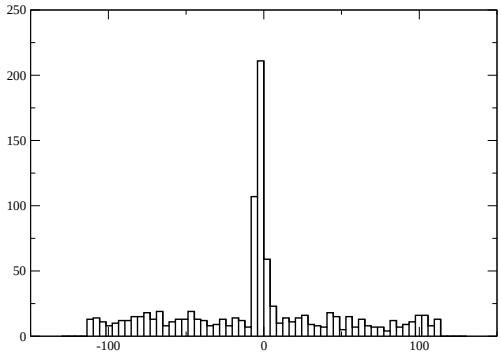


1 weak learners

10 weak learners



50 weak learners



100 weak learners

Figure 5.8: Histograms of texture transition detection error in pixels on 1000 test images using different numbers of trained weak learners. The error decreases with higher number of weak learners.

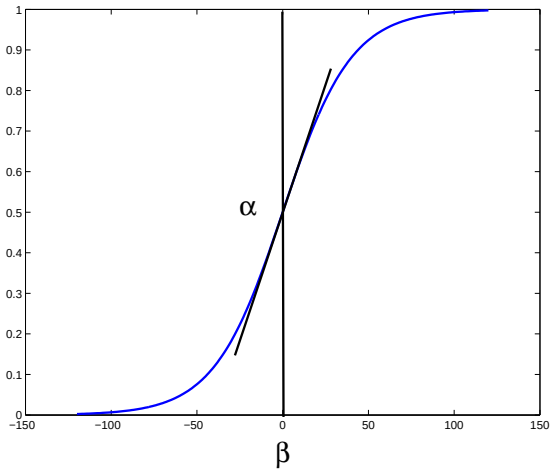


Figure 5.9: The sigmoid function used to model the conditional probability of texture cut classification score.

Figure 5.10: Small patches selected from borders of an object in a single frame are used for training of classifiers on a specific object.

Figure 5.11: Magnified samples of the border of a magazine. The database for learning the borders of a specific object is made using small patches selected from borders of the object in three levels of resolution. These patches are then concatenated against random backgrounds to form the training data.

Figure 5.12: Tracking in motion blur using a classifier trained using the database composed of patches selected from Fig. 5.10. The fitted model is shown by the white wire frame.

Figure 5.13: Tracking in motion blur using the classifier trained using a database composed of general texture boundaries as explained in Section 5.2. The fitted model is shown by the white wire frame.

Figure 5.14: More tracking results with cluttered background and motion blur using the classifier trained on the magazine. The fitted model is shown by the white wire frame.

Chapter 6

Imposing Geometric Constraints

In this chapter we show that a rigid geometric model of the object to be tracked or smoothness constraints in the absence of such a model can be used to coalesce the scanline or scanstripe texture crossing probabilities obtained using the methods described in Chapters 4 and 5.

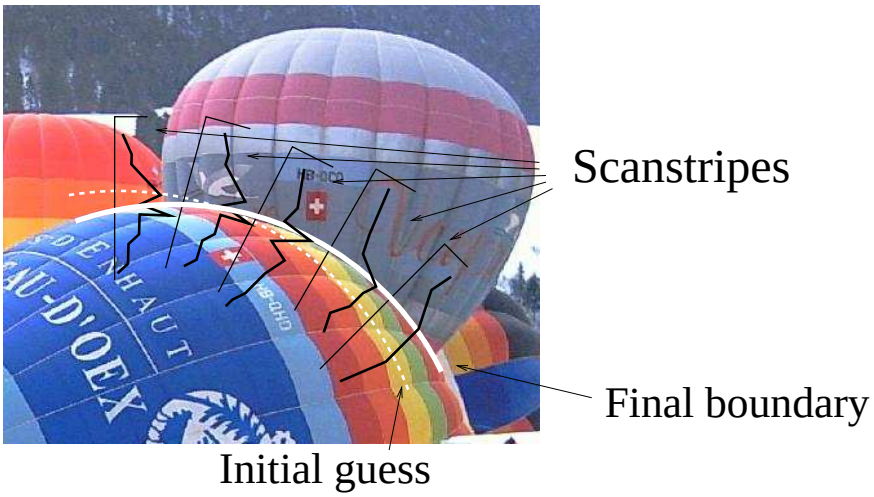


Figure 6.1: Geometric constraints can be exploited to extract object contours (white curve) by coalescing texture transition probabilities of scanlines (black lines) sampled along the object contour using an initial guess (dotted curve).

We propose efficient ways to aggregate robustly the sparse transition probabilities of scanlines sampled along the hypothesis model contour. As a result we

obtain the a posteriori distribution of contour paths that separates the textures in larger regions. This idea is illustrated in Fig. 6.1. Based on the nature of the available constraints we propose to exploit two different methods to coalesce robustly the scanline/scanstripe[1] probabilities.

6.1 Rigid Constraints

Given a geometric (3–D) model of the object, the probability of a hypothesized pose can be calculated from the scanstripe probabilities. Therefore we can use a robust estimator to maximize the pose probability given the scanstripe probabilities.

The RANSAC algorithm introduced by Fischler & Bolles (1981) is a robust model fitting algorithm. The main advantage of the RANSAC algorithm is its robustness to outliers in the observed data. It is used in a wide variety of computer vision applications such as feature matching, registration and detection (Cantzler, 2005) and can provide estimation with a desired probability of correctness given a large enough number of iterations in presence of significant outliers.

6.1.1 Hypotheses Generation

In the original RANSAC algorithm at each iteration input data samples are drawn uniformly and at random. The number of samples drawn is taken to be the minimum needed to compute all model parameters. For example to fit an ellipse to the data, five points are required to calculate the parameters of the ellipse. Using more samples is inefficient as it increases the probability of having samples that do not agree with the true model.

6.1.2 Hypotheses Verification

The degree of compliance of the estimated model parameters with the whole input data is a measure of quality of the model. This measure is referred to as the support of the model and usually corresponds to the number of data points

[1] Without loss of accuracy we use these terms interchangeably.

that agree with the model within a tolerance range. The hypothesis with the maximum support is then selected at the end of the algorithm as the best model fit. Commonly the model computed this way is not very precise and is used as initial value for a gradient descent and least square minimization in order to obtain accurate model parameters. If the data comes from several discrete models, RANSAC algorithm can be used iteratively to determine one model at each time and then compute the next one by removing the data points that agree with the previous model. The vigor of the algorithm lies in the fact that it is likely that at least one set of data samples concurs with the true model and thus derive a correct model.

6.1.3 Algorithm Enhancement

Several crafts can be solicited to improve the efficiency of the algorithm. The size of the data inputs and the probability of having a correct model are the main factors of its speed. The higher the required confidence in the model parameters the more the number of iterations required for convergence. As a matter of fact, it can be shown that given a failure probability, P_{fail}, and the probability that a randomly selected data item is part of the good model, P_g, the required number of iterations is determined by

$$L = \frac{\log(P_{fail})}{\log(1-(P_g)^N)} \ .$$

The effects of the second factor, the input data set size, can be controlled by measuring the support of each hypothesis on a random subset of the database first in order to filter out the models that are far from accurate. Speed of convergence can also be improved by smart selection of samples that are used in hypotheses generation. Since outliers in general follow uniform distribution, whereas inliers are often clustered, it is sometimes helpful to select uniformly a few samples and then select the rest by drawing from their neighborhood. As another alternative, the uniform draw of samples can also be replaced by a prior distribution on the database points. This would privilege drawing samples which are more likely to belong to the model and therefore reach faster convergence.

In the problem of fitting the projection of a geometrically well-defined model to the scanstripe probabilities, the enhanced RANSAC algorithm described above can be efficiently used with scanstripe texture transition probabilities as prior distribution. This idea is discussed in detail below.

6.1.4 Robust Model Fitting to Scanstripes

Figure 6.2: RANSAC fitting of the model. Some drawn hypotheses for the sail pose are shown. Some illustrative scanstripe probabilities which are used to calculate the support of each hypothesis are also depicted in black.

The conditional probability of texture change along scanstripes can be used as prior to draw sample points to which we can fit the projection of the 3–D model. This is illustrated in Fig. 6.2 where a polygonal model is used for tracking the sail of the boat. However, it is not trivial to procure model pose parameters using the samples point. This is due to the fact that establishing correspondences between sampled image points and 3–D points on the model and the pose parameters

involves solving an under-determined nonlinear system of equations to find the pose parameters. Solving this system requires introducing boundary limits of the model edges which leaves it nonetheless a perplexing problem which depends on the geometry of the subject.

As an alternative, we can draw observations for each model edge separately using the modified RANSAC method as illustrated in Fig. 6.3. At each iteration of the modified RANSAC algorithm, first, two scanstripes are selected randomly with uniform probability. Then, two points (shown by circles in Fig. 6.3) are drawn at random to generate a hypothesis for an edge of the model. The support of each line is then calculated by summing the scanline probabilities of the point on each hypothesis line. After enough iterations the line with the highest support is chosen as the observation (the thick line in Fig. 6.3). Pose parameters are then taken to be those that minimize the distance of the model's projection to the selected lines.

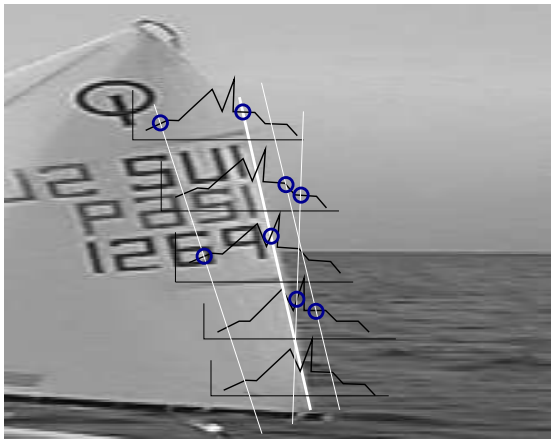


Figure 6.3: RANSAC fitting of the model. Observation generation and calculation of the RANSAC support.

6.2 Smoothness Constraint

When a geometric model is not available or is too complex to deal with, robust stochastic search for the optimum model parameters is not trivial. In this case we

present a Hidden Markov Model (HMM) to relate the responses of the classifier along different lines orthogonal to the candidate edge. This is for example the case when the model does not consist of straight or geometrically well defined edges.

6.2.1 Definition of Hidden Markov Model

The Hidden Markov Model consists of a finite set of states and their associated probability distribution. For each particular state, the probability of going to another state is given by state transition probabilities. The likelihood distribution of observing an output for each state is also part of the model. Usually only the output, not the states are observable and therefore states are referred to as "hidden"; hence the name Hidden Markov Model. The following elements are needed to define completely a Hidden Markov Model (Warakagoda, 1996):

- The number of states of the model, N and the number of output symbols, M.
- A likelihood distribution in each of the states.
- A set of state transition probabilities.
- The initial probability of being in each state.

Some hypotheses are often postulated to facilitate model parametrization and calculations. The assumption of first order Markov process is often associated with HMMs and implies that the state at time t depends only on the state at time $t-1$. Other important assumption in Hidden Markov models are the assumption of stationary state transition probabilities which states that the transition probabilities do not change over time and the independence of the observed output symbols. The last assumption is of limited credibility and might lead to unexpected prediction errors. In terms of application, HMMs are traditionally used in dealing with three types of dilemmas.

1. Decoding problem in which we are looking for the state sequence that maximizes the joint probability of hidden state and observation sequences, given the model parameters. This problem is efficiently solved using the Viterbi algorithm (Forney, 1973) and concerns our application of extracting contours given the search line probabilities as elaborated later in this section.

2. Evaluation problem in which we would like to compute the likelihood of the observation sequence given the model parameters. This problem is efficiently solved using the Forward Algorithm by using the definition of conditional probabilities and by recursion over time (Warakagoda, 1996).

3. Learning problem deals with adjusting model parameters given a model and a sequence of observations. Baum-Welch Algorithm which is based on the EM algorithm is utilized to minimize the output estimation error on the training database (observations) and estimate model parameters simultaneously (Warakagoda, 1996).

6.2.2 HMM and Smooth Silhouettes

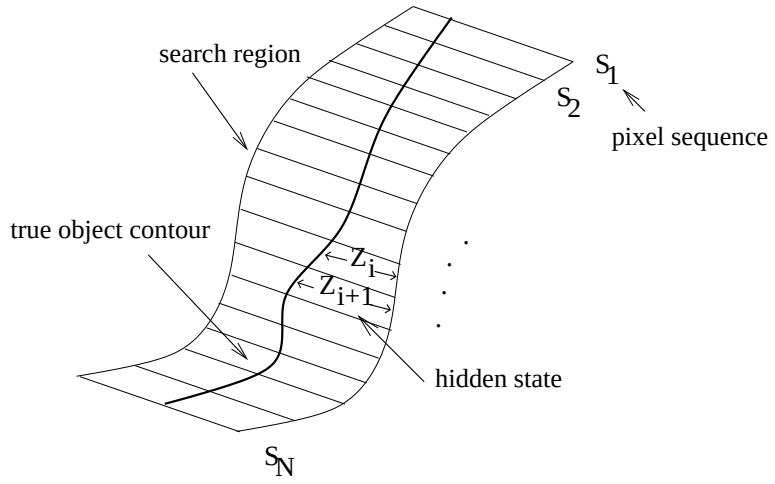
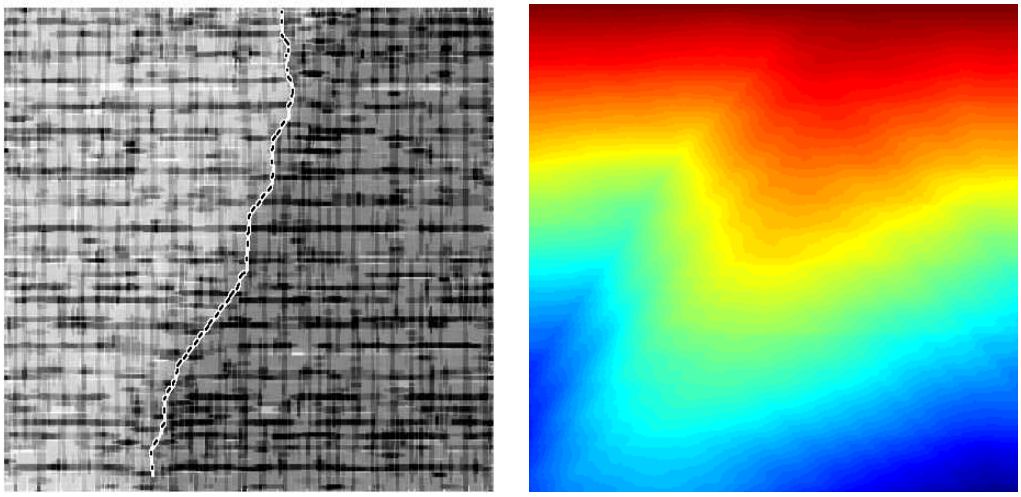


Figure 6.4: Definition of the HMM on the conditional probability classifier responses over a search image.

When the only constraint at hand on the outline of an object is its smoothness and continuity a Hidden Markov model can serve best to extract the optimal contour in terms of likelihood and smoothness by combining separate scanstripe/scanline probabilities. This is the case when a well-defined geometric model of the (tracking) target is not available.



(a) (b)

Figure 6.5: Detected boundary on a test image (a) and the corresponding pseudo color image, (b), showing the HMM probability field on the test image. Red is the highest and blue is the lowest probability value.

We define a HMM with, as observable state, a set of pixel sequences $\mathbb{S} = \{S_i \,|\, i = 1, ..., N\}$ where each sequence S_i has M pixels as illustrated in Fig. 6.4. The HMM is characterized by the visible state S_i which is the sequence of pixel intensities on line i and the hidden state Z_i which is the location of the real edge along the line i. The likelihood distribution in each of the states, $P(S_t \,|\, Z_t)$, is given by individual sequence probabilities of Chapters 4 and 5. The dependency between successive hidden states, $P(Z_{t+1} \,|\, Z_t)$, is modeled by a Gaussian kernel which ensures connectivity and smoothness of the boundary. A typical Gaussian kernel that we use, is centered around zero and has a standard deviation of a few pixels (for example 5). Finally, the initial state distribution $P(Z_0)$ is assumed to be uniform. We wish to maximize the probability of the hidden states for a given

set of sequences, $\mathbb{S}$:

$$\arg \max_{z_1,...,z_N} P(Z_1 = z_1, ..., Z_N = z_N \,|\, S_1, ..., S_N) \tag{6.1}$$

Solving Eq. 6.1 yields the state sequence which corresponds to the most likely contour in the search line and it can be efficiently done by dynamic programming.

The posterior distribution given by Eq. 6.1 can readily be used to obtain texture boundaries using the Viterbi algorithm, which provides an efficient way to extract the highest probability path in the distribution that separates two textures. This idea is illustrated in Fig. 6.5 where the most probable path obtained thus is superposed on the test image and the posterior probability of states, $P(Z_1 = z_1, ..., Z_N = z_N \,|\, S_1, ..., S_N)$, is shown in pseudo color (red is the highest and blue is the lowest value).

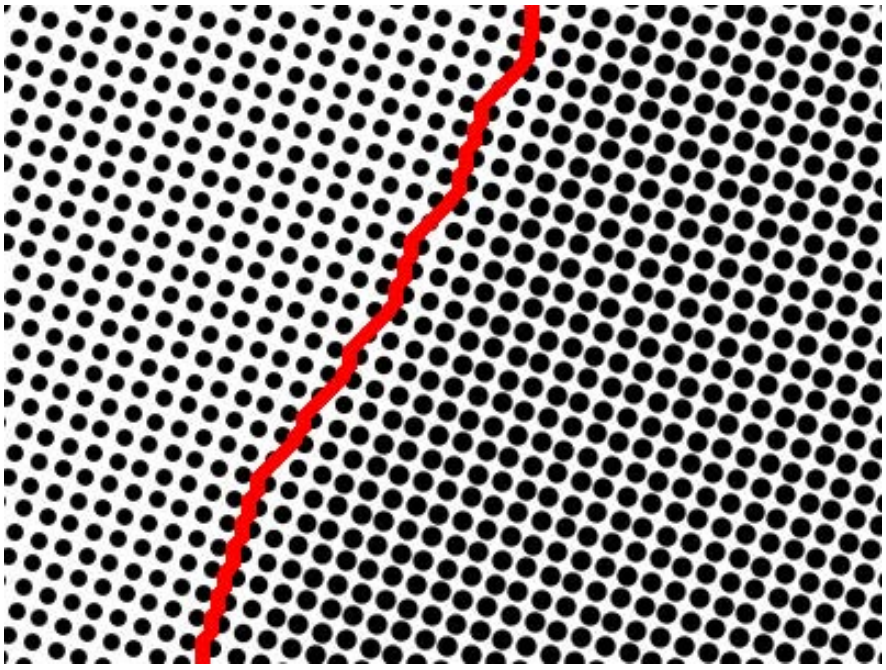


Figure 6.6: Halftone image and the detected boundary using the Viterbi algorithm.

The response of the Viterbi algorithm in a particular case of illusory boundary in a halftone image is also shown in Fig. 6.6. Finally the effects of occluded

boundary in the resulting detected boundary using the Viterbi algorithm are shown in Fig. 6.7. These examples show that the Viterbi algorithm and the presented probabilistic framework can handle robustly the presence of strong partial occlusion.

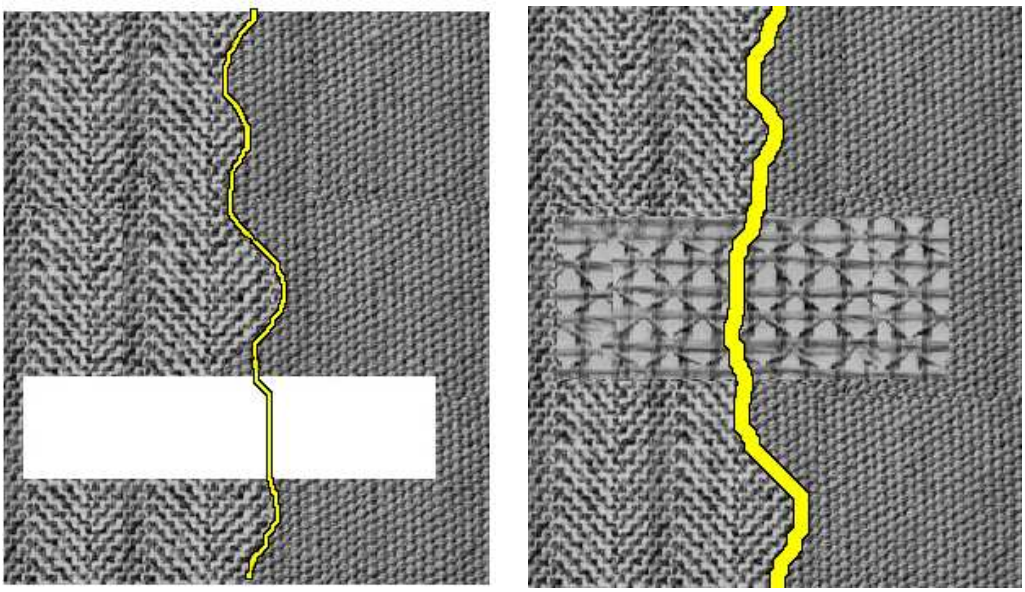


Figure 6.7: Texture boundary detection on occluded patterns using HMM.

