Hybrid Tracking for Augmented Reality

Pablo Berges del Arco
Supervisor: Prof. T.Ebrahimi
Resp. assistant: D. Marimon Sanjuan

April 22, 2005

Contents

List of Figures

Abstract

Accurate registration between real and virtual objects is critical for Augmented Reality (AR) applications. State of the art shows that no tracking device is individually adequate. We present a data fusion framework that combines orientation measurements of different tracker devices. It has been designed to work with a video–based tracker subsystem and an inertial tracker, but its flexibility permits it to work with orientation measurements produced by any kind and number of trackers, no matter their rate or physical configuration.

The core of this fusion system is a Kalman filter with only one process and measurement model shared by all the trackers. The system weights each tracker according to the quality of their measurements. We have tested the system with synthetic and real orientation data to evaluate its fusion capabilities and find its limitations. This analysis leads our future work to the development of a drift corrector and to extend the filter to make it dynamically adaptive.

Chapter 1

Introduction

Advances in technology of computer hardware, computer graphics and signal processing has permitted the recent birth of a new promising technology, Augmented Reality (AR). AR systems have the goal of enhancing a person's perception of the surrounding world. Adding useful information to user's vision opens the possibility to develop a great variety of applications and services for both professional and entertainment purposes.

Augmented Reality consists on combining the real scene viewed by the user and a virtual scene generated by a computer which augments the scene with additional information. The ultimate goal is to create a system such that the user can sense the virtual added images as part of the real world. To the user of this ultimate system it would appear that he is looking at a single real scene.

However, augmented reality is still far from its maturity. There is still much work to do to meet all the basic requirements of a flexible AR system. Preserving the illusion that the virtual and real world coexist demands a perfect alignment. This can only be achieved through a perfect knowledge of the user's pose and the exact pose of the virtual objects in real world coordinates. Any mis–registration will prevent the user from seeing the virtual and real scenes as fused together. Thus, the selection of appropriate motion tracker technology and elimination of registration inaccuracies during the motion tracking process are of great importance.

AR is a technology that has great potential but to fulfill its necessities a considerable amount of research on sensory information processing remains to be done. Nowadays, most projects on this field focus on developing better

tracking systems and creating new designs with lower latency and improved displays.

In particular, in the tracking field, researches are now trying to overcome limitations of individual tracker devices by combining the measurements of at least two tracking methods. This is called hybrid tracking.

1.1 Objectives and starting point

The goal of this project is to conceive and build a hybrid tracking platform. Previous work on our group has proven several weaknesses of video-based trackers. They are not suitable for fast motion and they fail when there is occlusion of the scene. This lack of continuity motivates our group to look for a stronger solution to camera pose estimation. Research in this area has proven that no individual tracking achieves high accuracy while being suitable for mobile applications. However, synergies of trackers keep the strength of each sensor while being suitable for almost every environment. These synergies are called hybrid systems.

This project aims at building a system that combines two tracking techniques: video-based tracking and inertial tracking. The former resolves fine registration on slow motion based on the video stream acquired by a camera attached to user's head. Inertial tracking measurements inclination and acceleration thanks to gyroscopes and accelerometers. This makes them self-dependant and suitable for mobile applications. Among their characteristics is the high quality given for fast motion.

The objective is to develop the platform that best uses both sensors to increase the individual quality of each tracker.

1.2 Organization of the report

The report is divided in four main parts: Theoretical Background (Part II), System design (Part II), Experiments and results (Part III), Conclusions and future work (Part IV).

The core of the fusion algorithm is the discrete Kalman filter. For the ease of the reader, the first part of this report, Part I overviews this filter's theory. Furthermore, for a better understanding of the report, there is also

a brief description of orientation theory and tracking theory background.

The specification of the overall system is exposed in Part II. The reader can find a description and a mathematical explanation of our hybrid system. In Part III, the system is tuned and tested. And finally Part IV concludes this work.

Part I

Theoretical Background

Chapter 2

Orientation Theory

When tracking the orientation of solid 3D objects, like in our project the user's head, we need a way to specify, store and calculate the orientation and subsequent rotations of this object. Rotational quantities are more difficult to represent than linear quantities. One method of holding this information is not suitable for all needs, therefore there are different ways to specify and perform this rotation. In our project the parameterizations used during the process of tracking user's head orientation are:

- Euler angles
- Quaternions
- Rotation Matrices

There are several other approaches, or parameterizations of the attitude, but these are the most suitable for our application.

2.1 Euler Angles

Euler set the basis of the Euler Angle method. In his rotation theorem he stated: "Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis" [1].

This means that we can represent an orientation with 3 numbers. A sequence of rotations around principle axes is called an Euler Angle Sequence. But there are several Euler angle conventions; the names of the angles changes depending on the area where they are used (navigation, computer graphics, ...), but also the order in which the angles are applied (in mathematical terms: rotations are not commutative), and whether the rotations are left or right handed.

Assuming we limit ourselves to 3 rotations without successive rotations about the same axis, we could use any of the following 12 sequences:

XYZ	XZY	XYX	XZX
YXZ	YZX	YXY	YZY
ZXY	ZYX	ZXZ	ZYZ

In our global project, we have chosen the one defined in the documentation of our inertial tracker *InertiaCube*^{2} (Subsection 6.2.2). We have decided to use this convention in the whole application to avoid continuous transformations from one convention to other.

Object attitudes are specified using values for Roll (ϕ), Pitch (θ) and Yaw (ψ) (see Figure 2.1). These represent a rotation of the object about the X, Y, and Z axes, respectively, to the desired orientation. The attitude is expressed as Yaw/Pitch/Roll (ψ, θ, ϕ), this means the sequence starts with a rotation by (+yaw) about the Z axis, followed by a rotation by (+pitch) about the new Y axis (i.e. body frame axis), followed by a rotation by (+roll) about the new X axis (i.e. body frame x axis).

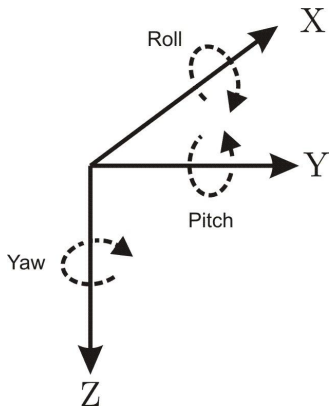

Figure 2.1: *Euler Angle Sequence*

However, one potential problem that they can suffer from is 'gimbal lock'. This results when two axes effectively line up, resulting in a temporary loss of a degree of freedom. This is related to the singularities in longitude that you get at the north and south poles.

2.2 Quaternions

Quaternions is another of the mathematical representations that has been used to symbolize the rotations. This method overcomes the 'gimbal lock' euler's problem and its better than euler angles when rotation interpolation is required.

A quaternion represents a three-dimensional rotation in space as a four-component row vector of unit length:

$$q = (W, X, Y, Z)$$

,

It is composed of one real value (W) and three imaginary values (X,Y,Z), and it represents a rotation (θ) about an arbitrary axis ($\vec{v}$); rotating around an arbitrary axis avoids gimbal lock.

$$W = cos(\theta/2)$$
$$X = v_x sin(\theta/2)$$
$$Y = v_y sin(\theta/2)$$
$$Z = v_z sin(\theta/2)$$

where:

$\vec{v}$ is the unit vector along the axis of rotation, and

θ is the total rotation angle.

A quaternion, is together with euler angles, the most common way to represent rotations and it has several advantages. First of all, as mentioned before, it resolves the euler problem of 'gimbal lock'. Moreover, its a great rotation representation when orientation interpolation is required (the other

two mathematic representations, euler angles and rotation matrix, are inappropriate) [2][1]. And finally, it requires fewer elements and constraints than most of the other parametric representations, as for example rotation matrices.

But there are also some disdvantages. Rigid body dynamics represented by quaternions can not be characterized by linear equations. When working with Kalman filter and quaternions, the process model can not be defined with linear equations, a non-linear model must be used and in consequence an Extended Kalman Filter has to be used. An Extended Kalman filter works with more complex operations, and the filtering processing time will increase in some way.

2.3 3D Rotation Matrices

Finally, in the implementation of our project, we also use a third mathematical representation of rotations: rotation matrices. The rotation matrix is the representation of choice for transforming points because only a simple and efficiently implemented matrix multiply is required. On the other hand, they are inappropriate for filtering or interpolation.

General rotations in 3D can be expressed as three successive rotations about different axes. For example, a transformation from reference axes to a new coordinate frame may be expressed as follows:

rotation ϕ about x axis,

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ cos(\phi) & sin(\phi) & \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix}$$

rotation θ about y axis,

$$R_y(\phi) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

rotation ψ about z axis,

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

By multiplying a vector representing a point by one of these matrices (with the values properly filled in), you can rotate the point around any axis. Therefore, the full transformation around the tree axis can be expressed as the product of these three separate transformations. For example, a 3d–rotation in space using the euler convention defined before (2.1) can be achieved by multiplying by the following matrix:

$$R_{\psi,\theta,\phi} = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi)$$

As the matrix multiplication is not commutative, different order in multiplication represents a different final rotation.

Chapter 3

Tracking Theory

The position (x, y, z) and orientation (yaw, pitch, roll) of the user's head is tracked by means of a tracker device. As the user looks around, the position and orientation information is continuously relayed to the host computer. Later, the computer calculates the appropriate view that the user should see in the augmented environment, and merges the virtual objects with the real image filmed with a camera. So the tracker should be accurate enough to achieve a perfect alignment of the virtual and real worlds.

Trackers that just track orientation (yaw, pitch, roll) or just position (x, y, z) are referred to as 3 degree of freedom, or 3 DoF trackers, while trackers that tracks both (pose) are referred to as 6 DoF trackers.

3.1 Tracker properties

When designing or evaluating an augmented reality system that will receive tracking information, it is important to pay attention to the latency, update rate, resolution, and accuracy of the tracking system [3]:

Latency is the "delay between the change of the position and orientation of the target being tracked and the report of the change to the computer" [4]. If the latency is greater than 50 milliseconds, it will be noticed by the user and the AR scene will not seem realistic.

Update rate is the number of times per second that the tracker device reports data to the computer, and depending on the trackers it can

vary between 20 or less and more than 1k updates per second. The pose update rate must be at least twice the true target motion bandwidth, or an estimator may track an alias of the true motion. Given that common arm and head motion bandwidth specifications range from 2 to 20 Hz [5], the sampling rate should ideally be greater than 40 Hz.

Accuracy is a measure of the error in the position and orientation reported by the tracker. in trackers based on emitters and receivers it decreases as the user moves farther from the fixed reference point [4].

Resolution is the smallest change in position and orientation that can be detected by the tracker. It will depend on the type of tracker used.

3.2 Tracker devices

Current tracking devices are based on a large number of different technologies. A short presentation of each of these approaches follows, together with a discussion of their advantages and disadvantages [3] [1].

Mechanical These devices measurement position and orientation by using a direct mechanical connection between a reference point and the target. Mechanical trackers have a very high accuracy and reliability. But they are not flexible, have a very limited range and are limited to one user. An example of such a mechanical tracking device is the Boom developed by Fake Space Labs and Fakespace[6] produces some mechanical trackers.

Optical Optical tracker aren't that robust but have a very high accuracy. And they need a powerful computer to keep tracking time short. There are two ways of tracking. Mounting a camera on a helmet and tracking fixed features (for example LEDs on the wall). Or mounting features on the helmet tracked by fixed cameras.

Acoustic Acoustic tracker systems use ultra sound waves for measuring the position and orientation of target objects. But to work properly, a large number of transponders is needed on known location. The system delay is rather high. As the optical tracker the acoustic needs line of sight to work. Existing systems are for example the Intersense IS-600 / IS-900.

Magnetic Magnetic trackers use one fixed transmitter, and sensors. They are used rather often, because they are very robust. But the error

increases with larger distances and they are also disturbed by ferromagnetic objects (like furniture) and other magnetic fields. Another disadvantage to these tracking devices is that the working volume tends to be rather small.

The most well-known producer of electromagnetic sensor technology is Polhemus. Their systems provide extremely low latency (on the order of 5 milliseconds) and have the ability to track multiple objects concurrently.

GPS The Global Positioning System (GPS) can be used for outdoor tracking. It should work on every place on earth. But its sensors need contact to the satellites. This is a problem in buildings. Its accuracy is very limited to about 10-100 meters. Differential GPS can achieve accuracy of about 0.1-1 meters. But this is by far not enough for most applications.

Inertial Accelerometers and gyroscopes are used to record users movements. Because the position can not be recorded directly but is integrated over all recorded data. The result tends to drift with time and gets imprecise. This sort of system is well suited for outdoor applications in combination with portable systems.

Vision–based Vision–based tracking is commonly employed for AR. Unlike other active and passive sensing technologies, vision methods can estimate camera pose directly from the same imagery observed by the user. The pose obtained is therefore relative to the viewed objects of interest, not a separate sensor or emitter attached to the environment. But they are low rate trackers and not as robust as other techniques (bad illumination and occlusions disturb its functioning).

3.3 Hybrid tracking

Through our investigation of tracking devices [1][6][4], we see that many different approaches have been tried, all of which have their own advantages and disadvantages. All of the trackers described are good for some environments and tasks, and fail on others. So, no tracker analyzed is completely prepared to work alone in an AR application.

Researchers are still seeking a tracking device which completely covers the requirements of an AR application. This means, a tracker with a large

working volume, high accuracy and resolution, very short lag time and high update rate.

Nowadays, a hybrid tracking system is the best solution to achieve a better pose estimation [1][6]. Hybrid trackers try to combine the strengths of at least two tracking methods, and all combinations are possible. Examples are inertial and optical tracking like in [7] or magnetic and vision–based tracking like in [8]. But hybrid systems are complex because two systems have to be handled and combined. However, they show the best results, and will be widely used in the future.