6.3 Summary

As a conclusion, we can state that scanstripe information can be reliability and robustly aggregated. While scanstripe probabilities are results of observations concentrated on local patterns, further linkage is required to enforce constraints characterized by the nature of the silhouettes of interest. The selection of the robust linkage using either the HMM or the RANSAC algorithm depends on the available geometric constraints on the object.

The HMM posterior distribution can be used to extract the most probable texture boundary efficiently using the Viterbi algorithm in case of tracking deformable objects or in cases where smoothness and continuity of the outline is the natural constraint.

On the other hand, stochastic robust pose estimation is preferable to find a pose that maximizes the sum of probabilities of contour points of rigid object with a geometrically well-defined model. In terms of implementation it is advantageous to break the model into simple building blocks for which a simpler model and therefore less parameters are needed to be estimated from data points. Furthermore, scanstripe texture transition probabilities can be used as prior distribution for hypotheses generation. Drawing hypotheses according the prior distribution leads to faster convergence towards the most probable observations.

Chapter 7

Evaluation and Comparison

To study the performance of the different methods presented so far, for detection of transition through different textures, we conducted several qualitative and quantitative tests. The methods that we are interested in are scanline and scanstripe boundary detection using the Markov model and Bayesian framework presented in Chapter 4 and the classifier-based method with different features introduced in Chapter 5. In addition to these methods we also consider two other techniques associated with the scanline approach, namely gradient-based and Fisher's metric for the purpose of comparison.

Among other quantitative and qualitative test that will be presented in this chapter, we perform an analysis of the error distribution for each method on a test set. The test set is built in a way to resemble long random scanstripes that can be encountered during tracking of arbitrary objects. It consists of 1000 randomly generated images. The images in the test set are 256×8 pixels and are made by blending of two random patches from various images. The blending is done around the center of the image and therefore we can assume to have a texture change in the middle. The randomness introduced in creation of the test set, makes it a challenging testbed which resembles the natural texture mixture effects as apperceived through images and the eye. some examples of the images in the test set are shown in Fig. 7.1. Note that other texture cuts can appear in the stripes.

The results obtained using different methods are represented in terms of histograms of error in pixel from the known texture crossing point in the test set.

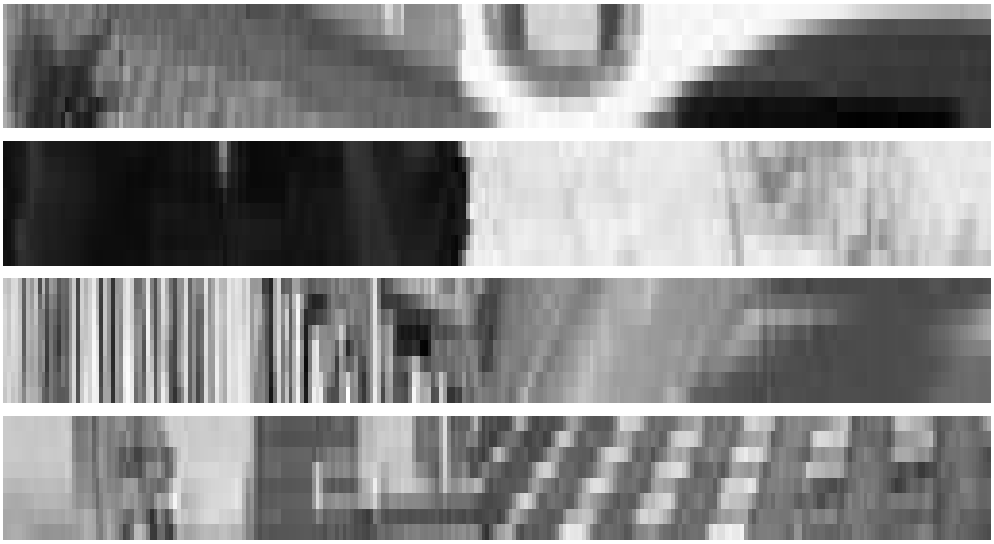


Figure 7.1: Magnified examples of the scanstripes of 256×8 pixels used as test bed for analysis of different methods.

Moreover we provide quantitative measure of performance such as mean error for each distribution and a distance metric to compare these histograms with the ground truth distribution. Finally we present illustrated results of the performance of each method on the test set and other images and also compare the performance of each method in terms of computation time.

7.1 Markov Texture Model

First we consider the transition matrix model representation of texture which is based on a 1^{st} order Markov sequence model as described in Chapter 4. We use and compare scanline method of Section 4.1.2 as well as scanstripe technique described in Section 4.4.

7.1.1 Scanline

The algorithm described in Section 4.1.2 is used with horizontal scanlines to detect the texture crossing position on the test images. Since we can have several scanlines per image, we take the median of the responses of all scanlines as the

texture boundary for each image. Examples of the texture cut positions are shown in Fig. 7.2 for six randomly selected images of the test set. As can be seen, the scanline method performs well and can detect complex texture transitions however it has some difficulty in finding smooth transitions such as the one shown in Fig. 7.2 bottom left and right.

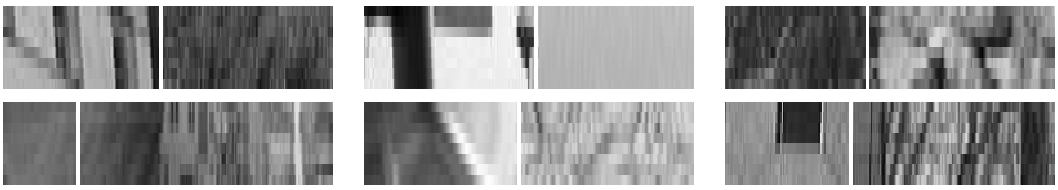


Figure 7.2: Examples of test set images and the detected texture cut positions using scanlines.

Histogram of error of the detected boundary position on the test set images is shown in Fig. 7.4-(a). As could be expected, the error distribution is normal and centered around 0.

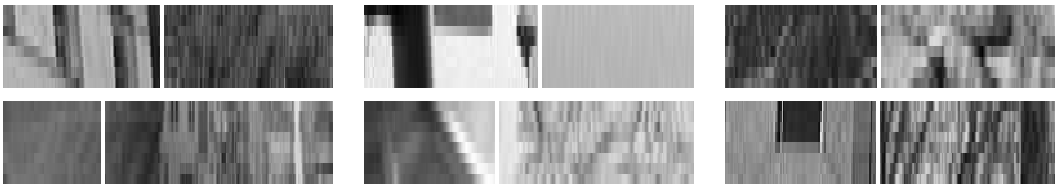


Figure 7.3: Examples of test set images and the detected texture cut positions using scanstripes.

7.1.2 Scanstripe

In Fig. 7.3 we show the detected boundary using a single scanstripe as described in Section 4.4 per image on the same samples of the test set as for the previous part. The error histogram for this method is shown in Fig. 7.4-(b). As can be seen in Fig. 7.4-(b) the error variance is considerably less than what we observed when using scanlines only. Scanstripe cuts shown in the set of examples of Fig. 7.3 are similar but slightly more accurate than those found using scanlines, nevertheless

the problem in finding smooth transition in Fig. 7.3 bottom left persists although the correct cut position is found for the bottom right image. Also in the bottom middle image, the boundary has been taken to be the middle of the narrow band in the middle of the image. This result can be interpreted as a compromise between the two strong boundaries on both sides of the detected one.

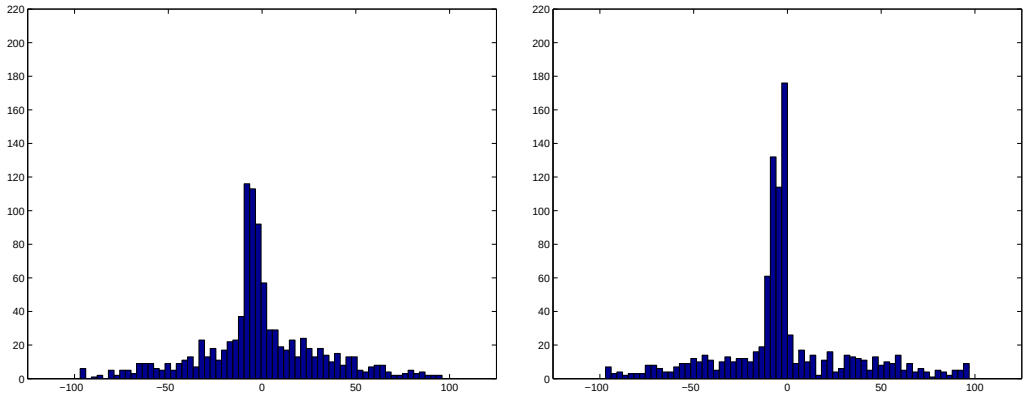
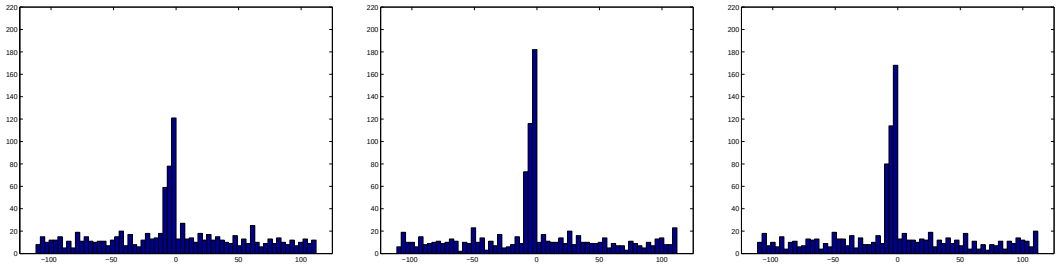


Figure 7.4: Error histogram for Markov texture model using (a) scanline and (b) scanstripes.

7.2 Classification

The classification method for texture boundary detection as described in Chapter 5 is also evaluated on the test set. We use a 32-pixel long sliding window, which has the same size of images used for training of the classifiers, on the 256×8 images of the test set to obtain a classification score as illustrated in Fig. 5.1. The detected boundary position corresponds to the position which maximizes the weighted sum of classifier's responses to the subimage under the sliding window. We analyze different features introduced in Chapter 5 for a classifier with different number of weak learners, namely, 10, 50 or 100. In the following, we evaluate different features described in Section 5.3 and combinations of them on the test set. Similar to the reported histograms in Chapter 5, we start the evaluation

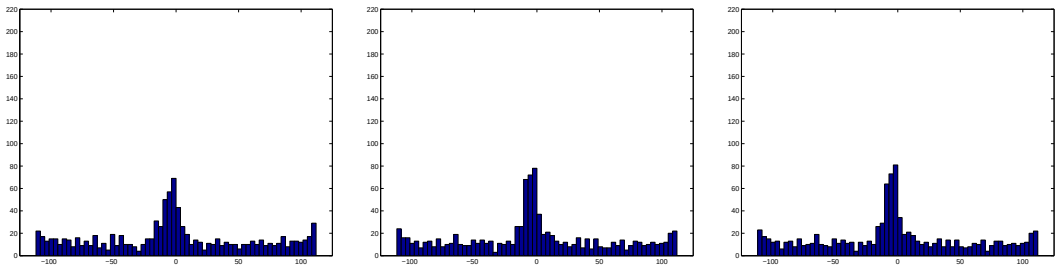
by classification using only intensity features as shown in Fig. 7.5 for different number of weak learners.



(a) (b) (c)

Figure 7.5: Error histogram for classification using intensity features (a) 10 weak learners (b) 50 weak learners and (c) 100 weak learners.

Similarly, using trained classifiers made of cooccurrence matrix features to classify the test set results in the error histograms shown in Fig. 7.6. Here we consider the maximum cooccurrence neighborhood size, denoted by CM, or the maximum radius of the position operator of cooccurrence matrices, to be 5 pixels.

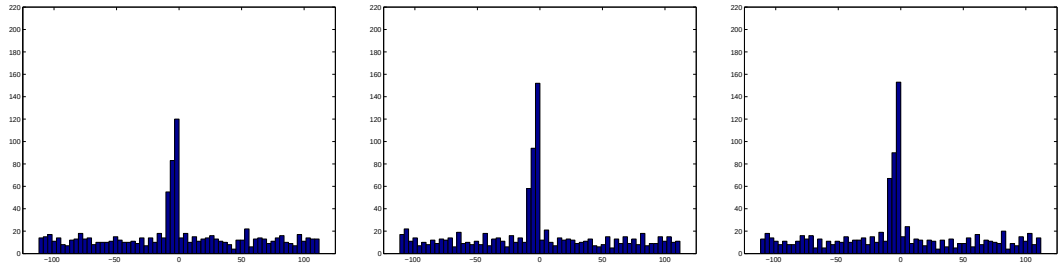


10 weak learners 50 weak learners 100 weak learners

Figure 7.6: Error histogram for classification using cooccurrence matrix features with varying number of weak learners. The neighborhood radius used is 5 pixels (CM = 5).

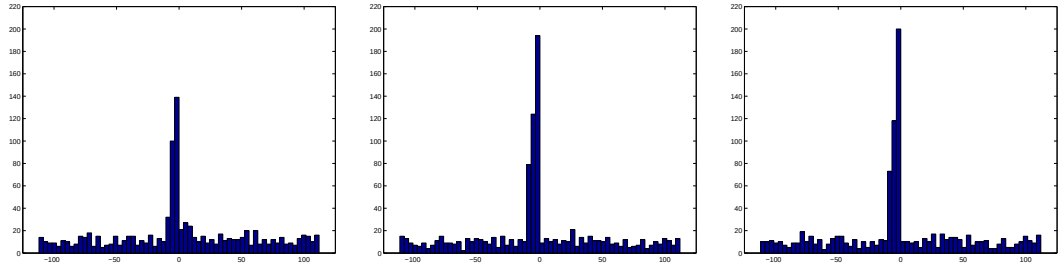
The results obtained by combination of features are shown in Fig. 7.7 for intensity and cooccurrence matrix features with neighborhood of size 5 and in

Fig. 7.8 for intensity, cooccurrence matrix features with neighborhood of size 5 and Fourier transform coefficients.



(a) (b) (c)

Figure 7.7: Error histogram for classification using intensity and cooccurrence matrix features using (a) 10 weak learners (b) 50 weak learners and (c) 100 weak learners.



(a) (b) (c)

Figure 7.8: Error histogram for classification using intensity, cooccurrence matrix and Fourier transform coefficients features using (a) 10 weak learners (b) 50 weak learners and (c) 100 weak learners.

In general, it can be seen that as the number of weak learners increases, the peak around zero error gets more prominent and the distribution elsewhere becomes flat. Moreover, it can be noted that a small number of weak learners is enough for an acceptably high chance of correct detection. Therefore, a small number of weak learners (< 10) with a probability of success as high as 70 percent in conjunction with a robust method of contour extraction, as described in Chapter 6, is usually enough for most applications. Visual inspection of the

histograms further reveals that the best results are obtained by combination of different types of features. On the other hand Fourier transform coefficients do not hand over remarkable improvements to compensate for their heavy computational cost.

For the purpose of illustration the cut positions on the same set of examples as before obtained using the combination of intensity and cooccurrence matrix features are shown in Figs. 7.9, 7.10 and 7.11 using 10, 50 and 100 weak learners respectively. Some cuts for the examples shown are placed in unexpected positioned by classifiers. This effect is speculated to be due to some training dataset examples which might resemble the pattern that has been selected for a cut.

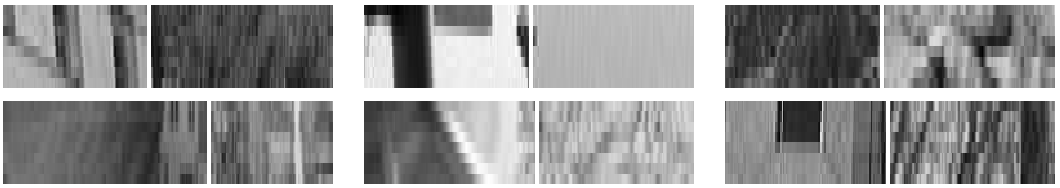


Figure 7.9: Examples of test set images and the detected texture cut positions using the classifier with 10 weak learners.

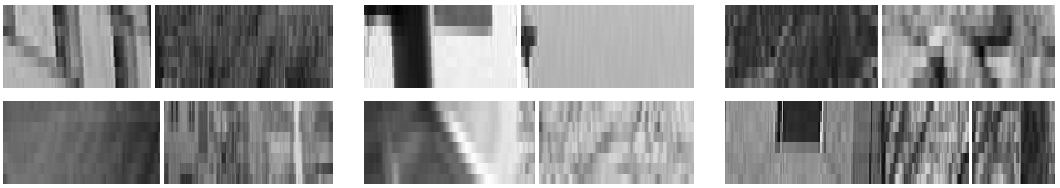


Figure 7.10: Examples of test set images and the detected texture cut positions using the classifier with 50 weak learners.

7.3 Gaussian Texture Model

Fisher's linear discriminant function provides a linearly optimal decision boundary for two classes that simultaneously maximizes the between class variance while minimizing the within class variance of intensity values thus detecting a

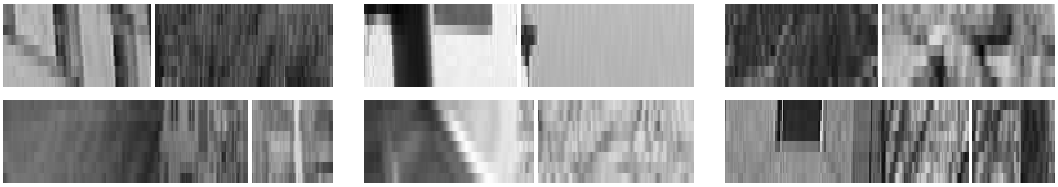


Figure 7.11: Examples of test set images and the detected texture cut positions using the classifier with 100 weak learners.

transition between patterns or classes. It can be shown that Fisher's function is equivalent to the Bayesian solution if the two classes have normal distributions with equal covariance matrix. We can therefore assume normal distribution for textures on both sides of a texture cut and measure the Fisher's score given by Eq. C.1. The best texture cut position according to Fisher's metric is then the position for which the Fisher's score is maximum. Furthermore we can normalize this score in order to obtain a distribution which can be used in conjunction with the Viterbi algorithm in order to extract the maximum a posteriori path separating two textures through out a search region. The histogram of error on the test set along with some example of the detected cut positions are shown in Figs. 7.12 and 7.13 respectively. It can be seen that error distribution has two peaks at the extremities of the histogram. This error accumulation is due to the premature convergence of the Gaussian model on one side of the scanline which can produce wrong local maxima. This means that the Fisher's metric requires a longer margin for convergence with respect to other techniques.

The results obtained using Fisher's discriminant are reliable as long as the assumption of Gaussian distribution for the two textures is not violated as shown in Fig. 7.13. This is further illustrated in Fig. 7.14 where the normal distribution yields a good relative approximation of the textures on both side of the detected boundaries. In Fig. 7.14-left, the texture of the grass can be well approximated with a normal distribution. Moreover, in the right hand side case, the stripes of the zebra are almost parallel to the search direction and thus follow normal distribution (corresponding to white or black stripes).

On the other hand presence of non-Gaussian patterns or textures causes the Fisher classifier to fail as shown in the two examples of Fig. 7.15. In Fig. 7.15-

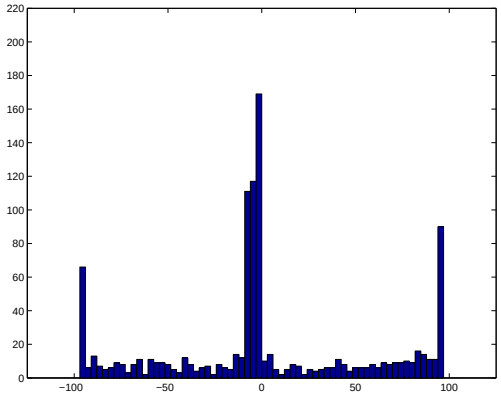


Figure 7.12: Error histogram for Fisher's discriminant metric.

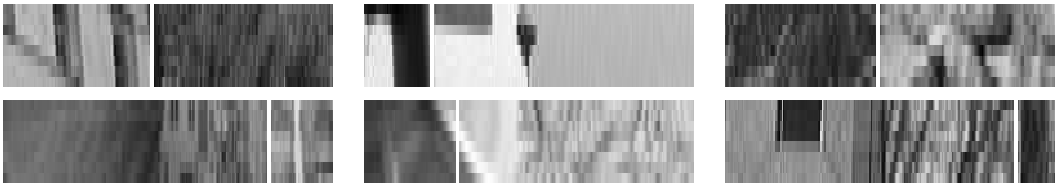


Figure 7.13: Examples of test set images and the detected texture cut positions using Fisher's discriminant metric.

(a) the border between the farthest zebra and the grass is misdetected due to the stripes of the zebra while in (b) the algorithm fails to correctly detect the boundary between the two zebras. These results are obtained using Fisher's metric and the Viterbi algorithm and reveal the limits of the Gaussian texture model assumption of the Fisher's metric method.

For the purpose of comparison. We apply the Markov texture model and scanstripes method on the boundary of the two zebras for which the Fisher's metric had difficulties as shown in Fig. 7.15-(b). The results are shown in Fig. 7.16 and demonstrate the capability of the Markov model method to detect boundaries of similar textures. In Fig. 7.16-(a) the detected boundary points on individual scanlines are shown while the final boundary obtained by the Viterbi algorithm is shown in Fig. 7.16-(b).

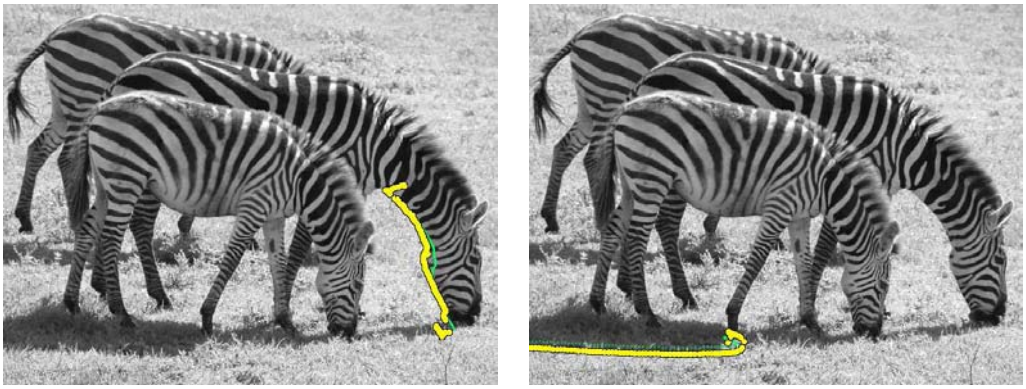


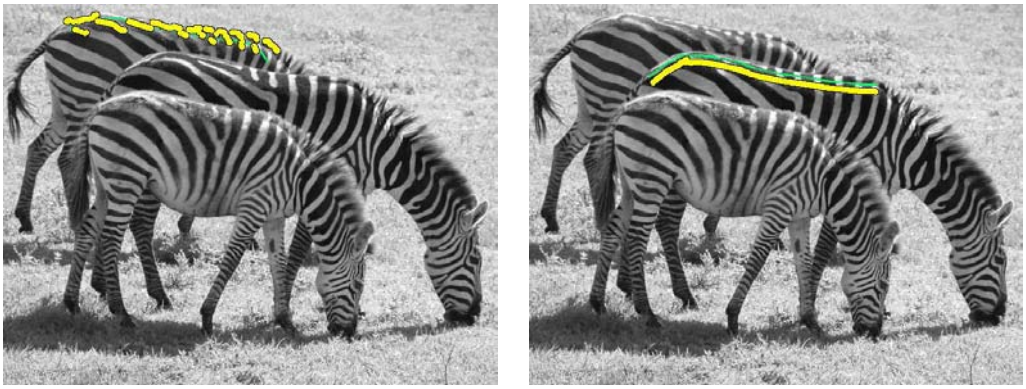
Figure 7.14: Fisher's discriminant-based classification for texture boundary detection.

7.4 Probabilistic Gradient Edge Detection

For the purpose of exhaustivity of analysis we consider also simple gradient measure as a technique to find the texture crossings. We refer to this method as probabilistic gradient measure because we associate a probability of texture transition at each pixel which is exponentially related to the unidirectional gradient value at that point. We then normalize all the values of exponents so that the values sum to one on the image, therefore yielding a normalized distribution, which is useful when combined with robust techniques of Chapter 6. The histogram of the error on the test set for this technique is shown in Fig. 7.17 and examples of cut positions are shown in Fig. 7.18. The asymmetry observed in the histogram shape is conjectured to be specific to the database samples which occur to contain stronger edges in their left side. Based on these observations it can be discerned that although the histogram is well centered around zero error and the method is simple and fast, the results are not reliable for random textures and patterns.

7.5 Tracking Performance

For the tracking applications, we tried different texture boundary detection methods in a unique tracking systems. The methods compared are gradient-based



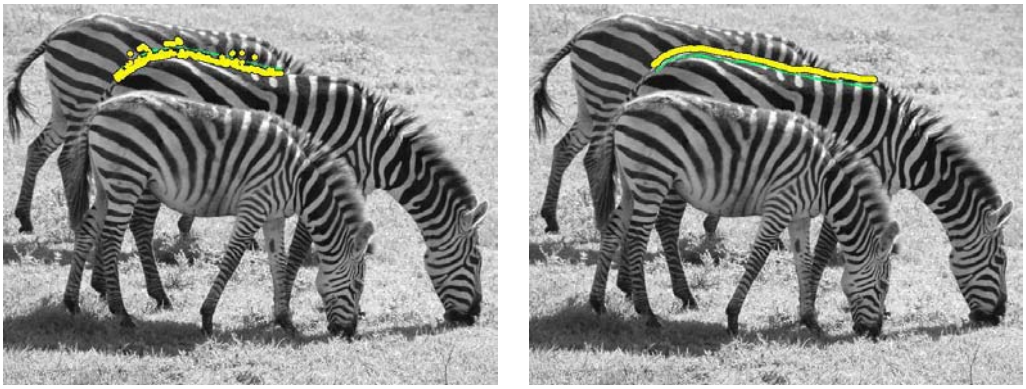
(a) (b)

Figure 7.15: Two examples where Fisher's discriminant-based classification fails to detect the true texture boundary. In (a) we wish to find the border between the farthest zebra and the grass while in (b) we are looking for the boundary between the two zebras.

tracker (Drummond & Cipolla, 2002), Fisher's metric, classifier- and Markov-based methods. The texture crossing probabilities obtained are then used with the RANSAC algorithm for robust tracking as explained in Chapter 6. The results obtained on a short sequence are shown in Fig. 7.19 and are discussed below.

The gradient-based tracker proposed by Drummond & Cipolla (2002) starts from the estimated projection of a 3–D object model and performs a line search in the direction perpendicular to the projected edges to find the boundary location as explained in Chapter 2. Pose parameters are then taken to be those that minimize the distance of the model's projection to those estimated locations. The corresponding implementation is fast and works well when the target object stands out clearly against the background. However it tends to fail for textured objects whose boundaries are hard to detect unambiguously using conventional gradient-based techniques, as shown in the first row of Fig. 7.19.

Fisher's discriminant analysis has been used successfully in applications such as lip tracking (Kaucic & Blake, 1998) in a scanline search framework similar the one used in this work. However, as can be expected from its definition, Fisher's criterion for edge-based tracking works well only when the assumption of two classes with normal distribution is valid around the object boundary. The



(a) (b)

Figure 7.16: Boundary detected by Markov model and scanstripes corresponding to Fig. 7.15-(b). In (a) We use the Markov model technique on independent scanlines while in (b) we use the scanline results in conjunction with the Viterbi algorithm to extract a continuous boundary between the zebras.

tracking results are shown in the second row of Fig. 7.19, where some of the detected edges are drifted from the real ones.

Scanstripe (scanline) Markov texture method is fast and adapted for real-time tracking. However, correct estimation of texture distribution relies on a relatively long scanline in order for the transition matrices to converge. The 3^{rd} row in Fig. 7.19 shows tracking results using this method.

Using the classification technique of texture boundary detection described in Chapter 5 reduces the length of the search line required to almost half (60 pixels) and yields good tracking results as shown in the last row of Fig. 7.19.

In another tracking experiment, we compare the results of 3–D tracking of a chair in a video sequence using the gradient-based method of Section 2.1 and the Markov model and scanline technique proposed in Section 4.1. The use of the latter technique improves the tracking results as shown in Fig. 7.20 while keeping the computational costs almost unchanged (tracking runs at 15 fps).

To further verify the chair tracking quality we plot in Fig. 7.21 the derived 3–D trajectory of the motion of the center of gravity of the model recovered using the texture-based method. The trajectory is given by the null space of the linear transformation matrix computed by the tracker at each frame. Since the chair

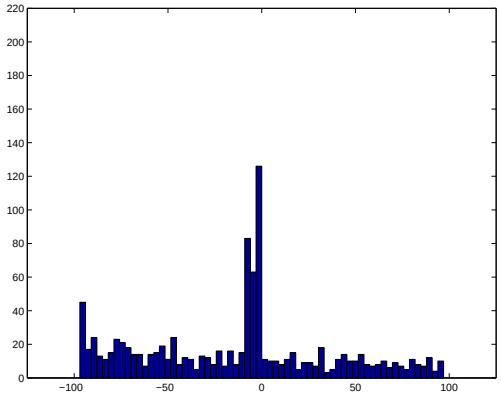


Figure 7.17: Error histogram for probabilistic gradient edge detection.

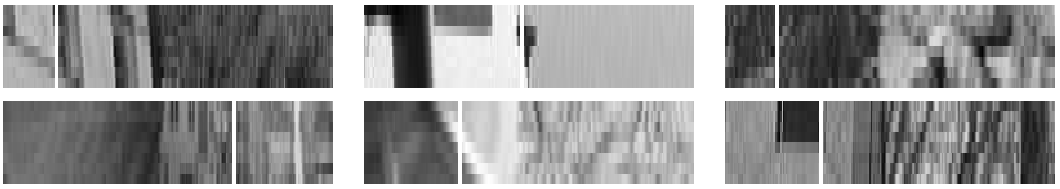


Figure 7.18: Examples of test set images and the detected texture cut positions using the probabilistic gradient method.

remains on the ground, its true motion is indeed planar. The tracker however is not equipped with such constraint and works with six degrees of freedom, three rotations and three translations. The fact that the recovered motion is also planar is therefore a good indication that the Markov-based 3–D tracking using silhouette information is accurate.

7.6 Quantitative Analysis of the Error Histograms

In this section we strive to extract useful information from the error histograms presented earlier in this chapter in order to better understand the characteristics of different methods of texture boundary detection. Intuitively we start by measuring the absolute value of error on the mean cut position for different methods as reflected in table 7.6. It can be seen that the gradient-based method has the

Figure 7.19: Comparison between different contour tracking algorithms. First row: Tracking results using an edge-based tracker. Second row: Tracking results using Fisher discriminant function. Third row: Tracking results using Markov scanstripe texture boundary detection and fourth row: Tracking results using classifier based method.

biggest mean error of almost 6 times higher than other methods. It can also be noticed that using fewer weak learners reduced the mean error in classifier-based detector. Finally scanline method has a lower mean error with respect to scanstripe version and is second best method in terms of mean error on cut position.

Although the mean error measure is to some extent demonstrative, it suffers

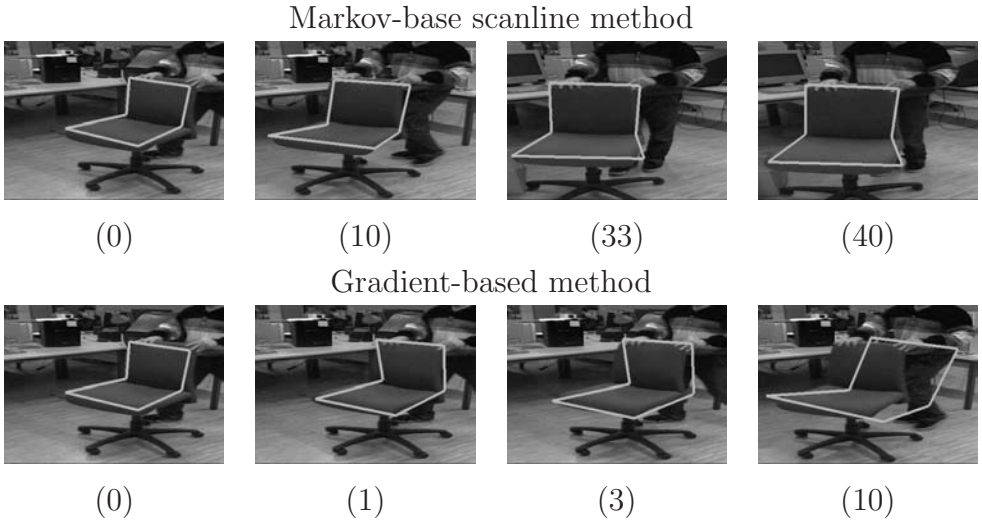
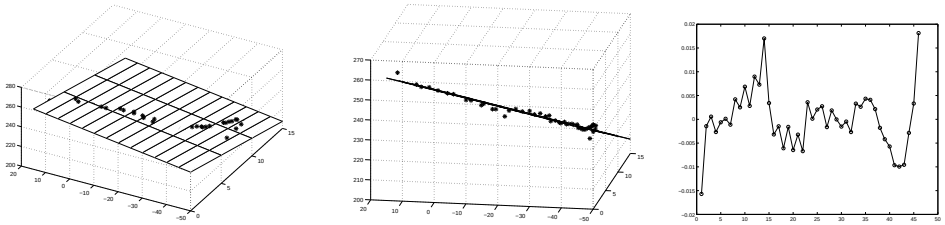


Figure 7.20: Tracking a chair using a simple model made of two perpendicular planes. Top row: Using the Markov texture model and scanline method, the chair is properly tracked throughout the sequence. Bottom row: Using a gradient-based method to detect contours, the tracker starts being imprecise after the 3^{rd} frame and fails completely thereafter. The numbers shown indicate the frame number.

from loss of large amount of information relative to higher moments contained in the histogram. Therefore we need to look for a way to measure quantitatively the similarity of the histograms obtained using different methods with the ground truth response on test set. The ground truth histogram basically states that all the 1000 images of the test set have the texture cut in the center of the image (corresponding to zero error). This is what we investigate next.

7.6.1 Earth Mover's Distance

A distance measure between two distributions that is a true metric and moreover, corresponds to human perception is an ideal means of evaluation and analysis of different methods. While the distance measures discussed in Appendix C are widely used in specific domains, most of them fail to fulfill the metric and perceptual similarity criteria. For example, KL divergence measures the average inefficiency of coding one distribution using the other but it considers only cor-



(a) (b) (c)

Figure 7.21: Plotting the motion of the center of gravity of the chair of Fig. 7.20. (a) and (b): Top and side views of the plane fitted to the recovered positions of the center of gravity. (c): Deviations of trajectory points from the plane, which are very small (all measurements are in mm).

respondences between bins with the same index. This is also true for statistical measures such as χ^2 or Minkowski form distance L_n defined in Appendix C. On the other hand, cross-bin dissimilarity measures such as quadric form of Eq. C.6 can be used as metrics but they fail to fulfill the perceptual similarly criterion (Rubner *et al.*, 2001).

Consequently, we chose the Earth Mover's Distance (EMD) proposed by Rubner *et al.* (2001) and defined in Appendix C to measure the dissimilarity between the presented error histograms and the ground truth. This measure corresponds to the least amount of work that is needed to transform a given histogram to the other. The most important advantage of EMD is that it matches perceptual similarity better than other distances and it can be made a true metric with a proper definition of "ground distance" as a metric (Rubner *et al.*, 2001). The EMD measures with respect to the ground truth for different methods are summarized in table 7.2.

The results show that adding more weak learners in the classification-based method reduces the EMD error distance. It can also be noted that in terms of EMD, the Markov scanline and scanstripe outperform other methods followed with a considerable gap by other methods including classification-based, Fisher's and gradient-based. The relatively high EMD error of classification-based approach with respect to the Markov methods can be explained by considering the shape of the histograms obtained using these techniques. The classification-based

histograms can be approximated with a sharp Gaussian plus a uniform distribution while the Markov-model-based histograms correspond more to a single normal distribution. The uniform component of the classification based histograms is the main reason for their high EMD value. However, in practice a uniform error distribution can not cause critical problems for boundary detection in presence of a sharp normal peak, since a robust estimator such as RANSAC efficiently eliminates the outliers introduced by the uniform law.

7.6.2 Time

Computational load of the discussed method is the last evaluation parameter that we consider in this chapter. The mean computational time in second for all presented methods is shown in table 7.6.2. The time given is for processing 1000 image of size 256×8 pixels in the test set used to generate the error histogram presented in this chapter. As explained earlier in this chapter, classification methods (the first three rows) are based on sliding a window of 32×8 pixels on the image. The scanstripe width equals 8 pixels on the other hand, scanline, Fisher and gradient methods work on single-pixel-width scanlines and are therefore the fastest. The overload introduced by scanline and scanstripe method is negligible compared to classification methods.

7.7 Summary

The performances of the different methods of texture boundary detection have been analyzed through different qualitative and quantitative assessments. The error distribution on a test set of 1000 images is obtained using different methods namely, Fisher's metric, gradient-based, classification-based, Markov scanline and Markov scanstripe methods. Markov-based scanstripe results are desirable in terms of histogram shape. Classifier-based detection using only cooccurrence matrix features, on the other hand, presents rather poor response with a large standard deviation. However, by using combined types of features, it can be seen that as the number of weak learners increases, the peak around zero error gets more prominent and the distribution elsewhere becomes flat.

In terms of mean error we also notice that all texture-based methods (that is excluding gradient-based) provide acceptable error range inferior to 3 pixels on 250-pixel long images. The similarity of the histograms to the ground truth histogram is measured using EMD which is a true distance metric. The results show that scanline and scanstripe outperform other methods in resemblance to the ground truth error histogram followed, with a considerable gap, by other methods including classification-based and Fisher's and gradient-based method. Finally in terms of computational performance, scanline, Fisher and gradient methods are fastest and most adapted for real-time applications while the overload introduced by scanstripe method is negligible compared to classification methods.

Method and parameters	Mean
G	13.620
I + CM , r = 50	2.890
CM = 5 , r = 100	2.570
CM = 3 , r = 100	2.570
CM = 7 , r = 100	2.540
F	2.510
CM = 5 , r = 50	2.490
CM = 3 , r = 50	2.490
CM = 7 , r = 50	2.460
I + CM + FFT , r = 10	2.240
I + CM , r = 100	2.240
Scanstripe TM	2.050
I , r = 50	1.920
I , r = 100	1.740
I + CM + FFT , r = 50	1.670
I , r = 10	1.540
CM = 7 , r = 10	1.480
CM = 5 , r = 10	1.480
CM = 3 , r = 10	1.480
I + CM + FFT , r = 100	1.300
Scanline TM	1.290
I + CM , r = 10	1.120

Table 7.1: The absolute value of error of mean cut position on the test set using different methods. Abbreviations refer to different methods and features as follows. TM refers to the Markov texture model method. r is the number of weak learners used in the classifier-based method. I, CM and FFT are intensity, cooccurrence feature and Fourier coefficients respectively, used as features associated with weak learners. G indicates probabilistic gradient-based method and F refers to Fisher's metric method. when CM is mentioned without a value the maximum radius used for the position operators of cooccurrence matrices is 5 pixels (CM = 5).

Method and parameters	EMD
CM=7 , r=10	47.611
CM=5 , r=10	47.611
CM=3 , r=10	47.611
CM=7 , r=50	45.015
CM=5 , r=50	45.004
CM=3 , r=50	45.004
CM=7 , r=100	44.915
CM=5 , r=100	44.900
CM=3 , r=100	44.900
I + CM , r=10	44.446
I + CM , r=50	42.341
I , r=10	41.821
I + CM , r=100	41.421
I + CM + FFT , r=10	41.358
G	40.685
F	39.839
I, r=50	37.972
I, r=100	37.663
I + CM + FFT, r=100	37.140
I + CM + FFT, r=50	35.897
Scanstripe TM	23.987
Scanline TM	22.336

Table 7.2: Earth Mover's Distance between the error histogram of texture cut detection using different methods and the ground truth. Abbreviations refer to different methods and features as follows. TM refers to the Markov texture model method. r is the number of weak learners used in the classifier-based method. I, CM and FFT are intensity, cooccurrence feature and Fourier coefficients respectively, used as features associated with weak learners. G indicates probabilistic gradient-based method and F refers to Fisher's metric method. when CM is mentioned without a value the maximum radius used for the position operators of cooccurrence matrices is 5 pixels (CM = 5).

Method	Time (sec.)
I + CM , r = 100	0.498
I + CM , r = 50	0.277
I + CM , r = 10	0.063
Scanstripe TM	0.010
Scanline TM	0.004
F	0.003
G	0.002

Table 7.3: Mean computation time of texture cut detection on the test set using different methods. Abbreviations refer to different methods and features as follows. TM refers to the Markov texture model method. r is the number of weak learners used in the classifier-based method. I, CM and FFT are intensity, cooccurrence feature and Fourier coefficients respectively, used as features associated with weak learners. G indicates probabilistic gradient-based method and F refers to Fisher's metric method. The maximum radius used for the position operators of cooccurrence matrices is 5 pixels (CM = 5).