Chapter 4

Kalman Filter

The Kalman filter is the core of our fusion algorithm, and a basic knowledge is required to understand more in detail how the hybrid tracking system works. For this reason, a brief explanation of the Kalman filter theory is exposed in this chapter.

If the reader wants to go more deeply into this topic, he should read "An Introduction to the Kalman Filter" written by Greg Welch and Gary Bishop [9], and further information of its properties can be found in by Lindsay Kleeman [10].

4.1 Properties of Kalman Filtering

The Kalman filter has been extensively used in tracking systems because of its remarkable properties [10]. To begin with, a Kalman filter is an optimal estimator, this means it infers parameters of interest from indirect, inaccurate and uncertain observations. If all noise is Gaussian, the Kalman filter minimises the mean square error of the estimated parameters.

And secondly, the Kalman filter is a recursive estimator. This means that only estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state. In contrast to batch estimation techniques, no history of observations and/or measurements are required.

Kalman Filtering is so popular in tracking because its remarkable properties:

- Good results in practice due to optimality and structure.
- Convenient form for online real time processing (recursive estimator).
- Easy to formulate and implement given a basic understanding.
- Measurement equations do not need inversion.

4.2 The Process to be estimated

The Kalman filter is used to estimate the state of a dynamic system from a series of noisy measurements. In order to accomplish this, the Kalman filter employs two models.

Process model The first model is called the state transition model or process model. This model describes how the state x is expected to change from one timestep to the next. This is necessary because the system is dynamic, which means its underlying state is always subject to change. In reality this model is an approximation to the true process which is driving the dynamics of system.

$$x_k = Ax_{k-1} + w_{k-1} \tag{4.1}$$

where A is the state transition model applied to the previous state x_{k-1}.

Because the state transition model is an approximation, a process noise model is used. This model is used to mask the errors caused by the approximation.

$$p(v) \sim N(0, R) \tag{4.2}$$

Measurement model The second model is called the observation or measurement model. This model describes how the estimation space maps into the observation space. For example,

The observation is assumed to be of the form:

$$z_k = Hx_k + v_k \tag{4.3}$$

where v_k is the measurement error and it is assumed to be independent (of each other), white, and with normal probability distributions:

$$p(w) \sim N(0, Q) \tag{4.4}$$

4.3 Origins of the filter

The goal of the Kalman filter is to minimize the a posteriori error covariance equation:

$$P_k = E[e_k e_k^T] \tag{4.5}$$

where $e_k = x_k - \hat{x_k}$, is the error in estimation

To do so, the filter is designed to produce the best possible estimation of the posteriori state estimate $\hat{x}_k$ by combining the known a priori state estimate $\hat{x}_k^-$ and the new measurement z_k in an optimal way

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{4.6}$$

The difference in equation $z_k - H\hat{x}_k^-$ is called the measurement innovation, or the residual. The residual reflects the discrepancy between the predicted measurement and the actual measurement. And the matrix K is the value that will minimize the a posteriori error estimate.

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{H P_k^- H^T + R} \tag{4.7}$$

Justification of why matrix K_k minimizes the error can be found in [11]. Analyzing the formulation of K, we see that as the measurement error covariance R approaches zero, the gain K weights the residual more heavily.

$$\lim_{R_k \to 0} K_k = H^{-1}$$

A low value of R, means a low measurement error. Consequently the new measurement value is highly trusted.

On the other hand, as the a priori estimate error covariance approaches zero, the gain K weights the residual less heavily.

$$\lim_{P_k^- \to 0} K_k = 0$$

A low value of P_k^-, means that our estimation is accurate, so the predicted measurement $H\hat{x}_k^-$ is trusted more heavily.

4.4 The Discrete Kalman filter equations

The Kalman filter algorithm has two phases: Predict and Update. The predict phase uses the estimate from the previous time step to produce an estimate of the current state. In the measurement correction phase, measurement information from the current time step is used to refine this prediction.

4.4.1 Time update: Prediction

The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step.

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{4.8}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4.9}$$

4.4.2 Measurement Correction

The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{4.10}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4.11}$$

$$P_k = (I - K_k H)P_k^- \tag{4.12}$$

4.5 Filter Parameters and Tuning

The filter performance depends on the choices of tuning parameters such as initial values or weighting matrices which the designer must analyze and fix. When implementing a Kalman filter, a critical part of this design is fixing the elements of the matrices Q and R. As described in previous sections of this chapter, matrix Q corresponds the covariance of the process, and matrix R to the measurement noise covariance.

For a linear system with gaussian noise the true covariance matrices give the best performance. But these matrices are not normally easy to determine [9]:

Determining R

The matrix R represents the confidence in the measurements. The measurement noise covariance is normally fixed before the filter begins to operate. Measuring the matrix is generally possible by an off–line evaluation of the measurement error of the sensors.

A large value means that we assume that the measurements are corrupted by noise with a large covariance, and will make the gain K smaller.

Determining Q

The matrix Q represents the uncertainty in the process model. The determination of the process noise covariance Q is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple process model can produce acceptable results if one "injects" enough uncertainty into the process by selecting high value elements of matrix Q.

As Q represents the uncertainty in the process model, large values of Q (high uncertainty in the model) gives a large P and in turn a large Kalman gain K (see Section 4.3). This means that the filter adjusts the parameters faster, so the estimation adapts faster to rapid changes. The drawback is that the filter is also more vulnerable to measurement noise.

Filter tuning

Whether or not we have a theoretical basis for choosing the parameters of the matrices, a better result can be achieved by tuning the filter. Tuning consists in changing the variables Q and R to get the best possible performance. This tuning is commonly done off-line before the filter starts to work.

But it is frequently the case that tuning should be dynamic. Sometimes, neither the measurement error R nor the process covariance Q remain constant.

For instance, when using video–based trackers, the noise in measurements R will increase under bad light conditions or if the movement of the camera is too high.

Also, the process noise Q is sometimes changed dynamically during filter operation in order to adjust to different dynamics. For example, in the case of tracking the user's head movement in an AR application we should reduce the magnitude of Q if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. In such cases, Q should be changed to account for both uncertainty about user's intentions and uncertainty in the model.[9]

Chapter 5

Related Research

Augmented Reality applications requires a highly capable head–pose tracking system. For ordinary virtual environments that completely replace the real world by a virtual one, an approximate knowledge of this pose is sufficient. But in AR, where the virtual objects are superimposed in the real scene, perfect alignment of these objects is required. Preserving the illusion that the two worlds coexist demands proper alignment. This can be achieved just by a perfect knowledge of the user's head–pose.

Low latency and high accuracy tracking systems are essential conditions [12], but none of the actual trackers completely satisfies these requirements. Hybrid tracking, which combines two trackers technologies, is frequently used to improve the performance of an AR system. These hybrid tracker systems attempt to overcome the limits of any single technology.