Chapter 8

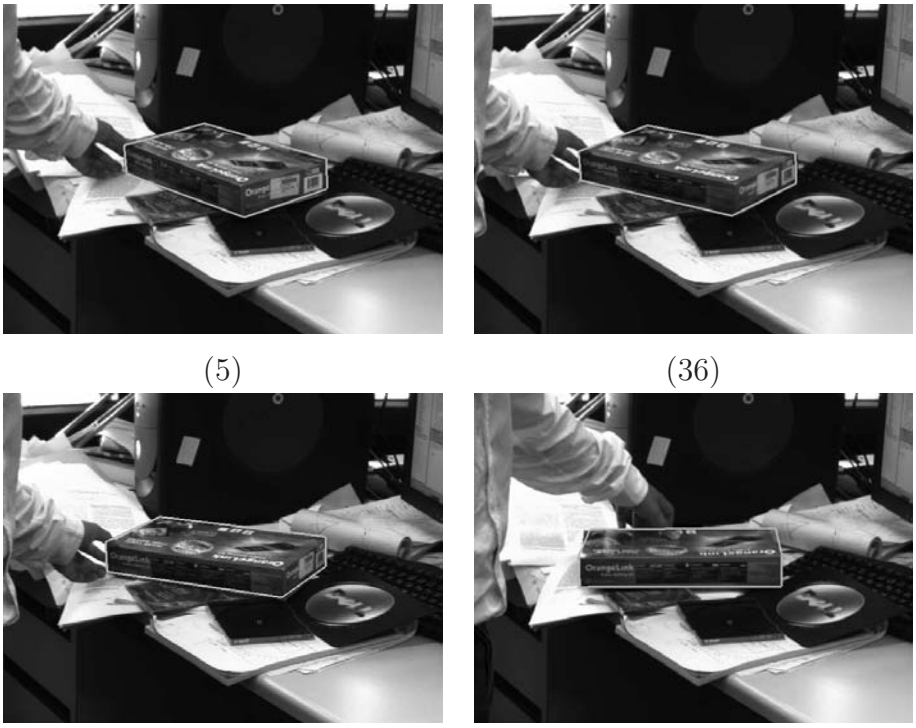
Results

In this chapter we demonstrate the aptitude of different schemes of texture boundary detection developed thus far. We first consider tracking results obtained solely through use of Markov model scanline search which has the benefit of being very fast and adapted for real-time tracking. Next we present tracking and interactive texture segmentation results achieved by Markov scanstripes and robust boundary reasoning. Finally, examples of tracking of geometric and deformable objects and interactive texture segmentation employing trained classifiers are apprised.

8.1 Scanline-based Tracking

We consider first the transition matrix model representation of texture based on a 1^{st} order Markov sequence model as described in Chapter 4 which we refer to as Markov-based scanline method. The algorithm described in Section 4.1.2 is used to detect the texture crossing position on scanlines which are perpendicular to regularly spaced sampled points on the projection of the object model. The independently detected boundary points are then used to estimate the 3–D model pose for the current frame.

Figs. 8.1 shows the results of real-time tracking of a diversely textured box against a cluttered background. The fact that transition matrices of each scanline are formed according to the pixel sequences that lie beneath them, makes it possible to deal with the presence of totally different patterns on the target. This



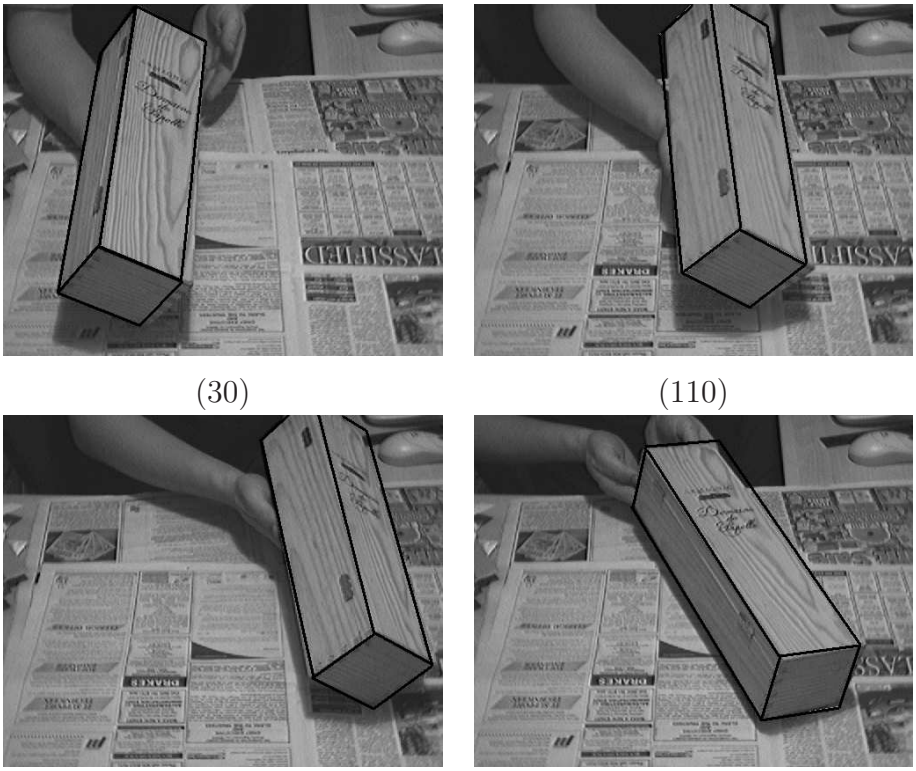
(5) (36)

(45) (101)

Figure 8.1: Tracking a textured box against a cluttered background. The numbers show the frame number.

is in contrast to the use of global texture models which fail to finely attain local characteristics of the texture.

To illustrate the effects of first order statistics of texture in the texture boundary detection method, we consider tracking an object against a background with similar histograms, where the spatial distribution of intensity values are different. Fig. 8.2 shows the tracking of a brightly colored box with dark traces. The histogram of the color intensity of the box resembles that of the newspapers in background, however, the alternation of dark to bright pixels and vice versa is more frequent on the regions corresponding to newspaper. This distinction is taken into account in the transition matrix since it represents the frequency of occurrence of different pixel intensities in neighboring pixels. Therefore we expect to be able to find texture boundaries in those cases which is confirmed by the observed results.



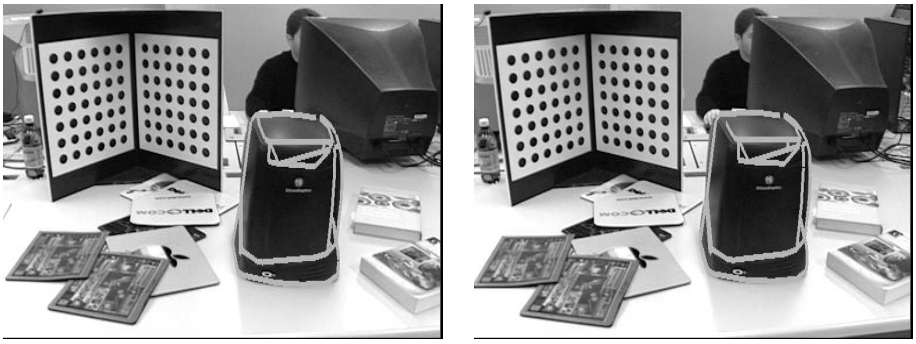
(30) (110)

(190) (260)

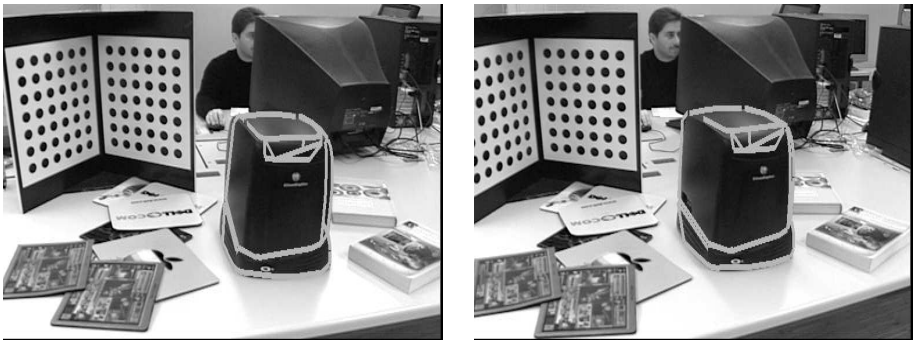
Figure 8.2: Tracking a textured box against a cluttered background without using prior models. The model is materialized by black lines. The numbers show the frame number. Images and results are courtesy of Tom Drummond, university of Cambridge.

Finally an O_2 computer is tracked and some frame of the results are shown in Fig. 8.3. The significance of this example is to demonstrate that the transition matrix model adapts also to objects which lack distinctive texture features.

The implementation used to generate the above results can process up to 120 fps on a 2.6 GHz machine. The tracking speed depends linearly on the number of samples (scanlines) per visible edge of the model. This low computational cost potentially allows the tracking of multiple textured and non-textured objects in parallel using a single PC.



(1) (12)



(29) (110)

Figure 8.3: Tracking of a non textured O_2 computer against a cluttered background. The numbers show the frame number.

8.1.1 Monocular Body Motion Tracking

Monocular articulated human tracking is challenging because it involves tackling such difficult issues as ambiguities associated with articulated motion seen from a single camera, very high dimensional search spaces, partial self-occlusions, and poor quality of image features in the absence of markers. In this part we present a multiple cue optimization framework which uses the Markov-based scanline silhouette extraction method for 3–D human motion tracking.

We start with a system that uses feature points to track and we add to it contour tracking and ambiguity detection. For the latter we chose a method similar to the one described in (Shimada *et al.*, 1998) which considers all possible 3–D positions that can generate the same projection. These 3–D positions are

mirror images of each other with respect to the image plane and we refer to them as "ambiguity conjugates". In case of ambiguous body motion, i.e. when the body part is nearly parallel to the image plane, the ambiguity conjugates can be used as starting guess for nonlinear optimization.

8.1.1.1 Overview of the Tracking System

Given a set of frames, and a reference pose for a single frame, we extract cues about silhouettes and point correspondences. These cues are used iteratively for the generated hypotheses along with known joint limits in an optimization framework to estimate the pose for the subsequent frames. This idea is schematically presented in Fig. 8.4.

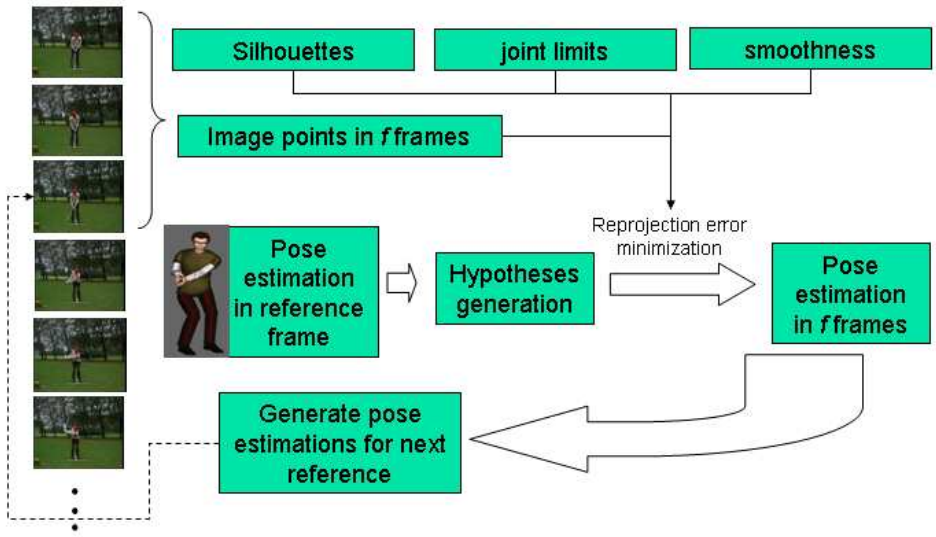


Figure 8.4: Overview of the tracking system.

We model each body part in the chain of articulations by a quadric (ellipsoid, cylinder or truncated cone) represented by a 4×4 matrix Q that describes both

its shape and its position in space, which is a function of body poses defined in the hypothesis set. Finding the pose in the next frame is then carried out by minimizing the cost associated to each hypothesis set element. The cost is a combined energy function consisting of the following terms:

$$\begin{aligned} \mathcal{F} &= \textstyle\sum_i F_i^{pnt} + \sum_i F_i^{silh} \\ F_i^{pnt} &= w_i^{pnt} \left\| \widehat{p}_i - \overline{p}_i \right\|^2 \\ F_i^{silh} &= w_i^{silh} \mathrm{Dist}_{s_i}^2 \end{aligned} \tag{8.1}$$

We describe these terms in detail below.

Figure 8.5: A set of matched point between two frames.

Regions Inside Target (Term F^{pnt}) Robust image point matching is used to generate point correspondences which serve as observations for projection error minimization in the subsequent frames. In order to increase accuracy of the correspondences, we incorporate affine 2–D image motion estimation for matching in the subsequent frames. The set of matched points is further refined by robust fundamental matrix computation and enforcement of epipolar geometry constraints on each body part separately. This provides the first term in the energy function given by Eq. 8.1, where $\overline{p}_i$ is a matched point in frame t, and $\widehat{p}_i$ is the estimated projected point using model pose corresponding to frame t. A set

of matched points in two successive frames are shown in Fig. 8.5, where different colors represent points belonging to different body parts.

Target's Outline (Term F^{silh}) Given an approximate body pose and corresponding model outlines in the image such as those shown in Fig. 8.6 where an ellipsoid is used to model the arm, the body silhouette points are detected using the Markov-based scanline method explained in Chapter 4. The silhouette points are then used to calculate distances Dist_{s_i} of the cast ray passing through a given silhouette image point $s_i = (u', v')$ from the associated quadric Q on the model in 3–D space. These distances are added to the energy function to be minimized. The silhouette terms along with the terms corresponding to the image point observations are robustly weighted to exclude outliers and normalized to have balanced impact on the energy function.

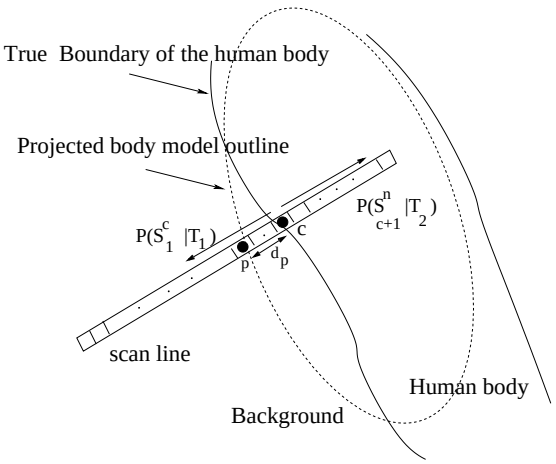
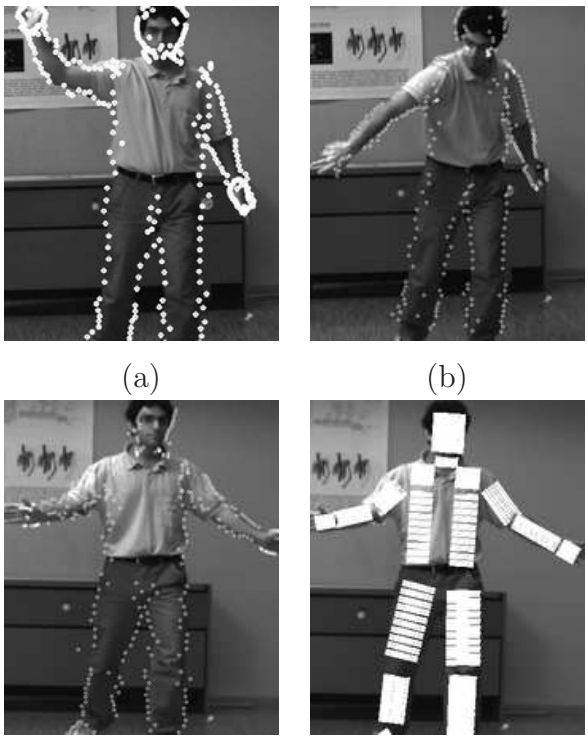


Figure 8.6: Scanning a line through model sample point p for which a 1^{st} order statistical model is used to find the texture crossing point c.

Fig. 8.7 shows some detected body outlines obtained from a quadric body model samples. The search starts from samples on the projected model outline in the previous position. The computation of the projection of generalized quadrics is described in Appendix B. This starting point is not necessarily close to the real outline. This is due to the fact that the model is an approximation of the real subject and moreover its pose comes from the previous frame. During tracking,

the statistical model of the texture of the tracked subject is constructed in the first frame in terms of a transition matrix for each body part. This is done by manual initialization and considering the pixels under the projection of the body pose on the corresponding frame. The patches used for this training phase are shown in Fig. 8.7-(d).



(a) (b) (c) (d)

Figure 8.7: Examples of the detected silhouette points on the body (a, b, c). White circles are the detected texture boundaries corresponding to the body outlines. (d) illustrates the training phase for Markov texture modeling, the white patches show the areas used to compute the model for each body part. This is done by rendering the initial pose given by manual initialization and obtaining the mask of its projection on the corresponding frame.

In practice not all possible joint angle values are acceptable and we constrain them to remain within an acceptable range. For each such constraint, we add a penalty term which is nonzero if the parameter goes beyond the limits. Another

plausibility factor is the smoothness of motion. Furthermore, we suppose that the human motion is reasonably steady in a short frame interval. It implies that body parts do not tend to change direction abruptly. This gives the last term in the energy function of Eq. 8.1 where θ_i^t is the joint angle i at frame t. In our implementation observations from three adjacent frames are used to minimize this objective function for each hypothesis. The resulting pose are iteratively used in the same manner throughout the sequence.

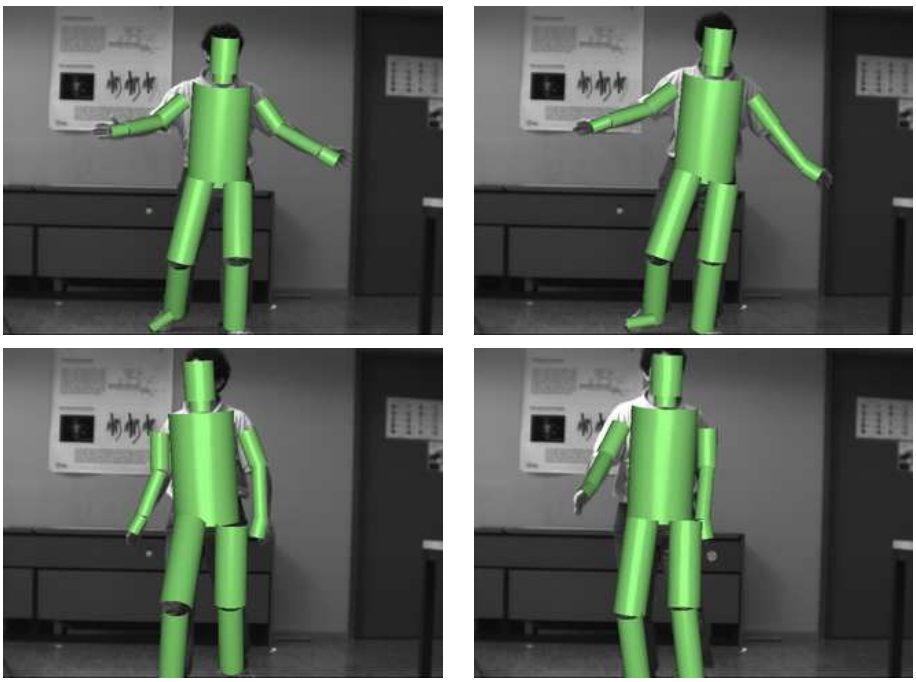


Figure 8.8: Several frames of a tracking sequence of 100 frames with the 3–D model superimposed on corresponding frames.