There is a great variety of trackers and possible combinations. Magnetic trackers are the most common alternative to combine with other sensors. They offer enough precision in the data acquisition and are rather robust to external conditions (providing magnetic interferences).

State et al. [8] developed a hybrid tracking scheme which combined a fiducial–based vision and a magnetic tracker. They created a hybrid system that has the (static) registration accuracy of vision-based trackers and the robustness of magnetic trackers. Auer and Pinz [13], created a similar magnetic–vision system, in which their vision subsystem used corners as visual features. In their solution, prediction from the magnetic tracker is used to reduce the search areas in the optical tracking subsystem achieving a faster an more robust tracking. Emura et al. [14] also suggested a hybrid tracking

system consisting of a magnetic tracker and gyro sensor that compensates for the latency in the magnetic tracker.

However, magnetic trackers are not appropriate for all applications. If we want to track the user's head in unprepared environments, we will need a more flexible tracking system. In this case, the use of sensors that require pre–placing instruments, transponders or landmarks in the scene are not possible. These include most of the mechanical, magnetic, acoustic and optical sensors.

Several options for fully mobile systems already exist. GPS, inertial and vision–based trackers are the most common options in this systems because they are not restricted to prepared environments. Some developers have used GPS in their AR tracking system. In [15], Columbia's Touring Machine tracks outdoors by combining a differential GPS with a compass and till sensors. Behringer [16] also implemented an outdoor system which combined vision with GPS.

However, GPS does not inform about the user's head orientation and, furthermore it has low resolution. So this technology is not sufficient in many applications.

Inertial and video trackers based on natural features

Another popular choice in unprepared environments is inertial and natural feature video–based tracker fusion. They are not limited to prepared areas [17], and the complementary nature of these two sensors, make them good candidates for combination.

You et al. (1999) [18] created a tracking system which combined a natural-feature vision system with three gyro sensors . Their system limitation is that it just provide 3 DoF orientation tracking and not a full 6 DoF. The fusion approach is based on the SFM (structure from motion) algorithm, in which approximate feature motion is derived from the inertial data, and vision feature tracking corrects and refines these estimates in the image domain. Furthermore, the inertial data also serves as an aid to the vision tracking by reducing the search space and providing tolerance to interruptions.

A more complex solution have been recently developed by Chen and Pinz (2004) [19]. They presented a structure and motion framework for real time tracking combining inertial sensors with a vision system based on natural features. Their model uses fusion data to predict user's pose and also to estimate a sparse model of the scene without any visual markers. An EKF is used to estimate motion by fusion of inertial and vision data, and a bank

of separate filters to estimate the 3D structure of the scene.

Inertial and video trackers based on landmarks

However, inertial and video–based trackers have also been combined for indoor tracking using algorithms based on landmarks. The use of landmarks of this systems limits this technology to prepared environments. In these implementations, the scene has to be previously prepared before AR applications start to run.

In 2000, Yokokohji et al. [20] propose a method for accurate image overlay on HMDs using vision and accelerometers. Their proposed method, which is based on the Extended Kalman filter, use acceleration information for predicting the head motion and make the vision-based tracking robust. Its vision–based tracking subsystem is based on landmark detection and tracking.

You and Neumann (2001)[21] extended their previous work [18] and they developed an AR registration system integrating 3 DoF (degrees of freedom) inertial gyroscopes and vision tracking technologies. Their implementation uses an Extended Kalman Filter (EKF) with two independent processing channels, one for low–rate vision measure and another for the high–rate inertial gyro.

A very similar approach was developed by Lang et al. (2002) [22]. In their work, they presented a combination of a 6 DoF inertial system with a vision–based tracking system. Their inertial tracker consists of a combination of automotive inertial sensors (silicon micromachined accelerometers and gyroscopes)and its sensor noise and drift is corrected by a vision tracker. Their fusion system is based on an Extended Kalman Filter. The vision-based subsystem tracks visual landmarks in the scene and recovers the camera pose by solving the perspective n-point problem.

Recently, Ababsa and Mallem (2004) [23] also designed a hybrid system combining an inertial and a vision tracker. Their system consists of a robust vision landmark tracker and an inertial measurement unit (IMU) providing 3D linear acceleration and 3D rate of turn. To predict head motion, they apply a particle filter Bayesian bootstrap, instead of using the most used filter for this task, the EKF.

Part II

System design

Chapter 6

System Description

The final goal of our AR system is to merge a real image and a virtual object in the most realistic way. To do so, the relation between the user's head pose and the position of the object in the real world must be calculated with high precision. A number of hardware components and algorithm modules are used to accomplish this objective.

The augmented reality framework created, was originally designed to work just with a video-based markerless tracker. Due to the multiple weaknesses of this technology (see Section 6.2.3), an auxiliary and complementary tracker was decided to add to the initial platform to aid in the orientation estimation. The inertial module included in the system provides the necessary robustness in fast head movements and a fusion block has been constructed to combine the data of the two trackers. This orientation fusion block, is the core of the project and its design and properties are explained in detail in Chapter 7

6.1 System overview

As we can see in Figure 6.1, the AR structure consists of seven main blocks:

- video tracker
- inertial tracker
- orientation Kalman fusion filter
- position Kalman filter

- pose estimator
- virtual object database
- rendering unit

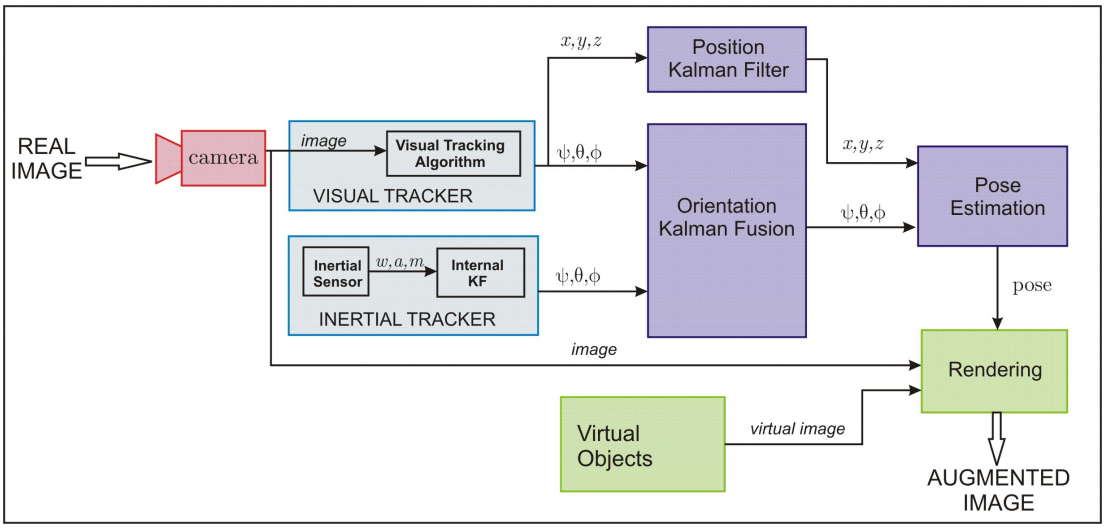

Figure 6.1: *Global Platform*

The video and inertial tracker are employed to sense the extern world and produce an estimation of the orientation of the head. This data should be treated to produce a less noisy final orientation and position calculation. To do so, an orientation Kalman fusion filter and a position Kalman filter are tuned. Finally, a pose estimator will receive the orientation and position and. At this point, the render unit is prepared to generate a new AR frame merging the real image captured by the camera and the virtual object stored in an 3D-objects database.

All these elements work together to accomplish the same goal, the creation of an augmented image, but they perform different tasks. These tasks can be divided in three different categories:

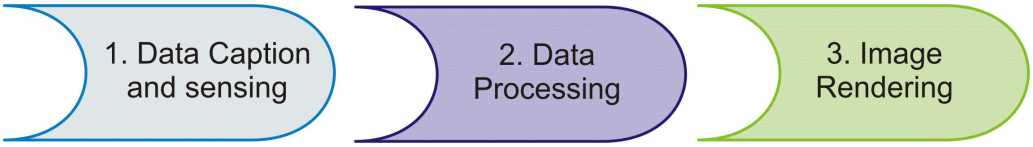

Figure 6.2: *AR processes*

1. The first step is sensing

   The video and the inertial tracker's job is to calculate the best possible estimation of the actual head pose. To achieve this, external data is needed. The raw data sensed by external hardware modules is transformed by these trackers into the head position estimation required to place the virtual image in the right place.

   On the visual part,a camera captures images from the real world. This pixel information will enter a video-based markless algorithm where image data will be extrapolated and head movement will be calculated analyzing changes among the consecutive images. This visual tracker module, calculates both the position and the orientation (pose).

   On the inertial module, a 3-DOF inertial hardware module obtains its motion sensing using vibratory gyros. These gyros sense the angular rates of the cube movements and by the simple integration of them the orientation is calculated. This inertial tracker component just detects the orientation changes and not the position.

2. The next step is filtering and combining the data detected.

   A position Kalman filter is employed to compute the translation obtained by the visual-tracker. Random noise included by using the video algorithm is minimized with this filter.

   The orientation Kalman fusion module receives the orientation data of both of the trackers and combines it in order to get a more accurate estimation. This is the core of our work and a detailed explanation on how it works is given in Chapter 7.

   Finally, a pose estimator joins the position and orientation computed in the two previous modules and calculates the exact point where the virtual object must be placed.

3. Final step: image merging and rendering.

   At this point, we have all the elements needed to produce the augmented image:

- the real image
- the virtual object
- the position of the virtual object in the real world

The final module, the rendering unit, will generate the virtual object in the right position and will fuse it with the real image obtained with the camera. The augmented image created is ready to be shown on a monitor or on a HMD (head mounted display).

6.2 Trackers

As mentioned before, to be able to estimate the head pose, external data must be sensed. The two trackers used in our global system for this task are a video tracker and a commercial inertial tracker. The first one will estimate the frame by frame rotation by means of a video data processing algorithm, and the latter will sense the forces applied to this tracker with internal gyroscopes and accelerometers.

A brief explanation of each of them is exposed in this section, and a further analysis of their weaknesses is done to conclude.

6.2.1 Video Tracker

The video tracker developed for the augmented reality platform is a markeless tracking algorithm with no constraints in the geometry of the scene. Users pose is estimated using natural feature point extraction and tracking, and the epipolar geometry constraint between camera views.

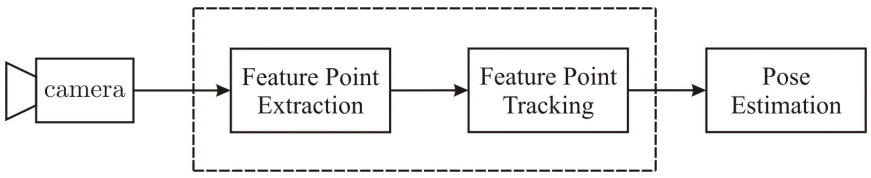

Figure 6.3: *Block diagram of the video tracker*

Two steps are taken while estimating the rotation of the camera:

- FPtracking

- Pose estimation

FPtracking:

First of all there is a Feature Point Extraction block which select remarkable points in the current image and tracks them in the future frames. The procedure is performed on a frame-by-frame basis. For each new frame, several FPs are extracted and all tracks are updated.

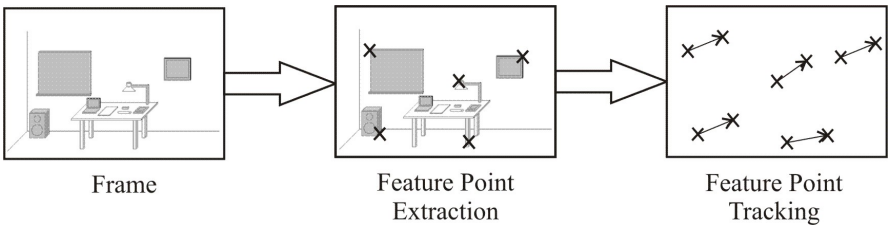

Figure 6.4: *FP tracking*

An intensity-based algorithm is applied for point extraction. Every track is matched again against all the extracted candidates to find its new position. Robust association is fundamental to ensure good quality of tracks. The system's performance relies heavily on FP tracking quality.

Pose estimation:

Later on, pose estimation of the user permits a correct augmentation of the scene. The output of this block is the increment of the head's pose (orientation and position in the space) from the last frame to the present.

6.2.2 Inertial Tracker

The inertial tracker used in the AR framework is a commercial hardware element: $InertiaCube^2$.

As described in the $InertiaCube^2$ manual [24], the InertiaCube2 is an inertial 3-DOF (Degree of Freedom) orientation tracking system. It obtains its motion sensing using a miniature solid-state inertial measurement unit, which senses angular rate of rotation, gravity and earth magnetic field along three perpendicular axes. The angular rates are integrated to obtain the orientation (yaw, pitch, and roll) of the sensor. Gravitometer and compass measurements are used to prevent the accumulation of gyroscopic drift.

Figure 6.5: *InertiaCube*^{2}

The InertiaCube2 is a monolithic part based on micro–electro–mechanical systems (MEMS) technology involving no spinning wheels that might generate noise, inertial forces and mechanical failures. The InertiaCube simultaneously measures 9 physical properties, namely angular rates, linear accelerations, and magnetic field components along all 3 axes. Micro-miniature vibrating elements are employed to measure all the angular rate components and linear accelerations, with integral electronics and solid-state magnetometers. The magnetometers are included for optional yaw drift correction. The geometry and composition of these elements are proprietary, but the functional performance of the multisensor unit can be understood sufficiently by reference to the equivalent diagram in Figure 6.6.