We consider a sequence of walking toward the camera with considerable change in target's dimensions. The motion towards the camera is used due to the fact that no motion model is assumed to handle occluding body parts. Fig. 8.8 shows some shots of the original 100 frame sequence with superimposed tracking results. The results are further verified by resynthesizing the scene as seen by an arbitrary camera. Fig. 8.9 shows the resynthesized view of the tracking results from frontal top to let us observe the forward motion of the model towards the camera. We

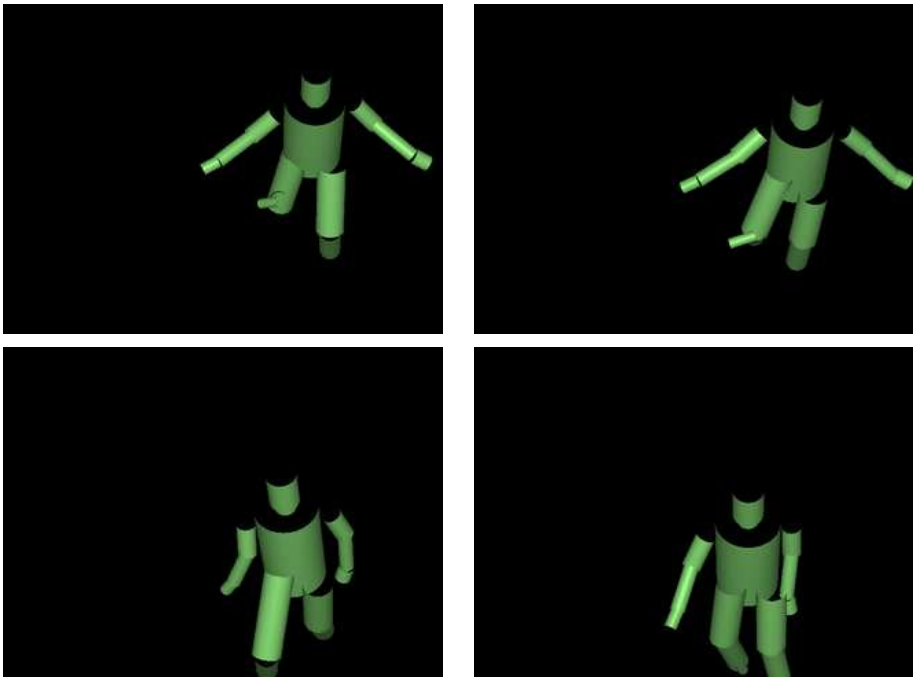


Figure 8.9: The resynthesized view of the tracking results shown in Fig. 8.8 seen from frontal top.

have further tested the accuracy of the results and the validity of the 3–D pose tracking by augmenting the frames obtained by a second camera that was not used during tracking as shown in Fig. 8.10. We can see that the virtual logo on the shirt and the virtual box attached to the hand of the moving person are placed in the right position in the scene throughout the sequence. Finally for the purpose of illustration of applicability of our tracking method in Fig. 8.11 we have applied the recovered motion to a 3–D human animation model.

The second example consists of tracking a person wearing a highly textured dress. Moreover the background is highly cluttered and contains a moving person. Fig. 8.12 shows some frames of tracking results with the superposed model.

We also compare the Markov-based algorithm with the gradient-based method. Fig. 8.13 shows part of the body and the silhouettes extracted using texture-based (left) and gradient-based (right). This results illustrate the advantage of using the Markov-based texture boundary detection technique instead of gradient-based

Figure 8.10: The credibility of the results in 3–D space are verified by augmenting the frames obtained by a second camera that was not used for tracking.

techniques.

8.2 Bayesian Scanstripe Robust Silhouette Extraction

In this section we lay on our experimental results of applying the scanstripe Markov model of texture cut and the robust texture boundary detection to both tracking and interactive texture segmentation. The resulting system is obviously computationally more expensive with respect to the system presented in previous section which is composed of independent scanlines, however, it has the advantage of being robust and reliable. Scanstripes cause faster convergence of the texture transition matrices and consequently require shorter search lines. Furthermore, robust reasoning restricts the extracted boundary to conform with the object geometry or outline characteristics such as smoothness. Geometrical constraints are

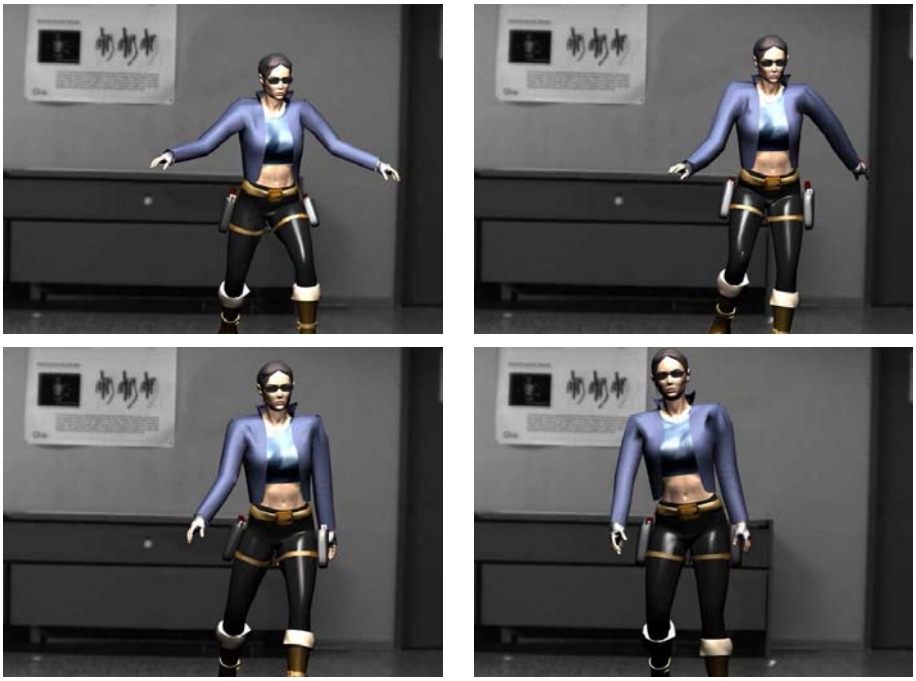


Figure 8.11: Walking avatar of the tracked subject.

realized in terms of RANSAC algorithm are illustrated in tracking examples. On the other hand, smoothness constraints are enforced using the Viterbi algorithm as shown in the remainder of this section.

8.2.1 Tracking Using RANSAC

Robust scanstripe texture boundary tracking with KL divergence-based update described in Section 4.5.2 is tested using a video sequence of a shiny magazine cover. The KLD is used to update the prior texture model in order to cope with changes in lighting conditions. The magazine reflects light differently as the camera moves around it. Moreover, there is a high amount of motion blur due to the motion of the camera and the interlaced images.

In this tracking example, we use the fact that model is made of straight edges and therefore robustly fit straight lines to the scanstripe change probability distribution directly using the enhanced RANSAC method explained Section 6. Moreover, we update the texture model in course of tracking using Eq. 4.23

Figure 8.12: Tracking under extreme conditions. The subject is wearing a highly textured dress. The background is highly cluttered and contains a moving person.

with the parameter α which depends on the KLD of the prior learnt model and the current target texture according to Eq. 4.22. Sample frames of the 300-frame sequence of tracking are shown in Fig. 8.14. The hypothesis line segments generated by RANSAC algorithm for each edge are shown in red and the white wire frame shown is the final fitted model position.

On the other hand, if we use a constant (including zero, i.e. no update) value for α in Eq. 4.23, the tracking fails after a few seconds as shown in Fig. 8.15.

Next, we compare results obtained on the magazine sequence employing other methods, namely, an edge-based tracker (Drummond & Cipolla, 2002), the tracker with only scanlines and only scanstripes without robust linkage. Observations show that the RANSAC linkage of scanstripes with the KLD-based update outperforms other methods. Moreover the speed of the method remains close to real-time (6-10 fps on a 2.8 GHz Pentium 4). The edge-based tracker goes astray by the strong edges in the cover of the magazine as can be seen in Fig. 8.16-(a)

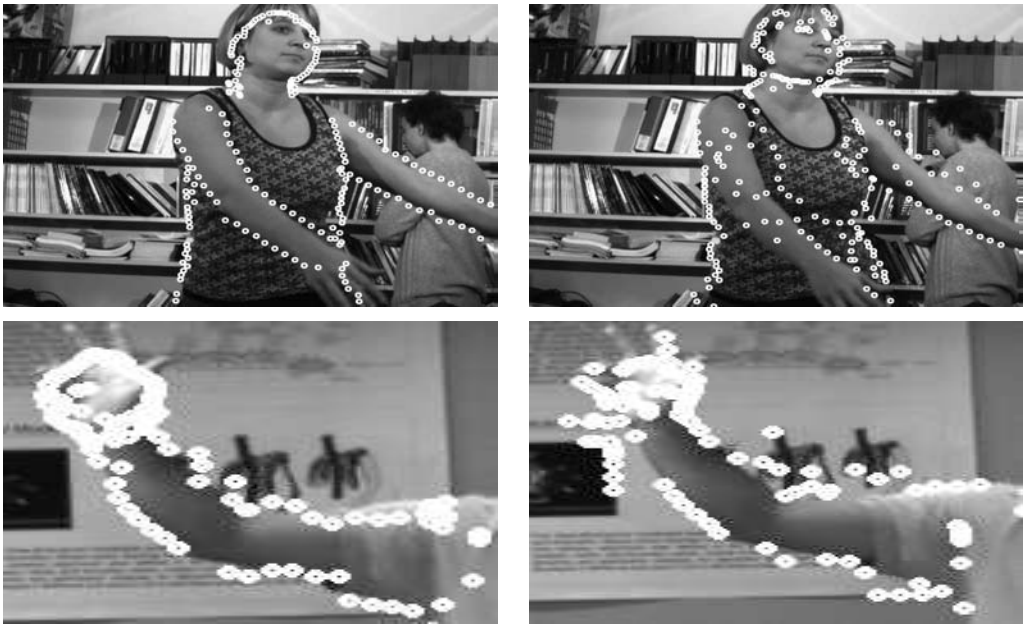


Figure 8.13: Examples of silhouettes obtained using Markov scanline model (left) compared with those obtained by a gradient-based method (right).

for the 90^{th} frame. The scanline tracker is also adrifted due to poor estimation of texture parameters for some edges as shown in Fig. 8.16-(b). Using scanstripes improves the results (Fig. 8.16-(c)), while RANSAC linkage of scanstripes which enforces the fact that the searched boundary is composed of straight lines improves the stability of the tracking results as shown in Fig. 8.16-(d)

8.2.2 Delineation Using HMM

Another application of the proposed robust outline detection method is interactive texture segmentation. The impulse for this application is the fact that our developed algorithms take as input an initial estimate of the boundary position which defines the samples and the search direction. In the case of tracking the initial guess comes usually from previous frame. In the case of interactive segmentation the initial guess is provided by a small gesture of the user.

Fig. 8.17 shows some detected texture boundaries starting from an initial guess. The user draws a circle or some straight lines which are used to define the

Figure 8.14: Tracking with KLD-based update of the prior model. RANSAC is used to fit straight lines (shown in red) to the scanstripe probabilities directly which are then used to calculate the pose of the model (white wire frame).

scanstripe directions. Scanstripe probabilities are then linked using the HMM described in Section 6. The state transition of the hidden Markov model is represented by a Gaussian kernel which ensure that boundary points on succeeding scanstripes are spatially smooth. The optimal boundary is then obtained using the Viterbi algorithm.

Using a single transition matrix in modeling horizontal and vertical intensity cooccurrences in scanstripes might seem counterfactual as discussed in Section 4.4, however, the results on detection of the boundary between two zebras illustrate that the a single scanstripe transition matrix can handle difficult cases such as presence of similar non-isotropic textures with a single distinction of being differently oriented.

To deal with general and complex outlines, a relatively accurate initial guess

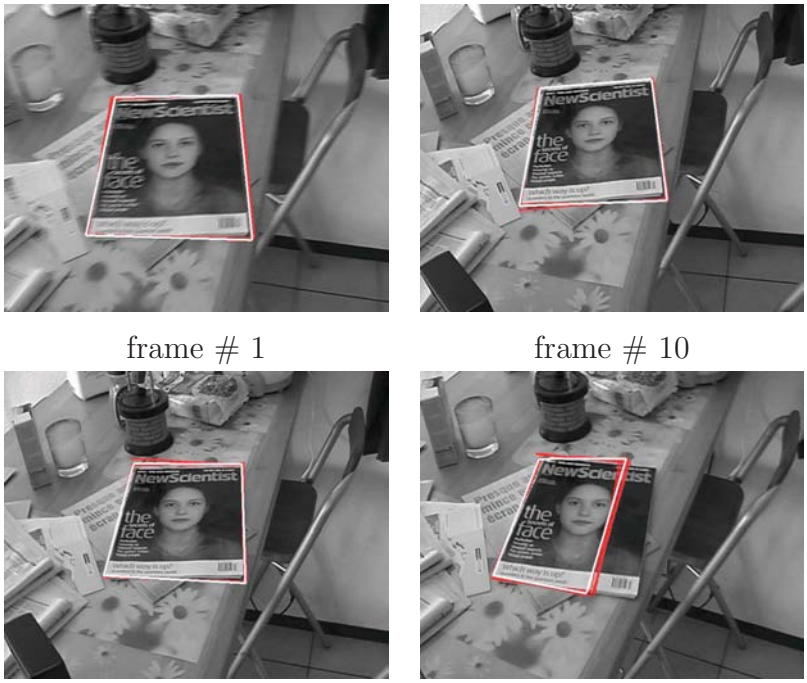


Figure 8.15: Tracking without dynamic texture model update. Tracking is lost well before 100 frames due to changes in target appearance.

is required. This is due to the effect of the state transition distribution which assumes a good alignment of scanstripes in a rectified sense using the associated normal directions. To deal with such cases we can use a spline technique to fit a smooth curve to a few control points provided by user. Some segmentation results are shown in Fig. 8.18 where the initial guess in obtained by fitting an spline curve on a few user-provided points.

8.3 Classification-based Approach

Similar to the scanstripe and scanline tracking scheme, we assume that at each frame during tracking to have an initial guess for the object silhouette. We consider tracking of deformable and rigid objects. The trained classifier's conditional probability model is applied at each model sample point in a directional search

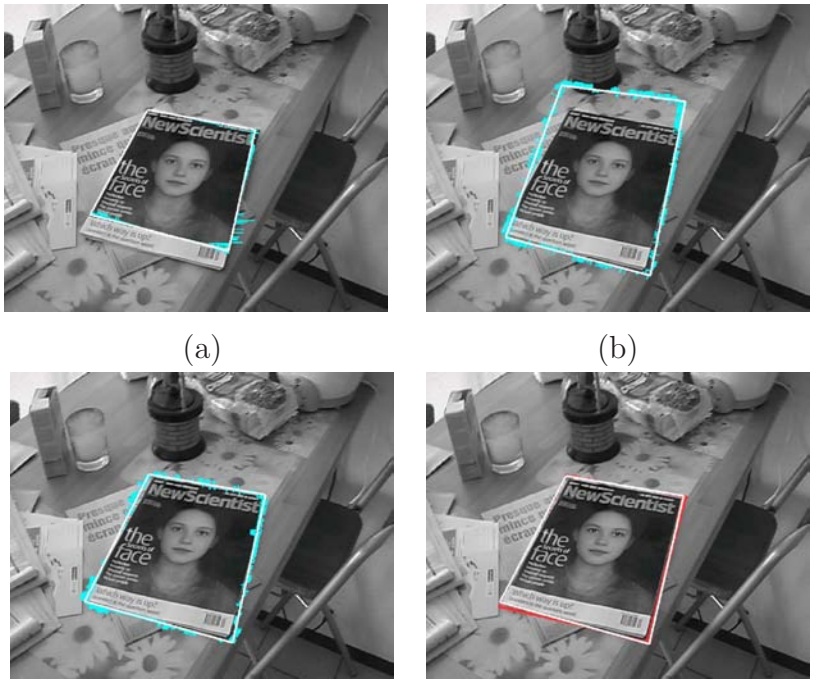


Figure 8.16: Frame # 90 for tracking using different methods: (a) edge-based tracker, (b) scanlines only, (c) scanstripes only, and (d) RANSAC linkage of scanstripes.

normal to the model edge at that location. Each scanline gives independently the probability of texture change across it. These probabilities are then aggregated to yield a probability distribution for the most probable contour, i.e. the object boundary.

The implementation of the classifier-based system can reach a speed up to 8 fps on a 2.8 GHz Pentium 4. Its speed mainly depends on the number of samples on the model, length of the pixel sequences on each sample point, and the number of weak learners used in the conditional probability classifier. According to our experiments a low number of classifiers (e.g. 4 to 10) depending on the complexity of the sequence is enough for reliable contour tracking. In the remainder of this section we provide examples of tracking of rigid and deformable objects as well as interactive segmentation.

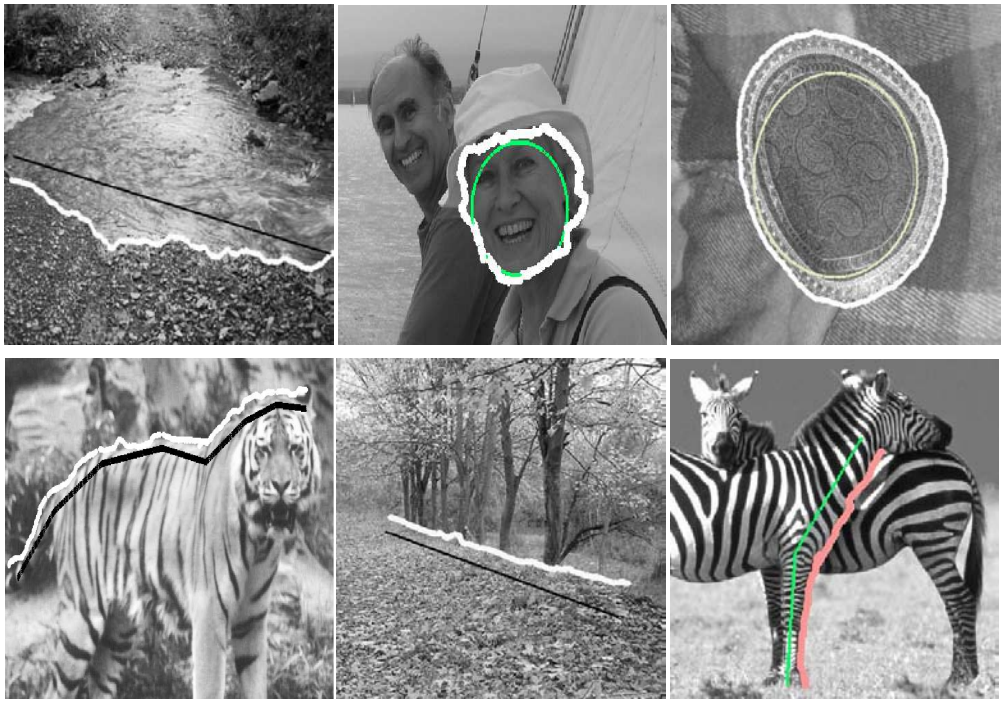


Figure 8.17: Fast interactive texture segmentation. An initial guess is given by the user (thin circles or straight lines). The HMM is used to link scanstripes and the Viterbi algorithm gives the final texture boundary shown as a thick curve. In the case of the zebras, note that the algorithm finds the boundary between similar textures of different orientations.

We track a rigid object using the classification-based system and RANSAC pose optimization which used the available CAD model of the target. Fig. 8.19 shows several frames of the tracking results in presence of clutter on a sequence of 600 frames with large and fast camera motions.

When the model does not consist of straight or geometrically well defined edges stochastic search for the optimum model parameters is not trivial. In this case we apply the HMM with a Gaussian state transition to the classifier's responses similar to the scanstripe framework of Section 8.2.2.

The performance of this approach is demonstrated by tracking of the upper body deformation. In this system, the user defines an initial path by clicking on some points around the body outline. These points are used to fit a spline

Figure 8.18: More interactive texture segmentation results with initial curve defined by a spline curve. The black curve on white band marks the detected outline and the thin gray line is the spline curve.

curve which serves as the starting guess for the body outline. The outline is then obtained using the conditional probability classifier in conjunction with HMM and it serves as the initial guess for the successive frames. Several frames of the tracking results are shown in Fig. 8.20.

Provided a 3–D mesh we can use the extracted silhouettes to deform the mesh to track 3–D deformations using an implicit representation of the mesh (Ilić & Fua, 2005). Fig. 8.21 shows some frames of the 3–D reconstruction results obtained by using the extracted 2–D silhouette observations to track 3–D implicit mesh deformations.

Finally we can use the classifier with a small set of weak learners to perform interactive texture segmentation. Examples shown in Fig. 8.22 use 4 weak learners to calculate the conditional texture cut probability. The user provided spline curve is shown by the thin line and the calculated outline is marked by the thick curve. In the case of the llama a Gaussian state transition kernel with standard deviation of 5 pixels is used in the HMM to extract the outline. On the other hand a uniform state transition probability in neighborhood of 5 pixels is used to handle great curvatures and distant initial guess in the case of the flower.

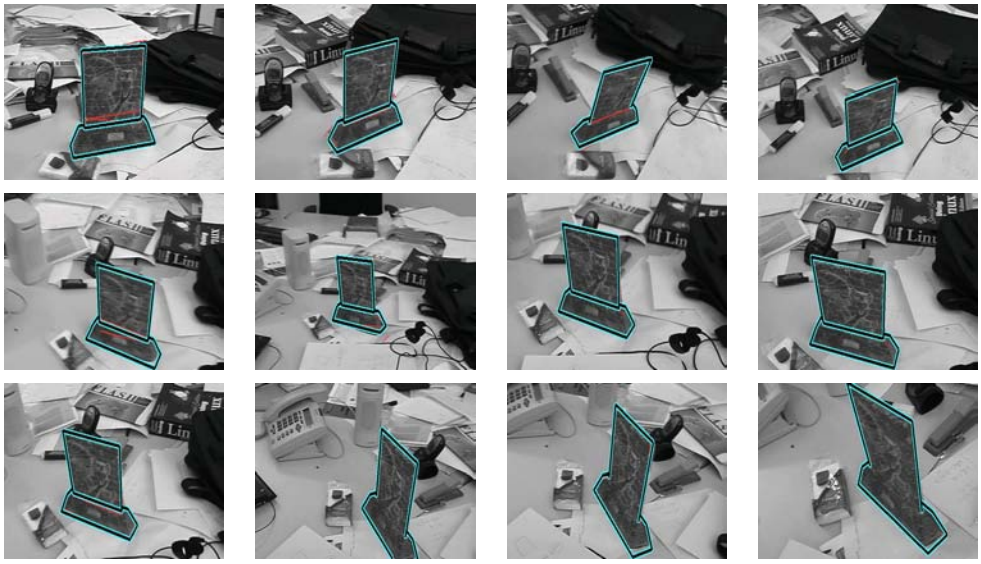


Figure 8.19: Tracking textured object against cluttered background.

8.4 Summary

Some experimental results of tracking and interactive texture segmentation using our proposed methods are presented in this chapter. Independent Markov model scanline search has the benefit of being very fast and adapted for real-time tracking. However robustness can be achieved by employing scanstripes, classifiers and object specific constraints in expense of more computational cost which are nonetheless considerably low with respect to existing texture segmentation methods. Among other covered results, the advantage of using KLD adaptive prior model update is also illustrated in a tracking case. Lastly, interactive texture segmentation results are reported for challenging examples using both methods of Markov and trained classifier with HMM concluding.

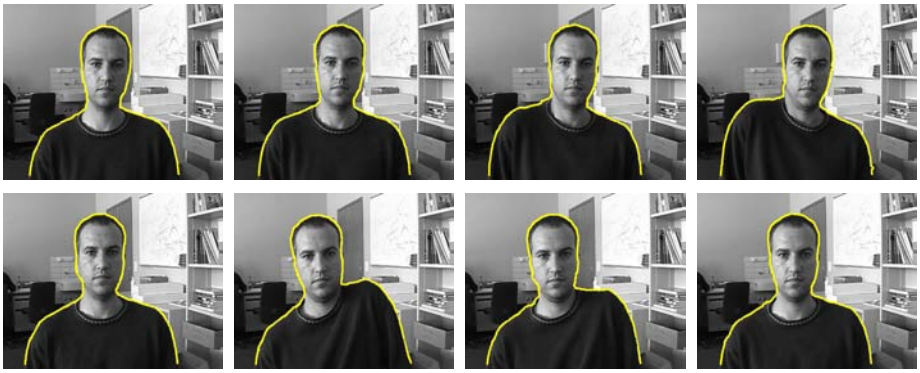


Figure 8.20: Tracking of deforming body outlines.

Figure 8.21: 3–D Tracking of deforming body outlines.

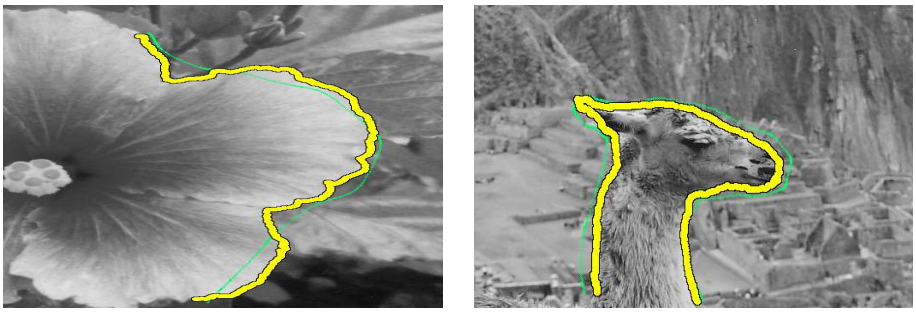


Figure 8.22: Interactive segmentation using trained classifiers.

Chapter 9

Conclusion

We have presented a novel probabilistic substratum for silhouette detection and tracking which has the advantage of being fast and robust. While tracking using gradient-based algorithms are computationally fast and therefore favorable, they suffer from vulnerabilities due to loss of intensity information. Texture segmentation techniques and graph cuts on the other hand are computationally expensive and not adapted to tracking. We have explored two distinct approaches to fast silhouette extraction for tracking applications which draw together the speed of gradient-based and the robustness of texture segmentation algorithms.

We have introduced a fast algorithm based on a Markov model that uses a simple line search to detect the transition from one texture to another that occurs at object boundaries. This algorithm provides the exact solution to Bayesian estimation of the boundary given a sequence of pixels. Furthermore, direct relationship between the texture entropy and the texture probability given by the algorithm is derived and exploited to update the dynamic prior texture model during tracking to cope with changes in illumination or object's appearance adaptively.

For the cases where the given observations are too few for reliable Bayesian estimation of probability of texture change we have proposed an innovative machine learning technique to generate a probabilistic texture transition model. The problem is expressed in terms of classification of a set of images with texture transitions in the middle. The classification is then handled by a small set of trained weak learners. The weak learners are associated with different features namely,

mean energy of intensity or Fourier transform coefficients as well as cooccurrence matrix elements. The total classifier score is a weighted sum of weak learners responses where training and weights are calculated by the Adaboost algorithm. In order to estimate the conditional probability of texture cut, given the neighborhood we have developed a normalization scheme which uses the classification score on the training set to model that probability with a Sigmoid function.

The advantage of using the trained classifier over the Markov Bayesian estimation of texture crossing lies in the fact that training is carried on off-line and therefore it offers better convergence for complex texture models compared to the Markov method. Moreover, the classifier-based method is capable of learning a specific target texture. However it does not include dynamic texture models and it requires more computational power.

The probabilistic model of texture transition given by the classifier or the Markov model accommodate a convenient way to incorporate the constraints associated with the tracked object. These constraints are expressed in terms of a likelihood distribution given in the vicinity of each contour point, thus allowing the extraction of the optimum contour for tracking. This is done in two different fashions depending on the nature of the tracked object:

- **HMM.** A Hidden Markov Model is defined to calculate the joint law of local conditional probabilities along the contour which is particularly useful in case of tracking deformable objects or in cases where smoothness and continuity of the outline is the natural constraint.

- **Stochastic pose estimation in case of rigid objects.** Enhanced RANSAC method is used to find a pose that maximizes the sum of probabilities of contour points given by the classifier. This method is preferable to find a pose that maximizes the sum of probabilities of contour points of rigid objects with geometrically well-defined models. Furthermore, scanstripe texture transition probabilities can be used as prior distribution for hypotheses generation. Drawing hypotheses according the prior distribution leads to faster convergence towards the most probable observations.

The performances of the different methods of texture boundary detection have been analyzed through various qualitative and quantitative assessments. The results show that Markov texture model with scanline and scanstripe search outperform other methods in term of relative similarity to ground truth results and computational load. We have also demonstrated the effectiveness of tracking and interactive texture segmentation using our proposed methods. Independent Markov model scanline search has the benefit of being very fast and adapted for real-time tracking. However more robustness can be achieved by employing scanstripes, classifiers and object specific constraints in expense of more computational cost which are nonetheless considerably low with respect to existing methods.

In short, our approach provides a reliable texture-based alternative to gradient-based techniques under difficult conditions. It combines the desirable speed of edge-based line search and the sophistication of Bayesian texture analysis given a small set of observations. In terms of direction for further improvements and future work, it would be interesting to examine the ways the classification approach can be combined with the Markov texture model. It is possible to train classifiers on scanline probabilities obtained using the proposed Markov model on a training set. This method has the potential of being fast, stable and robust using a few observations. The stability is due to the fact that the classification method is based on a large training set. Moreover this method is fast since the classifiers are trained over and use the output of Markov scanlines not on images. Moreover as shown in this thesis, the log of Markov scanline probabilities is related to texture entropy and therefore this approach has the virtue that classifiers will be trained to detect changes in texture entropies.

Appendix A

Human Perception of Contours and Texture

In this Appendix we provide a brief study of the human visual perception and how outline and silhouettes are interpreted and treated by human brain to understand our surroundings. This study is useful in coming across guidelines in development of computerized techniques for silhouette detection and tracking.

Understanding how the complex system of visual perception and cortical processing leads to identification of 3–D structure of the objects and environment is an arduous task. According to the studies of the visual systems, various image primitives and their location within the scene are processed by the visual cortex and a neural description is produced to represent them. The original retinal image is thus approximated by this description. The details of the visual image are believed to be processed hierarchically by the neural mechanisms of visual perception (Tse & Hughes, 2004). This processing is carried out at different steps, the first of which is a filtering that allows extraction of different building blocks, clusters and primitive features.

Independent modules in the cortical tissues are responsible for extracting and processing different primitive features from visual information and higher levels of visual processing combine mixtures of those features into progressively more complicated structures. This higher level interpretation can identify surfaces and objects either by their contours and boundaries as perceived by the visual system

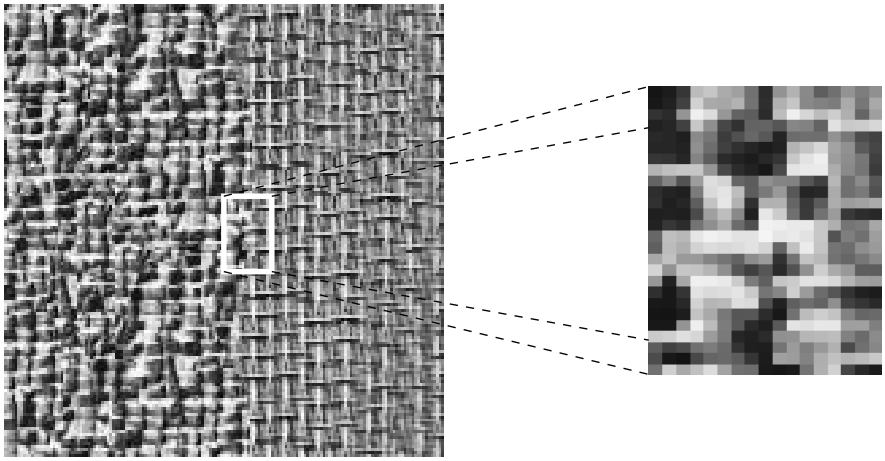


Figure A.1: Unless we are given enough observations to build a model of texture on both side we are not able to visually detect a texture boundary. The edge in the left image is easy to detect, on the other hand we look only to a small patch on the edge of the texture we cannot perceive the boundary (right).

or by employing localized spatial frequency filters (Tse & Hughes, 2004). The discernment of edges and contours by humans is remarkably excellent, provided that there is a large enough patch of texture on each side available as demonstrated in Fig. A.1. This observation indicates the underlying fact that a decent texture boundary detection requires a reliable texture model of the adjacent textures to enable precise localization of edges.

Low- and high-pass filtering in the early cortical stage can convey information about overall image structure and may contribute to certain Gestalt[1]-like operations such as grouping, closure and good-continuation. The painting shown in Fig. A.2 is a good example of perceptional effects of low pass filtering, where, the blurred version turns into a human skull. On the other hand, it can happen that the low-spatial frequency information is masked by high spatial frequencies as shown in the painting of Fig. A.3, which becomes the portrait of Abraham Lincoln when blurred.

Many of the contour extraction, pattern analysis, segmentation and tracking

[1]The perceptual process of separating figure and ground to create an overall visual understanding of an image.

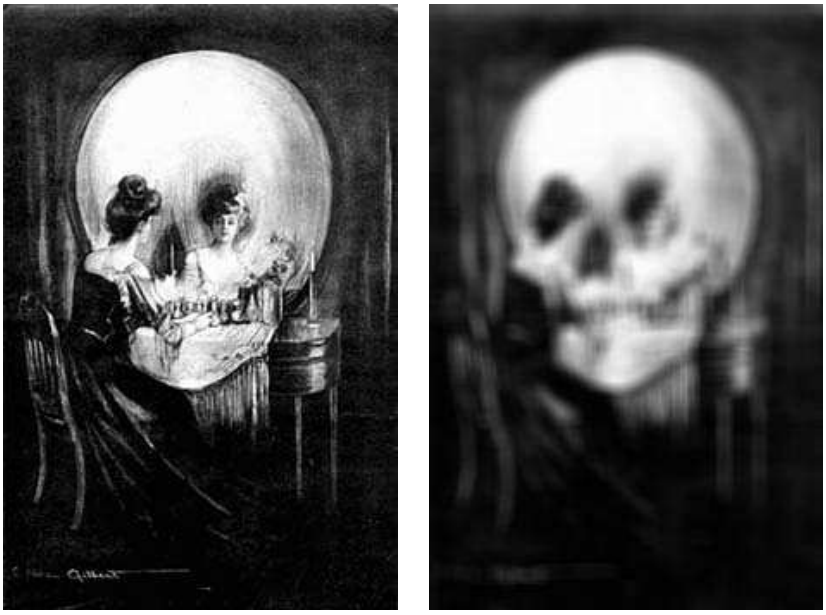


Figure A.2: Visual effect of low-pass filtering. “All is Vanity” by Charles Allen Gillbert (1873-1929). Is it a young woman at her mirror, or a skull?

methods rely on different cues based on color and texture information. It is therefore of great importance to consider the relationship between color and texture perception in the human visual system. Intuitively, texture analysis methods work with gray-level images. In addition to simplicity of using intensity values instead of 3-channel color information, the human visual system also treats color and texture information separately (Mäenpää, 2003) and color information is not always necessary for inference of images by brain. Fig. A.4 shows a case where the only incertitude about the gray-level image (right) is the color of the scene and all other information about texture of the entities can equally be extracted from it. Then we can roughly say that color is a complementary source of information and it acts as a cue for detailed visualization of the image. This hypothesis also explains the fact that color blind people can still recognize textures of the objects.

Color is apperceived by the brain as a regional property (Wandell, 1995). therefore the high frequency information of color is not treated by the visual system. This fact is exploited in fabrication of imaging sensors and image and video compression techniques. Most CCD sensors have one color receptor (R, G,

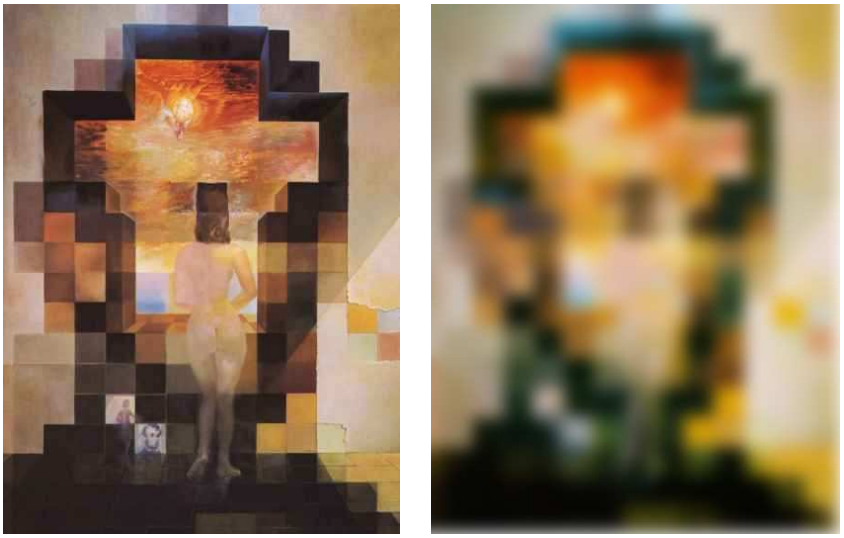


Figure A.3: Visual perception influenced by high-pass filtering. "Gala Contemplating the Mediterranean Sea" by Salvador Dali (1904-1989). The original painting (left) seen from a distance of 30 meters turns into a portrait of Abraham Lincoln (right).

or B) for each pixel and the color is then interpolated between pixels and the resulting image adequate in terms of human perception.

As further evidence on separation of color and texture information, scientific research shows that the image signal in the brain is composed of luminance and chrominance components (Mäenpää, 2003). These signals are treated in different regions of brain with some secondary interaction (DeValois & DeValois, 1990). Furthermore, the psychological investigation of Poirson & Wandell (1996) illustrate that color and pattern processing occurs separately.

Another study proposed by Mojsilovic *et al.* (2000) shows that color pattern is formed through interaction of luminance, chrominance and achromatic signals. The chrominance and luminance components are treated in early visual cortical areas while higher levels of image formation are responsible for the processing of the achromatic signal. The luminance and chrominance generate color information while the achromatic signal is used to extract texture. Mojsilovic *et al.* (2000) conclude that human perception of patterns is independent of the color informa-

Figure A.4: Lack of color information (right) has no effect on visual inference of texture and patterns.

tion in the image. Therefore it can be stated that human visual perception can be decomposed into independent dimensions. The major dimensions corresponding to color are overall color and color purity which correspond to representative and dominant color and level of colorfulness respectively. On the other hand texture dimensions form a subspace with major dimensions such as "directionality and orientation", regularity and placement rules"and "pattern complexity and heaviness" (Mäenpää, 2003).

Finally, another impressive aspect of our visual system is its capability to construct and complete object contours that are occluded, or even imagine illusory contours as illustrated in Fig. A.5. This characteristic of the visual system is produced by the grouping procedure in the early visual cortical areas that compare contour orientations across the image combined with low- and high-pass filtering operations (Tse & Hughes, 2004). These observations also suggest that a reliable object (texture) segmentation algorithm should be able to acquire the fact that individual objects or a set of similar objects in the world tend to produce image regions covarying in certain ways.

The completion shape and salience problem addresses the computation of the shape and relative likelihood of the family of curves that potentially connect a set of boundary fragments. *Stochastic completion fields* and a neural model of illusory contour shape and salience are treated by Williams & Jacobs (1997). They show that the probability distribution of boundary completion shape can be modeled

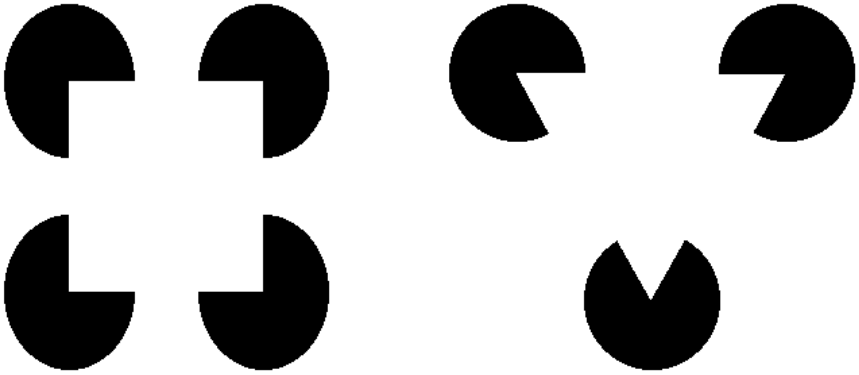


Figure A.5: The optical illusion introduced by circles with missing sectors. The visual system percepts fictitious opaque white shapes in front of circles.

by a Hidden Markov model and a random walk of position and orientation in image plane.

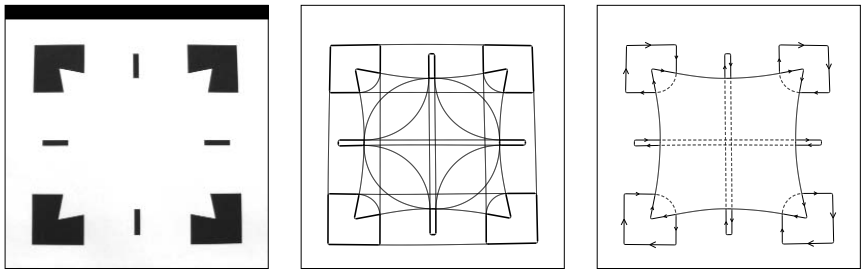


Figure A.6: Stochastic completion field. Input image(left), potential boundaries (middle) given by the stochastic completion field. (Right) is the human perception of the complete image. Image courtesy of Williams & Jacobs (1997).

A stochastic completion field is a parallel distributed representation of the family of curves that potentially connect two subsets of states, P and Q representing the beginning and ending points of a set of boundary fragments. The stochastic completion field provides the probability that a particle, initially in state (x_p, y_p, θ_p), for some $p \in P$, will do a random walk, ruled by the prior on completion shape, pass through state (x_q, y_q, ϕ_q), for some $q \in Q$, for all values

of u, v and ϕ, this is illustrated in Fig. A.6. The random walk model allows definition of a Markov process defined on three dimensional state space inspired by the receptive field of neurons in area V1 of visual cortex and governed by some equations of motion.

A. HUMAN PERCEPTION OF CONTOURS AND TEXTURE

Appendix B

Projective Geometry and Camera Model

In this appendix we review briefly the projective geometry and concepts such as points, lines, planes, conics and quadrics in two or three dimensions. We also present the pinhole camera model that is used to represent the image generation process. We use the projective camera model extensively in the 3–D tracking framework represented in Section 2.1 as well as 3–D articulated human body tracking results of Section 8.1.1. The conic formulation and analysis of its different forms are used in Section 8.1.1 to project body parts represented by generalized quadrics and resampling of their contours.