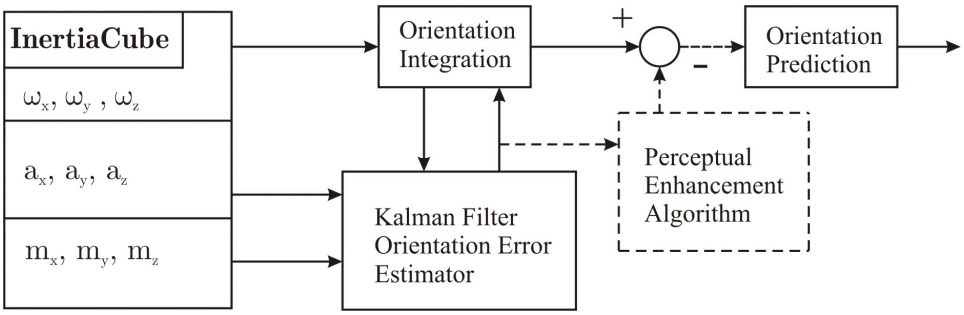

Figure 6.6: *GEOS mode tracking algorithm*

6.2.3 Tracker problems

As described in previous sections (see Chapter 3), none of the existing trackers are completely prepared to generate enough accurate measures in all conditions. Every kind of sensor has fundamental limitations.

Our data fusion system (Chapter 7) compensates for weaknesses in each component.

Visual tracker problems

Vision-based tracking has been extensively employed for AR. Its main advantage over other active and passive sensing technologies is that vision trackers use the video–frames of the camera to estimate its pose, so no extra elements are required, not a separate sensor or emitter attached to the environment. It estimates the camera pose with the same image information as the user's vision.

Nevertheless, vision-based trackers have lots of weaknesses and are not accurate enough to work alone in a demanding AR application. They suffer from a notorious lack of robustness, and a high end-to-end system delay due to the high computational expenses of video algorithms. Furthermore, since vision sensors (cameras) nominally sample at video rates (30Hz), they are not appropriate for abrupt camera rotations.

Strengths of the video-based tracker:

- Low error in static or quasi-static processes.
- Tracking measurements made from the viewing position minimize the visual alignment error.

Weaknesses of the video-based tracker:

- High latency: high computational expense.
- Low rate (maximum 30 Hz).
- Bad response to fast or high frequency movements.
- Lack of robustness under bad light conditions and vulnerable to occlusions.

Inertial tracker problems

Inertial sensors are commonly used for motion tracking ([25],[26]). The main property of this kind of trackers is that they are selfcontained, small and lightweight so it is easy to adapt to a head–mounted display. Moreover, they are high rate trackers with good response to rapid movements.

But the principal problem of this trackers is drift. Pose is calculated by integration of the measurements of the accelerometers and gyroscopes and noise or a small bias in these measurements produce an increasing drift in the pose. To correct the accumulated drift, periodic measurements from other sensors must provide absolute pose data.

Strengths of the inertial tracker:

- Small and lightweight: It can be easily included in a head–mounted display.
- No physical limits on the working volume: No previous preparation in the working area is needed (like in acoustic, magnetic or optical tracker sensors)
- High update rate: Superior to 180Hz.
- Low latency: Inferior to 2ms.
- Relatively low error at high frequencies and velocities
- Very responsive: readily reacting to fast movements.

Weaknesses of the inertial tracker:

- Error accumulation due to integration (numerical): It needs a periodic recalibration, so it does not normally work alone but in hybrid systems where the complementary tracker recalibrate the inertial one.
- Drift in the axis of rotation of a gyroscope due to the remaining friction between the axis of the wheel and the bearings
- High error at lower frequencies and velocities: At low velocities (very slow or no movement) one must in practice contend with pronounced bias and drift error. For slow rotations and small changes, the sensors exhibit a very low signal to noise ratio.

Conclusion

As described in this section, neither the visual tracker nor the inertial produce high quality measures in all conditions. While the visual tracker has a good response in slow motion circumstances it has problems when the speed of the head movements increase.

These mediums exhibit a complementary behavior which our system takes advantage of. Our fusion filter, explained in detail in the next chapter, can dynamically trust more to one or the other tracker depending on each situation; its way of weighting the data of each module is variable, and adaptable to achieve a enough accurate orientation under any condition. So our system is able to combine the low-frequency stability of vision sensors with the high-frequency motion tracking of inertial rate-gyroscope sensors.

Chapter 7

Fusion Filter

As analyzed in the last Chapter, no single tracker is optimal to work alone in the task of tracking the head's pose. The fusion filter implemented has been designed to fuse the orientation data that is produced by our vision tracker and the commercial inertial tracker (see Section 6.1).

In this Chapter, we describe the three possible implementations that has been considered when designing the filter and the reasons why just one of them has been chosen. There is a description of each of the implementations and the analysis of their properties, and a further comparison among them.

Furthermore, there is a general description of our fusion module and a mathematical explanation of the fusion algorithm that combines the tracker's measures. Moreover, you can find a deep analysis on how we have dealt with the problem of the asynchrony. Finally, a large explanation of the properties of the filter is given at the end of this Chapter.

7.1 Level of the sensor fusion

As Thomas Bak stated in [27], observational data may be combined, or fused, at a variety of levels. Three levels are explained: high–level, midlevel and low–level fusion. Basing on this classification, three different fusion methods have been considered to develop. Depending on the level of sensor fusion, the data combination is done at an earlier or later stage of the tracking platform.

This three options have been thought while deciding to fuse inertial and visual trackers data for this project, but it can be extended to any kind and

number of sensors.

High–level fusion: fuses decisions.

One independent Kalman filter is designed for each of the sensor modules. This means, each of the trackers estimates the orientation independently. Finally, a decision–module will choose which the optimal tracked orientation is. As the quality of the estimation of each of the trackers varies depending on external conditions and the velocity of the movements, the decision-module will continuously compute and select the best orientation estimation.

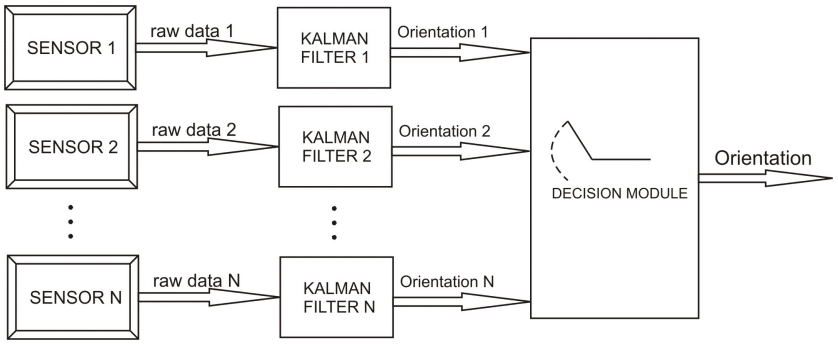

Figure 7.1: *High-level fusion*

The main problem with this kind of data combination is that there will be jumps in the final orientation estimation each time the decision–module determines to change of sensor's data. This problem is produced because each of the fusion filters is independent, and there is no relation between the orientation state–vector of each of the sensors.