B.1 Projective Geometry

In homogeneous coordinate system a point in projective 3–D space, $\mathcal{P}^3$, is given by a 4-vector of coordinates $\mathtt{x} = [x_1 \dots x_4]^\top$. If x_4 is zero then point is defined to be at infinity. Two points $\mathtt{x}$ and $\mathtt{y}$ are equal if for a nonzero scalar λ we have $x_i = \lambda y_i$, for every i $(1 \leq i \leq 4)$. This is denoted by $\mathtt{x} \sim \mathtt{y}$. Under a transformation represented by a matrix $\mathbf{H}$ points are transformed linearly: $\mathtt{x} \mapsto \mathtt{x}' \sim \mathbf{H}\mathtt{x}$. Note that matrices $\mathbf{H}$ and $\lambda\mathbf{H}$ with λ a nonzero scalar represent the same transformation.

In projective 3–D space $\mathcal{P}^3$ a point $\mathtt{M} = [X\,Y\,Z\,W]^\top$ is located on a plane Π,

which is also represented by a 4-vector, if and only if

$$\Pi^\top \mathtt{M} = 0 \ . \tag{B.1}$$

Therefore, in $\mathcal{P}^3$ planes are the dual entities of points.

On the other hand, in the projective plane $\mathcal{P}^2$ a point $\mathtt{m} = [x\, y\, w]^\top$ is located on a line $\mathtt{l}$, which is also represented by a 3-vector, if and only if

$$\mathtt{l}^\top \mathtt{m} = 0 \ . \tag{B.2}$$

Therefore a line is the dual of a point in 2–D projective plane. A line $\mathtt{l}$ passing through two points $\mathtt{m_1}$ and $\mathtt{m_2}$ is given by their vector product $\mathtt{m_1} \times \mathtt{m_2}$:

$$\mathtt{l} \sim [\mathtt{m}_1]_\times \mathtt{m}_2 \text{ with } [\mathtt{m}_1]_\times = \begin{bmatrix} 0 & w_1 & -y_1 \\ -w_1 & 0 & x_1 \\ y_1 & -x_1 & 0 \end{bmatrix} \ . \tag{B.3}$$

The duality of lines and points implies that the intersection of two lines is obtained through vector product of two lines.

B.2 Transformations

Linear transformations are represented by a matrix T. Under such a transformation a point is transformed as follows:

$$\mathtt{m} \mapsto \mathtt{m}' \sim \mathbf{T}\mathtt{m} \ . \tag{B.4}$$

The point transformation T transforms the planes in 3–D space $\mathcal{P}^3$ in the following manner:

$$\mathtt{M} \mapsto \mathtt{M}' \sim \mathbf{T}\mathtt{M} \tag{B.5}$$

$$\Pi \mapsto \Pi' \sim \mathbf{T}^{-\top}\Pi \tag{B.6}$$

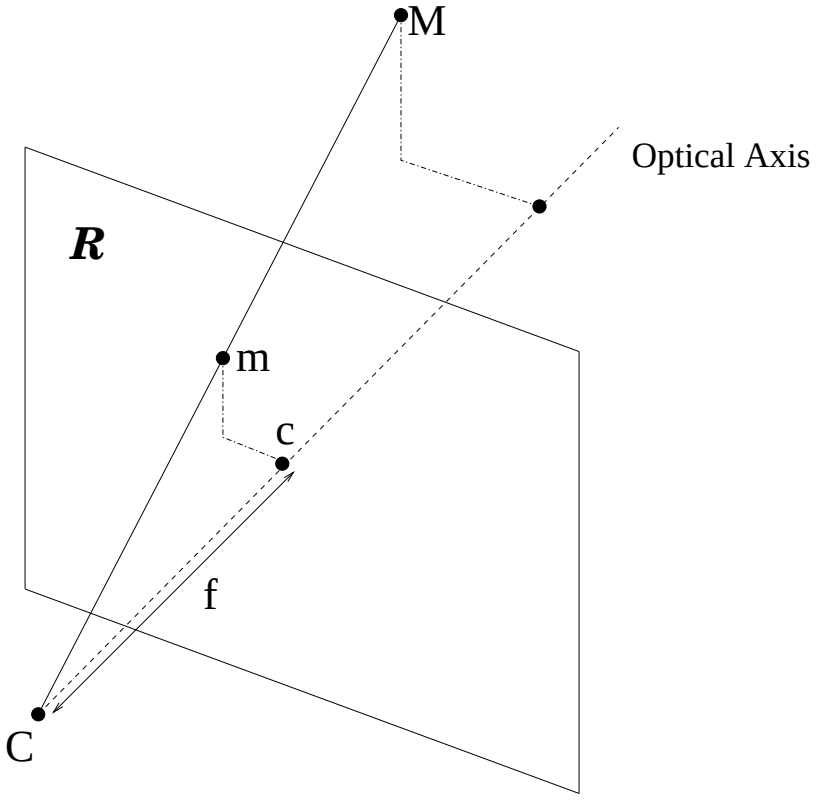


Figure B.1: Perspective projection principles.

B.3 Camera Model

The perspective camera model can be represented by an ideal pinhole camera. The geometric process for image formation in a pinhole camera is illustrated in Fig. B.1. The process is completely determined by choosing a perspective projection center and a retinal or image plane. The projection of a scene point is then obtained as the intersection of a line passing through this point and the center of projection $\mathtt{C}$ with the retinal plane $\mathcal{R}$.

The optical axis passes through the center of projection $\mathtt{C}$ and is orthogonal to the retinal plane $\mathcal{R}$. It's intersection with the retinal plane is defined as the principal point $\mathtt{c}$.

B.4 The Projection Matrix

The image of a world point on a pinhole camera is obtained through the following expression:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathtt{t} \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{B.7}$$

with f_x, f_y being the focal length in x and y directions and (c_x, c_y) the coordinates of the principal point $\mathtt{c}$ and with a specific position and orientation defined by $\mathbf{R}$ and $\mathtt{t}$ respectively.

Eq. B.7 can be simplified to

$$\mathtt{m} \sim \mathbf{K} \left[\mathbf{R}^\top \text{-} \mathbf{R}^\top \mathtt{t}\right] \mathtt{M} \tag{B.8}$$

or even

$$\mathtt{m} \sim \mathbf{P}\mathtt{M} \ . \tag{B.9}$$

The 3×4 matrix $\mathbf{P}$ is called the camera projection matrix.

B.5 Projection of a Quadric

A quadratic surface can be represented by a 4×4 matrix $\mathbf{Q}$ which can be decomposed to the product of matrices containing shape and position and orientation parameters:

$$\mathbf{Q} = \mathbf{A}^\top \mathbf{Q}' \mathbf{A} \tag{B.10}$$

with $\mathbf{A} = [\mathbf{R} \,|\, \mathbf{t}]$ is the transformation from world coordinate system to the canonical reference system of the quadric and

$$\mathbf{Q}' = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{B.11}$$

contains the shape parameters of the quadric. Given a projection matrix of the form $\mathbf{M}$, an image point $\mathbf{p} = [\ u \quad v \quad 1\ ]'$ defines a line of sight $\mathbf{l} \mapsto \lambda \tilde{\mathbf{m}} \mathbf{p} + \mathbf{O}$

in 3–D space, where $\mathbf{O}$ is the optical center of the camera and $\tilde{\mathbf{m}}$ is the inverse of the 3×3 leftmost submatrix of $\mathbf{M}$. Therefore, the trajectory of the projection of the tangent line to the quadric, i.e. the outline of the quadric, is obtained by calculating the double root of quadratic equation given by the intersection of the ray $\mathbf{l}$ and the quadric. Mathematically speaking, a 3–D point $\mathbf{P} = \lambda\tilde{\mathbf{m}}\mathbf{p} + \mathbf{O}$ on the ray $\mathbf{l}$ is on the quadric $\mathbf{Q}$ if and only if $\mathbf{P}^\top\mathbf{Q}\mathbf{P} = 0$.

This implies a quadratic equation in λ which is parametrized by u and v. Forcing this equation to have a double root will yield the equation of the conic which is the trajectory of the projection of the silhouette on the image plane. The conic curve is in the form of general bivariate quadratic curve and it can be written as

$$au^2 + 2buv + cv^2 + 2du + 2fv + g = 0 \tag{B.12}$$

This equation can be written in homogeneous system as:

$$\mathbf{p}\,\mathbf{C}\,\mathbf{p}^\top = 0 \tag{B.13}$$

with

$$\mathbf{C} = \begin{bmatrix} a & b & d \\ b & c & f \\ d & f & g \end{bmatrix} . \tag{B.14}$$

The conics can be classified into the several curves by defining the following quantities:

$$\begin{aligned} \Delta &= \begin{vmatrix} a & b & d \\ b & c & f \\ d & f & g \end{vmatrix} \\ J &= \begin{vmatrix} a & b \\ b & c \end{vmatrix} \\ I &= a + c \\ K &= \begin{vmatrix} a & d \\ d & g \end{vmatrix} + \begin{vmatrix} c & f \\ f & g \end{vmatrix} . \end{aligned} \tag{B.15}$$

Using these parameters we can classify the conic curve into the following categories[1]:

[1] source: www.mathworld.com

curve	Δ	J	Δ/I	K
coincident lines	0	0		0
ellipse (imaginary)	$\neq 0$	> 0	> 0	
ellipse (real)	$\neq 0$	> 0	< 0	
hyperbola	$\neq 0$	< 0		
intersecting lines (imaginary)	0	> 0		
intersecting lines (real)	0	< 0		
parabola	$\neq 0$	0		
parallel lines (imaginary)	0	0		> 0
parallel lines (real)	0	0		< 0

In the case of real intersecting lines (e.g. projection of truncated cones) the equation of these lines are obtained by eigenvalue decomposition of the conic $\mathbf{C} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ where $\mathbf{U} = [\ \mathbf{V_1}^\top \mathbf{V_2}^\top \mathbf{V_3}^\top\ ]$ is the matrix of eigenvectors $\mathbf{V_i}^\top$ and D is a diagonal matrix containing the eigenvalues of $\mathbf{C}$. After some manipulation the conic equation of Eq. B.13 yields the following.

$$[(\mathbf{V_1}^\top + \sqrt{-\frac{\lambda_1}{\lambda_2}}\,\mathbf{V_2}^\top)\cdot\mathbf{X}][(\mathbf{V_1}^\top - \sqrt{-\frac{\lambda_1}{\lambda_2}}\,\mathbf{V_2}^\top)\cdot\mathbf{X}] = 0 \qquad \text{(B.16)}$$

where λ_1 and λ_2 are the nonzero eigenvalues associated with $\mathbf{V_1}$ and $\mathbf{V_2}$. Eq. B.16 gives the equations of the lines that are the projection of the outlines of the truncated cone.

In the case of projection of a cylinder, its outline is given by the two eigenvectors associated with the nonzero eigenvalues.

Appendix C

The Distance Between Distributions

Measuring the difference between two textures on two sides of a potential boundary is a reliable tool to verify and analyze the quality of the boundary. We use this tool to evaluate different methods of texture transition detection in Chapter 7 and to adaptively update the texture model of the object to be tracked in Section 4.5. Different metrics are available in literature for non parametric and parametric texture representations. A survey of dissimilarity distances and their characteristics can be found in Rubner *et al.* (2001). The distance between two texture representations can be categorized in measures for parametric and non parametric models of textures. In this section we introduce some of these measures that we use and refer to in this thesis. We also provide illustrative examples of how distance between two distributions can be used to express the quality of a boundary between image regions.

C.1 Parametric Approaches:

Here we consider the simple Gaussian model for the two distributions $K_1(\mu_1, \sigma_1)$ and $K_2(\mu_2, \sigma_2)$. Where μ_1 and μ_2 are means and σ_1 and σ_2 are standard deviations of the two distributions. Among these methods we mention only Fisher's metric

in this category which is defined as

$$D(K_1, K_2) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{C.1}$$

and measures the intraclass coherency and interclass separation at the same time.

C.2 Non Parametric Measures:

Non parametric models for the texture are expressed in terms of histograms. In the following definitions, the histogram entry $K(n; T_i)$ corresponds to the number of pixels in bin n in image patch or distribution T_i. Some of the dissimilarity measures for histograms are the following.

- Minkowski-form distance is defined based on the L_p norm:

$$D(K_1, K_2) = \left(\sum_n |K_1(n; T_1) - K_2(n; T_2)|^p\right)^{1/p} \tag{C.2}$$

  where $K_1(n; T_1)$ and $K_2(n; T_2)$ are two distributions and $1 \leq p \leq \infty$. L_1 computes the city block or Manhattan distance, L_2 is the Euclidean distance and finally L_∞ measures the maximal difference.

- Bhattacharyya distance measures the statistical separability of classes, leading to an estimate of the probability of correct classification. For two given distributions $K_1(n; T_1)$ and $K_2(n; T_2)$ this distance is given by:

$$d(K_1, K_2) = \left(1 - \sum_{n=1}^{N} \sqrt{K_1(n; T_1) * K_2(n; T_2)}\right)^{1/2} . \tag{C.3}$$

  The Bhattacharyya distance provides a measure of pattern similarity using the correlation between the bin contents. This implies that empty bins do not contribute to the distance.

- The χ^2 statistic measures the likeliness of one distribution being drawn from another one and is given by:

$$D(K_1, K_2) = \sum_n \frac{(K_1(n; T_1) - K(n))^2}{K(n)} \tag{C.4}$$

where $K(n) = [K_1(n;T_1) + K_2(n;T_2)]/2$ is the mean histogram. The main disadvantage of χ^2 distance is that it yields inaccurate results with low number of observations. Moreover no bin in the histograms can have zero counts.

- The Kullback-Leibler divergence (KLD) or relative entropy is a quantity which measures the difference between two probability distributions. It is expressed as:

$$L = -\int p(x) \ln \frac{\tilde{p}(x)}{p(x)} dx \tag{C.5}$$

it can be shown that $L \geq 0$. For two discrete distribution the integration becomes a summation over the bins of $K_1(n;T_1)$ and $K_2(n;T_2)$.

- The Quadric Form (QF). The histogram quadratic distance is designed to measure the weighted similarity between histograms Niblack *et al.* (1993). The quadratic distance provides better results than "like-bin" only comparisons at the price of high computational costs. The quadratic distance between histograms and is given by:

$$D(K_1, K_2) = \left[(K_1(n;T_1) - K_2(n;T_2))^t \, A \, (K_1(n;T_1) - K_2(n;T_2)) \right]^{1/2} \tag{C.6}$$

where $A = [a_{ij}]$ is a $N \times N$ matrix and a_{ij} is the similarity coefficient between indices (bins) i and j. aij is given by $a_{ij} = 1 - d_{ij}/d_{max}$ and $d_{ij} = |K_1(i;T_1) - K_2(j;T_2)|$ and denotes the similarity between bins i and j.

- The Earth Mover's Distance (EMD): This distance considers each value of one distribution as a quantity to be moved (earth) to the other distribution (holes). The EMD is thus defined as the minimum cost of transferring one distribution to another one. The computation of EMD comes from a well-known transportation problem (Rubner *et al.*, 2001). The advantage of EMD is that it works on distributions with a different number of bins. The most important advantage of EMD is that it can be used as a true metric.

Different measures presented above should be employed based on the nature of the problem at hand. In the case of texture boundary detection for tracking, usually we should resort to distances which are accurate even with small number of observations. For small sample sizes, parametric distances such as Fisher which have few number of parameters to estimate perform best provided that the Gaussian model assumption is valid. These distances only estimate means and variances and are less sensitive to sampling noise. EMD also performs well for small sample size. In general, marginal distributions do better than multidimensional distributions when we have few samples. On the other hand distances like χ^2 and KL perform better with large number of observations. In terms of speed parametric distances such as Fisher are the fastest while EMD requires heavy computations.

C.3 Measure of Confidence

The texture distance measures described above can be used to evaluate a measure of confidence in the detected boundary. We show by a few examples that they can be used to provide mathematical measures of the quality of the boundary points in terms of visual perception.

Fig. C.1 shows Bhattacharyya distances between the distributions on both sides of hand marked silhouette of a Swiss cow (a) and the hedge boundary (c). The distances for boundaries in Fig. C.1-(a) and (c) are shown (b) and (d) respectively. The textures on both side of the cow silhouette are (visually) different while in the case of the hedge the boundary is not visually well-defined. We choose Bhattacharyya distance because it varies between 0 and 1 and therefore is self normalized. It can be observed that for the case where the textures on both sides are distinguishable, the distance is close to 1 while for the complex texture case the mean value of the distance falls well below 1.

In the next example we compare the Bhattacharyya distance between distributions on both sides of boundary points detected using the Bayesian scanline method of Section 4.1 as shown in Fig. C.2-(b). The scanlines are vertical lines passing through the two Brodatz textures shown in Fig. C.2-(a). The detected

boundary points are marked by red dots. For the purpose of comparison the dotted line in Fig. C.2-(b) shows the distance between two region selected from the upper texture while the solid thick line shows the Bhattacharyya distance for two regions selected from upper and lower textures. We can observe that in general the distances for the detected boundaries on scanlines are closer to the distance between two regions selected from different textures (solid line). Therefore we can eliminate outliers by comparing the distance for the detected points with the distances between two similar and two different textures. This measure of confidence can be improved using a better learning scheme than the ad-hoc try explained here and it can be applied to any kind of distribution model.

Furthermore, the maximum likelihood framework can be used in conjunction with a training set of textures and their distances obtained using the metrics described above, in order to measure the dissimilarity of two texture. For this purpose we need to label the distance on the training examples into "similar" and "non-similar" textures and assume a prior distribution model (e.g. Gaussian) for each label. Assuming equal chance of having similar and unsimilar textures and using the Bayes formula we can calculate the posterior probability of error in boundary detection.

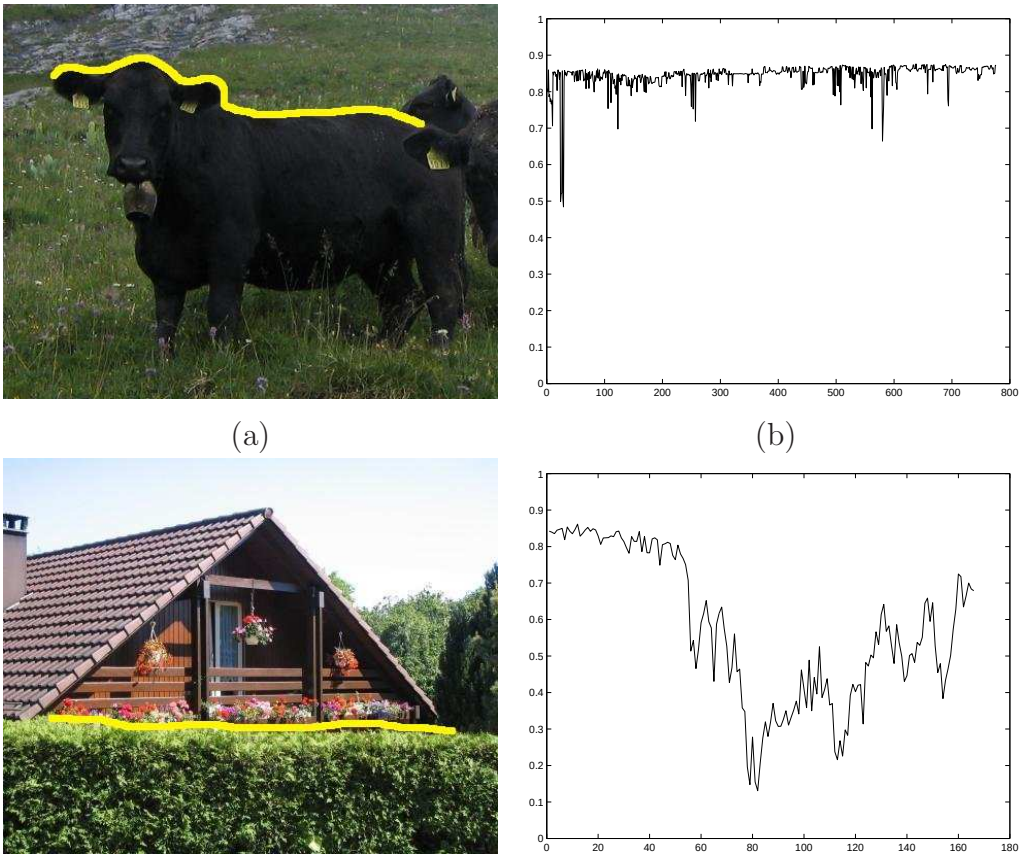
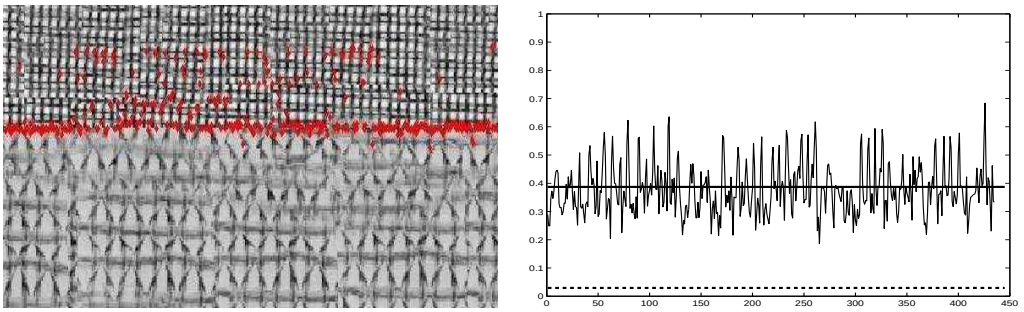


Figure C.1: Bhattacharyya Distance for different and similar texture on both sides of hand marked boundary points. The x-axis corresponds to the points along the detected boundary. The y-axis shows the Bhattacharyya distance of pixel intensity distributions on both sides of each detected point on the scanline that passes through it.



(a) (b)

Figure C.2: Detected boundaries on vertical scanlines. (a) Test image, (b) distance between textures on both sides of the detected boundary points. The dotted line shows distance between two regions selected from the upper texture and the thick solid line shows the distance between two textures selected from the upper and lower regions. It can be observed that the distance found on boundary points is close to the distance between two texture selected from the two textures. This idea can be used to measure the quality of the detected boundary.

Bibliography

Agarwal, A. & Triggs, B. (2004a). 3d human pose from silhouettes by relevance vector regression. In *Conference on Computer Vision and Pattern Recognition*. 1.3, 3

Agarwal, A. & Triggs, B. (2004b). Tracking articulated motion with piecewise learned dynamical models. In *European Conference on Computer Vision*, III 54–65, Prague. 3

Athitsos, V. & Sclaroff, S. (2003). Estimating 3d hand pose from a cluttered image. In *Conference on Computer Vision and Pattern Recognition*, 432–439, Madison, WI. 1.3, 3

Balcisoy, S., Kallmann, M., Torre, R., Fua, P. & Thalmann, D. (2001). Interaction Techniques with Virtual Humans in Mixed Environment. In *International Symposium on Mixed Reality*, Yokohama, Japan. 2.4

Blake, A., Rother, C., Brown, M., Perez, P. & Torr, P. (2004). Interactive image segmentation using an adaptive gmmrf model. In *European Conference on Computer Vision*, vol. 1, 428–441. 3.2.2

Boykov, Y. & Jolly, M. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *International Conference on Computer Vision*, Vol I:105–112, Vancouver, Canada. 3.2.2

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**, 123–140. 5.1, 5.1.1

BIBLIOGRAPHY

Breiman, L. (1998). Arcing classifiers. The Annals of Statistics, 26(3), 801–849. 5.1

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **8**. 3.1.4, 3.3

Cantzler, H. (2005). Random Sample Consensus (RANSAC). CVonline. 6.1

Comport, A., Marchand, E. & Chaumette, F. (2003). A Real-Time Tracker for Markerless Augmented Reality. In *International Symposium on Mixed and Augmented Reality*, Tokyo, Japan. 2, 2.1

Conners, R. & Harlow, C. (1980). A theoretical comparison of texture algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **2**. 3.1.2

Cover, T.M. & Thomas, J. (1991). *Elements of Information Theory*. Wiley Interscience Press, New York. 3.1.4

Davis, J. & Bobick, A. (1998). A robust human-silhouette extraction technique for interactive virtual environments. In *Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, 12–25, Geneva, Switzerland. 1.3

Deriche, R. & Faugeras, O. (1990). Tracking Line Segments. *Image and Vision Computing*, **8**, 271–270. 2.3

DeValois, R. & DeValois, K. (1990). *Spatial Cision*. Oxford University Press. A

Drummond, T. & Cipolla, R. (2001). Real-time tracking of highly articulated structures in the presence of noisy measurements. In *International Conference on Computer Vision*, Vancouver, Canada. 1.4.1

Drummond, T. & Cipolla, R. (2002). Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**, 932–946. (document), 2, 2.1, 2.2, 2.2, 3, 4, 7.5, 8.2.1

Evans, R. (1990). Filtering of Pose Estimates Genrated by the RAPiD Tracker in Applications. In *British Machine Vision Conference*, 79–84, Oxford. 2.1

Fischler, M. & Bolles, R. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications ACM*, **24**, 381–395. 6.1

Fleuret, F. & Geman, D. (2002). Fast Face Detection with Precise Pose Estimation. In *Proceedings of International Conference on Pattern Recognition*, I:235–238. 3.3

Forney, G.D. (1973). The Viterbi algorithm. In *Proceedings of IEEE*, vol. 61, 268–278. 1

Freund, Y. & Schapire, R. (1996). Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, 148–156, Morgan Kaufmann. 5.1.2

Gennery, D. (1992). Visual Tracking of Known Three-Dimensional Objects. *International Journal of Computer Vision*, **7**, 243–270. 2, 2.3

Gupta, H., Roy-Chowdhury, A. & Chellappa, R. (2004). Contour based 3d face modeling from a monocular video. In *British Machine Vision Conference*, 367–376, Kingston University, England. 3

Hanek, R. & Beetz, M. (2004). The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria. *Int. J. Comput. Vision*, **59**, 233–258. 3.1.4

Harris, C. (1992). *Tracking with Rigid Objects*. MIT Press. 2, 2.1, 2.1

Heikkila, M., Pietikainen, M. & Heikkila, J. (2004). A texture-based method for detecting moving objects. In *British Machine Vision Conference*, 187–196, Kingston University, England. 3.1.1

Ilić, S. & Fua, P. (2005). Implicit Meshes for Surface Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, accepted for publication. 8.3

Julesz, B. (1981). Textons, the elements of texture perception, and their interactions. *Nature*, **290**, 91–97. 3.1.4

Kaucic, R. & Blake, A. (1998). Accurate, real-time, unadorned lip tracking. In *International Conference on Computer Vision*, 370–375. 7.5

Koller, D., Daniilidis, K. & Nagel, H.H. (1993). Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes. *International Journal of Computer Vision*, **10**, 257–281. 2, 2.3, 2.3

Kollnig, H. & Nagel, H.H. (1997). 3D Pose Estimation by Directly Matching Polyhedral Models to Gray Value Gradients. *International Journal of Computer Vision*, **23**, 283–302. 2.4

Kolmogorov, V. & Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**, 147–159. 3.2.2

Konishi, S., Yuille, A.L., Coughlan, J. & Zhu, S.C. (1999). Fundamental bounds on edge detection: An information theoretic evaluation of different edge cues. In *Conference on Computer Vision and Pattern Recognition*, 573–579. 3.1.4

Kosaka, A. & Nakazawa, G. (1995). Vision-Based Motion Tracking of Rigid Objects Using Prediction of Uncertainties. In *International Conference on Robotics and Automation*, 2637–2644, Nagoya, Japan. 2, 2.3, 2.3

Kumar, M.P., Torr, P.H.S. & Zisserman, A. (2005). OBJ CUT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*. 3.2.2

Langley, P., Iba, W. & Thompson, K. (1992). An analysis of bayesian classifiers. In *Proceedings of AAAI-92*, 223–228. 5.6

Lee, K.L. & Chen, L.H. (2001). Unsupervised Texture Segmentation by Determining the Interior of Texture Regions Based on Wavelet Transform. *International Journal of Pattern Recognition and Artificial Intelligence*, **15**, 1231–1250. 3

Leung, T. & Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, **43**, 29–44. 3.1.4

Lowe, D.G. (1992). Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, **8**, 113–122. 2, 2.3

M. Armstrong and A. Zisserman (1995). Robust Object Tracking. In *Proceedings of Asian Conference on Computer Vision*, 58–62. 2, 2.2

Mäenpää, T. (2003). *The local binary pattern approach to texture analysis-extensions and applications*. University of Oulu, Oulu Finland. 3.1, 3.1.3, 3.1.4, A, A

Malik J, S.J..L.T., Belongie S (1999). Textons, contours and regions: cue integration in image segmentation. In *International Conference on Computer Vision*, vol. 2, 918–925, Kerkyra, Greece. 3.1.4

Marchand, E. & Chaumette, F. (2002). Virtual Visual Servoing: A Framework for Real-Time Augmented Reality. In *Computer Graphics Forum*, 289–298, Sarrebruck, Allemagne. 2.2

Marchand, E., Bouthemy, P. & Chaumette, F. (2001). A 2d-3d model-based approach to real-time visual tracking. *Journal of Image and Vision Computing*, **19**, 941–955. 2, 2.1, 2.2, 2.4

Martin, D., Fowlkes, C. & Malik, J. (2004). Learning to detect natural image boundaries using local brightn ess, color and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**. (document), 3.5, 3.3, 3.3, 3.6

Meila, M. & Shi, J. (2001). A random walks view of spectral segmentation. AISTATS. 3.2.1

Mittal, A., Zhao, L. & Davis, L.S. (2003). Human body pose estimation using silhouette shape analysis. In *IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS'03)*, 263, Miami, Florida. 1.3

Mojsilovic, A., Kovacevic, J., Kall, D., Safranek, R. & Ganapathy, S. (2000). Matching and retrieval based on the vocabulary and grammar of color patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**, 38–54. A

Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Petkovic, D., Yanker, P. & Faloutsos, C. (1993). The qbic project: Querying images by content using color, texture, and shape. In *In Storage and Retrieval for Image and Video Databases*, vol. SPIE Vol. 1908. C.2

Nitzberg, M., Mumford, D. & Shiota, T. (1993). *Filtering, segmentation, and depth*, vol. 662. Springer-Verlag. 3.1.4

Ojala, T., Pietikäinen, M. & Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, **29**, 51–59. 3.1.1

Ozyildiz, E., Krahnstoever, N. & Sharma, R. (2002). Adaptive texture and color segmentation for tracking moving objects. *Pattern Recognition*, **35**, 2013–2029. 3.2.2, 3.4

Paragios, N. & Deriche, R. (2001). Coupled Geodesic Active Regions for Image Segmentation: A Level Set Approach. In *European Conference on Computer Vision*, vol. II, 224–240. 3.1.4

Paragios, N. & Deriche, R. (2002). Geodesic Active Regions and Level Set Methods for Supervised Texture Segmentation. *International Journal of Computer Vision*, 223–247. 3.1.4

Paragios, N. & Deriche, R. (2005). Geodesic Active Regions and Level Set Methods for Motion Estimation and Tracking. *Computer Vision and Image Understanding*, 259–282. 3.1.4

Paragios, N. & Ramesh, V. (2001). A MRF-based Real-Time Approach for Subway Monitoring. In *Conference on Computer Vision and Pattern Recognition*, vol. I, 1034–1040. 3.2.2

Patel, D. & Stonham, T. (1992). Texture image classification and segmentation usin rankorder clustering. In *International Conference on Pattern Recognition*, vol. 3, 92–95, Jerusalem, Israel. 3.1.3

Perona, P. & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 629–639. 3.1.4

Pietikäinen, M., Nurmela, T., Mäenpää, T. & Turtinen, M. (2003). View-based recognition of 3-D textured surfaces. In *International Conference on Image Analysis and Processing*, 530–535, Mantova, Italy. 3.1.4

Poirson, A. & Wandell, B. (1996). Pattern-color separable pathways predict sensitivity to simple colored patterns. In Vision Research, vol. 36, no. 4, pp. 515–526. A

Qiu, H. & Hancock, E.R. (2005). Image segmentation using commute times. In *British Machine Vision Conference*, 929–938, Oxford, England. 3.2.1

Randen, T. & Husoy, J. (1999). Filtering for texture classification: a comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**, 291–310. 3.1.4

Rätsch, G., Onoda, T. & Müller, K.R. (1998). Regularizing adaboost. In *Neural Information Processing Systems*, 564–570, MIT Press. 5.1.2, 5.4

Rivera, M. & Gee, J.C. (2004). Two–level mrf models for image restoration and segmentation. In *British Machine Vision Conference*, 809–818, Kingston University, England. 3.2.2

Rother, C., Kolmogorov, V. & Blake, A. (2004). "GrabCut": Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics*, **23**, 309–314. 3.2.2

Rubner, Y. & Tomasi, C. (1996). Coalescing texture descriptors. Proc. ARPA Image Understanding Workshop. 3, 3.1.4

Rubner, Y., Puzicha, J., Tomasi, C. & Buhmann, J.M. (2001). Empirical evaluation of dissimilarity measures for color and texture. *Comput. Vis. Image Underst.*, **84**, 25–43. 3.1.4, 7.6.1, C, C.2

Ruf, A., Tonko, M., Horaud, R. & Nagel, H.H. (1997). Visual Tracking by Adaptive Kinematic Prediction. In *Proceedings of International Conference on Intelligent Robots and Systems*, 893–898. 2, 2.3

Ruzon, M. & Tomasi, C. (1999a). Color Edge Detection with Compass Operator. In *Conference on Computer Vision and Pattern Recognition*, vol. 2, 160–166. 3.1.4

Ruzon, M. & Tomasi, C. (1999b). Corner Detection in Textured Color Images. In *International Conference on Computer Vision*, 1039–1045. 3.1.4

Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**, 888–905. 3.2.1

Shimada, N., Shitirai, Y., Kuono, Y. & Miura, J. (1998). Hand Gesture Estimation and Model Refinement using Monocular Camera – Ambiguity Limitation by Inequality Constraints. In *Automated Face and Gesture Recognition*, 268–273, Nara, Japan. 8.1.1

Simon, G. & Berger, M.O. (1998). A two-stage robust statistical method for temporal registration from features of various type. In *International Conference on Computer Vision*, 261–266, Bombay, India. 2.2

Sminchisescu, C. & Triggs, B. (2003). Kinematic Jump Processes for Monocular 3D Human Tracking. In *Conference on Computer Vision and Pattern Recognition*, vol. I, 69, Madison, WI. 1.3

Taylor, C., Malik, J. & Weber, J. (1996). A real-time approach to stereopsis and lane-finding. In *Intelligent Vehicles*, 207–212. 3

Thayananthan, A., Torr, P. & Cipolla, R. (2004). Likelihood models for template matching. In *British Machine Vision Conference*, 949–958. 3

Tomita, F. & Tsuji, S. (1990). Computer analysis of visual textures. Kluwer. 3.1.2

Tse, P.U. & Hughes, H.C. (2004). Visual form perception. Entry in the Encyclopedia of Neuroscience. Adelman, G. and Smith, B. (Eds.). Elsevier. A, A, A

Vacchetti, L., Lepetit, V. & Fua, P. (2004). Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *International Symposium on Mixed and Augmented Reality*, Arlington, VA. 2, 2.2

Valkealahti, K. & Oja, E. (1998). Reduced Multidimensional Cooccurrence Histograms in Texture Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 90–94. 3.1.2

Varma, M. & Zisserman, A. (2002). Classifying materials from images: to cluster or not to cluster. In *2nd International Workshop on Texture Analysis and Synthesis*, 139–143. 3.1.4

Viola, P. & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. In *Conference on Computer Vision and Pattern Recognition*, 511–518. 3.3

Wandell, B.A. (1995). Foundation of vision. Stanford University, Sinauer Associates, Sunderland, Mass. A

Warakagoda, N.D. (1996). *A Hybrid ANN-HMM ASR system with NN based adaptive preprocessing*. Master's thesis, Norges Tekniske Hogskole, Institutt for Teleteknikk, Transmisjonsteknikk. 6.2.1, 2, 3

Weszka, J., Dyer, C. & Rosenfeld, A. (1976). A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man, and Cybernetics*, **6**, 269–285. 3.1.2

Will, S., Hermes, L., Buhmann, J. & Puzicha, J. (2000). On learning texture edge detectors. In *International Conference on Image Processing*, vol. III, 877–880, Vancouver, Canada. 3.3

Williams, L.R. & Jacobs, D.W. (1997). Stochastic completion fields: a neural model of illusory contour shape and salience. *Neural Comput.*, **9**, 837–858. (document), A, A.6

SEYED ALI SHAHROKNI

Date of Birth: 20/09/1976
Marital status: single

Rue du Centre 2B
1025 st-Sulpice
Switzerland

tel: (+41) (0)78 722 88 59
fax: (+41) (0)21 693 75 20
email: ali.shahrokni@epfl.ch
http://cvlab.epfl.ch/∼ali

Education

Ph.D. In Computer Science: Computer Vision
Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland
2001–(graduation date 12/2005)
Thesis subject: Probabilistic Modeling of Texture Transition for Fast Tracking and Delineation

M.Sc. in Communications Systems Engineering
University of Tehran, Tehran, Iran, 1998–2000
Thesis subject: volumetric data quantification, visualization and feature extraction in confocal microscopic images

B.Sc. in Electronics Engineering
University of Tehran, Tehran, Iran, 1994–1998

Experience

Tracking and surveillance

- Developed a 3–D markerless monocular tracking of human motion system (European Union joint research project VIBES: Video Browsing, Exploration and Structuring system)
- Assisted in development of an automatic vehicle tracking using a mobile camera, electronics Lab. Amir Kabir university of technology, Tehran, Iran, Sept. 2000–May 2001
- Implemented a vision-based vehicle tracking program, DSP Lab. university of Tehran, Tehran, Iran, April 1998–November 1998

Statistical learning

- Developed a texture-based segmentation method using Markov models and different pattern recognition techniques

Biomedical engineering

- Developed a method for volumetric data quantification, visualization and feature extraction in confocal microscopic images (also master's thesis), Institute for studies in theoretical Physics and Mathematics (IPM),Tehran, Iran, 1999–2001

Vision for sports

- Visual analysis of sports activities. Assisted in design and development of a golf swing tracking system (project partly funded by Dart Fish Co. LTD.) June-Dec. 2001

Advised student diploma projects at EPFL

- Background subtraction on sequences with a moving camera, spring semester 2003
- Pose estimation for real-time tracking, spring semester 2003
- Texture based tracking, winter semester 2003
- Application of visual tracking results to animate a 3dSMax character, winter semester 2003

Management

- Founding chairman of the first IEEE student branch in Iran at University of Tehran, 1997
- Entrepreneurship course, Venture lab, Switzerland, spring 2005

Languages

Persian native language **English** fluent **French** fluent
German basic knowledge **Arabic** basic knowledge

Honors

- Awarded IEEE region 8 Dr. Larry k. Wilson student activities prize (1998)
- Ranked 15 among over 200,000 participants in nation-wide university entrance exam
- Invited visiting researcher at university of Cambridge, summer 2004

Technical proficiency

Languages	C/C++ ($> 100,000$ lines of code), Delphi, MaxScript, Java, CSS, XML
APIs	MatLab(advanced), Maple, OpenGL, 3dSMax, QT, Maya (Basic), Open Inventor (Basic)
Systems	Linux, Unix , Sun,Windows XP
Others	LaTeX, CVS, MS .NET, OpenCV

References Available upon request

Hobbies Sailing
Ice skating and winter sports

Publications

1. A. Shahrokni, T. Drummond, P. Fua. **Fast Texture-Based Tracking and Delineation Using Texture Entropy**. In proceedings of International Conference on Computer Vision, Beijin, China, 2005
2. A. Shahrokni, F. Fleuret, P. Fua. **Classifier-based Contour Tracking for Rigid and Deformable Objects**. In proceedings of British Machine Vision Conference, Oxford, England 2005
3. A. Shahrokni, V. Lepetit, T. Drummond, P. Fua. **Markov-based Silhouette Extraction for Three–Dimensional Monocular Body Tracking in Presence of Cluttered Background**. In proceedings of British Machine Vision Conference, Kingston, England 2004
4. A. Shahrokni, T. Drummond, P. Fua. **Texture Boundary Detection for Real-Time Tracking**. In proceedings of European Conference on Computer Vision, Prague, Czech Republic, May 2004
5. V. Lepetit, A. Shahrokni, P. Fua. **Robust Data Association**. In proceedings of International Conference on Computer Vision and Pattern Recognition, USA, June 2003
6. A. Shahrokni, V. Lepetit, and P. Fua. **Bundle Adjustment for Markerless Body Tracking in Monocular Video Sequences**. In ISPRS workshop on Visualization and Animation of Reality-based 3D Models, Switzerland, 2003
7. A. Shahrokni, L. Vacchetti, V. Lepetit, P. Fua. **Polyhedral Object Detection and Pose Estimation for Augmented Reality Applications**. In proceedings of Computer Animation 2002, Geneva, Switzerland
8. H. Soltanian-Zadeh, A. Shahrokni, R.A. Zoroofi. **A voxel-coding method for quantification of vascular structure from 3D images**. In rroceedings of SPIE Medical Imaging Conference, San. Diego, CA, Feb. 17-23, 2001
9. A. Shahrokni, R.A. Zoroofi, H. Soltanian-Zadeh. **A fast skeletonization algorithm for 3-D elongated objects**. In proceedings of SPIE Medical Imaging Conference, San. Diego, CA, Feb. 17-23, 2001
10. A. Behrad, A. Shahrokni, S. A. Motamedi and K. Madani. **A Robust Vision-based Moving Target Detection and Tracking System**. In proceedings of Image and Vision Computing conference (IVCNZ2001), University of Otago, Dunedin, New Zealand Nov. 26-28, 2001
11. A. Shahrokni. **Fuzzy-Based Segmentation of 2- and 3-D images**. In proceedings of the 3rd symposium on intelligent systems, University of Tehran, Tehran, Iran, April 2000