Midlevel fusion: fuses parameters concerning features (state vector) sensed locally.

One only Kalman filter is constructed for all of the trackers. In such a configuration, the Kalman filter uses a common process and measurement model. The entry values to the fusion block are the orientation estimated by each of the sensors, and the deviation of this measure (this data indicates the quality of the measure). The Kalman fusion will weight and incorporate the new data to the present orientation estimation according to the deviation value.

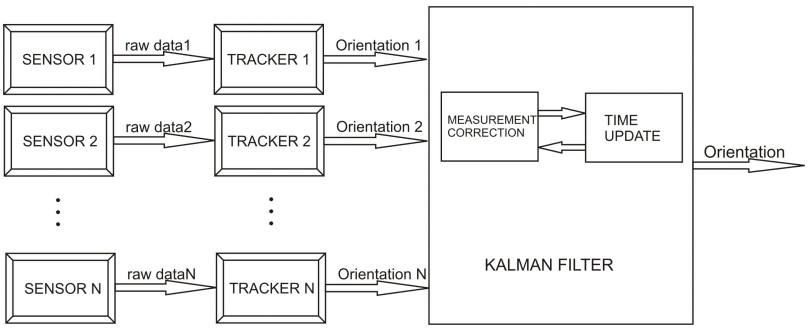

Figure 7.2: *Midlevel fusion*

This solution overcomes the problem that is produced in a high–level fusion implementation. No jumps are produced because there is a common Kalman filter for all the trackers, and consequently there is a unique state vector (orientation estimation).

Low–level fusion: fuses the raw data from sensors

One only Kalman filter is constructed for all of the trackers. In such a design, the Kalman filter uses a common process model, but a distinct measurement model for each of the inertial and optical subsystems. The entry values to the fusion block are the raw data produced by each of the sensors.

The measurement correction is different depending on whether the data is produced by the inertial or the visual sensor unit. The time update, which projects the vector state values and the present orientation to the future, is common for both modules.

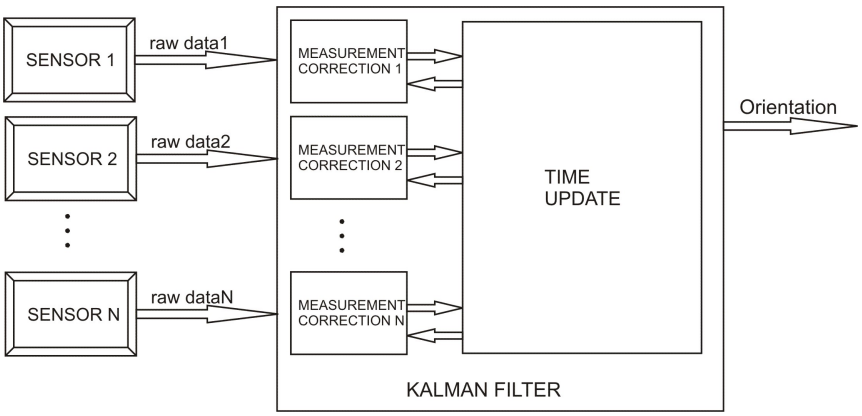

Figure 7.3: *Low-level fusion*

The higher the level of fusion, the smaller the amount of information that must be processed. But also, a higher–level fusion implies that we achieve a less robust combination of data. In a low–level implementation, each of the raw data variables is modelled and its error is quantified. For example, as we can see in [25] and in [22], in its measurement model, gyroscopic angular rates error and accelerometer measurements error are modelled. On the other hand, in a midlevel and high–level combination, no individual raw data error is modelled but a global orientation error.

Although the low–level design is the most common selection, our final chose has been to construct a **midlevel platform**. There are several reasons for choosing this level of combination and not other. First of all, our

commercial inertial–tracker does not present the raw data of the accelerometers and the gyroscopes. This data is internally processed by the tracker and the measure that we get is the final orientation estimated.

Moreover, this design guaranties a greater flexibility and adaptability because both modules of the Kalman filter are common for all the trackers. No individual measurement model is designed for each tracker. This permits us to change one of the tracking modules by another without modifying the structure of the platform. For instance, its possible to replace the inertial tracker by a magnetic or acoustic tracker, without any variation of the fusion configuration (see Section 7.4).

7.2 Description of the fusion module

The basic idea is to use the Kalman filter to weight the different mediums more heavily in the circumstances where they each perform best, thus providing more accurate and stable estimates than a system based on single medium alone.

In this section, first of all we describe the information needed by our fusion system to weight the new measurements in accordance with their quality. A further description of the this module will detail the elements of our fusion block and its principal processes. And finally, the reader can find a mathematical explanation of the fusion algorithm, and the importance of knowing the exact time of the arrival of new data.

7.2.1 Orientation tracker data

Our fusion module will ponder each new measurement according to its quality. To do so, in addition to the measured orientation, our system needs to know the noise variance of the measurement (the quality of this new data) and the instant of time when it has been produced.

This means that the orientation tracker data produced by each of the tracker modules is not going to be just the euler measurement, but also the time in *seconds* and the measurement noise variance in $degrees^2/second$.

$$\text{Orientation Tracker Data} \begin{cases} \text{Orientation measure: } (\psi, \theta, \phi)_i \\ \text{Time of measurement: } t_i \\ \text{Measurement noise variance: } r_i \end{cases}$$

The fusion filter has been designed to work with two possible orientation methods: euler angles ($(\psi, \theta, \phi)_i$) and quaternions (see Section 2). Due to some problems with quaternion measurements of our commercial inertial tracker (see Section 8.1.2), our final decision has been to use the euler angle method.

Moreover, the time when each measure is produced is needed to update in time our estimation in the right way. Time lapse between consecutive measures plays an important roll in the fusion filter algorithm.

And finally, our algorithm should receive a parameter indicating the quality of each new data (r measurement error noise). This value is required to weight the new orientation measurement in the estimation according to its quality.

7.2.2 Fusion module elements

Our fusion module is continuously waiting for new tracker data to arrive to process it. This new measurements are going to be stored and computed. So two elements are included in the fusion module:

Fusion Module elements $\begin{cases} \text{A pool: orientation data storage;} \\ \text{A Kalman fusion filter: orientation data processing.} \end{cases}$

The first component, the data pool, is an element intended for receiving orientation data from any of the trackers and storing it (in order of arrival) until the Kalman fusion filter is prepared to process it.

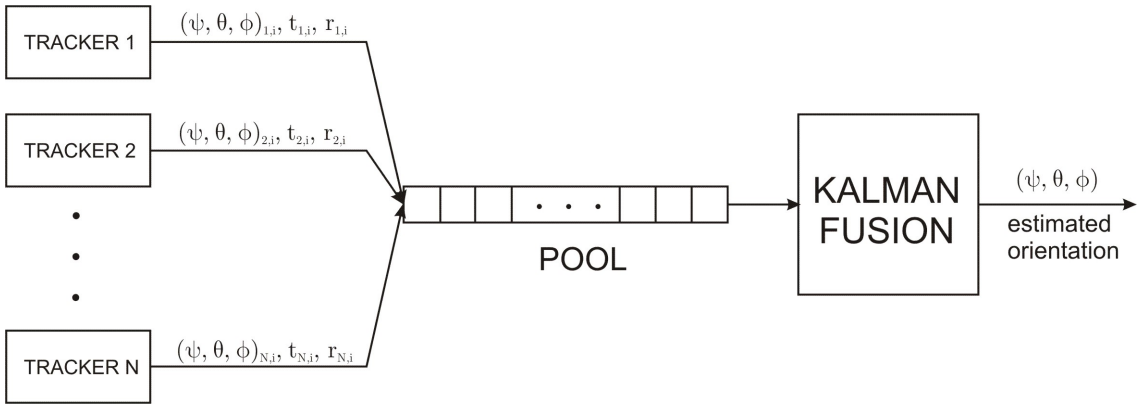

Figure 7.4: *Fusion Module structure.*
On the left, tracker input data to our module. On the right, module elements: pool and Kalman filter.

So, each of the trackers is going to produce asynchronously new measurements of the head's orientation and our Kalman fusion filter is going to get each new data and process it to update the estimation of the current orientation.

If the Kalman fusion filter is still processing the last data when a new incoming one arrives, this new measurement will be stored in the pool. A high input measurement rate can cause an accumulation of more than one measurement in the pool.

We have considered two different ways to solve this problem, the system designed can work in two modes:

Ignore old and process the latest measurement: When there is more than one element in the queue, our first solution consists in emptying old data piled in the pool and then process the most recent measurement. By employing this alternative, our system will erase and not process some tracker data. But our system can deal with loses of data (see the properties of our fusion block, Section 7.4), and this solution will reduce the final latency.

Process all the measurements: Our second solution is to process all the measurements piled in the pool. More measurements data will be processed so a more accurate estimation will be done. However, this alternative implies a greater latency because the system will have to compute all accumulated information before processing the last incoming data.

The first alternative should be employed if the output rate of the trackers is so elevated that our processor is not able compute all the measurements in real time. Moreover, this solution should be also chosen if latency is a critical factor in our application. The latter alternative produces a more accurate estimation but will increase the end-to-end system latency if the measurement data input is too high.

7.2.3 Fusion module Flow Chart

Our Kalman fusion filter will get data from the pool and process it to update the estimation. Its flow chart is presented in Figure 7.4.

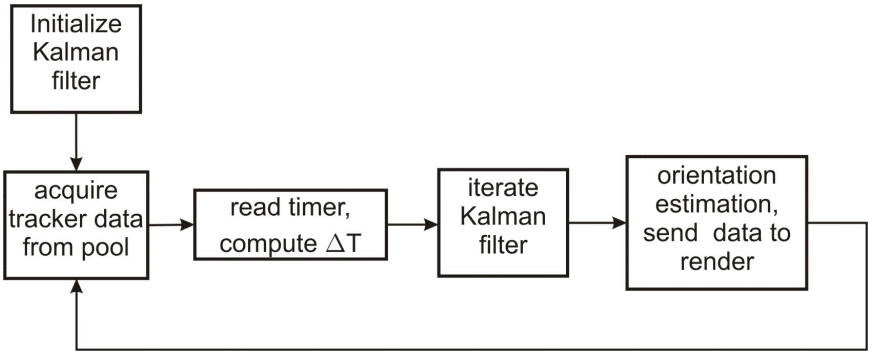

Figure 7.5: *Fusion Module process*

The initialization block is executed once at program start–up and sets the initial estate estimates and covariances as described in the fusion algorithm section (Section 7.3). Then the program is going to wait for the new measurements arrival of any of the trackers. Each time a new measurement arrives to our system, our fusion module acquires this tracker data and calculates ΔT, that is the lapse of time between this new measurement and the last computed one.

Later on, the Kalman filter is iterated in the following order:

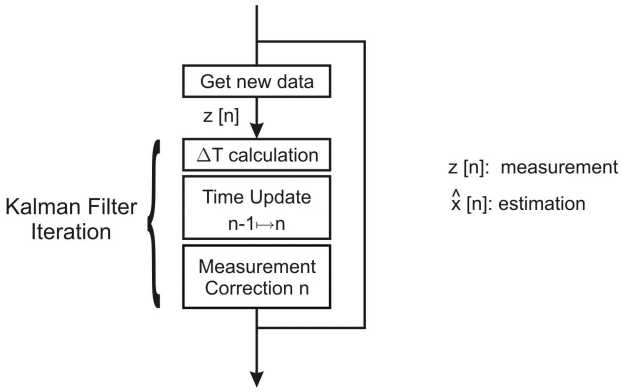

Figure 7.6: *Kalman filter iteration*

Justification of this processing order is given in the next section. And the parameters of the time–update and measurement–correction equations are

described in Section 7.3.

Finally, the estimated orientation is sent to the render block which produce the augmented scene.

7.2.4 Dealing with the asynchrony

As it has been previously described in the Kalman filter theory (Chapter 4), the state-prediction block needs to know the time between two consecutive measurements to process the time update. This value is needed to update to the current time the state vector correctly.

In other platforms with just one tracker with constant data rate, the period between two consecutive incoming measures ΔT is known. However, as our system works with two trackers with different data rates there exists the problem of asynchrony: it is not possible to know exactly when this new data is going to arrive, and the value of ΔT is not constant and difficult to predict.

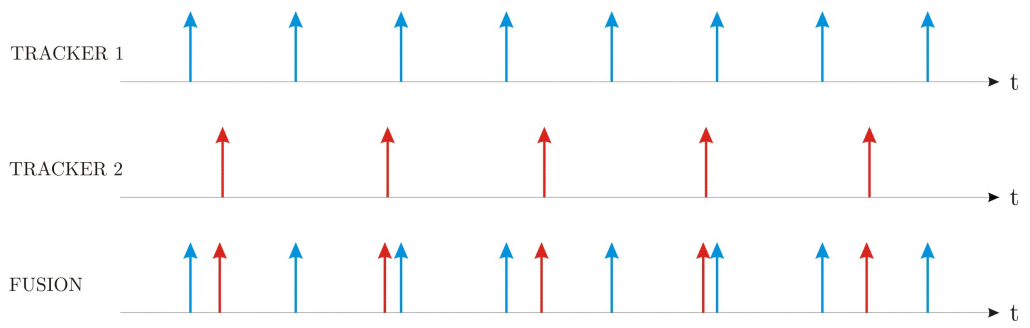

Figure 7.7: *The problem of asynchrony, variable $\triangle T$*
Each arrow represents a measurement arrival.

Systems with constant data rates normally work in the following way (see Figure 7.9); when a new data arrives to the system the filter will do a measurement correction, and then will project the state vector to the future. This is possible due to the fact that the instant of time when the future data will arrive is known.

But as explained before, we do not know when the future measurement is going to be produced.

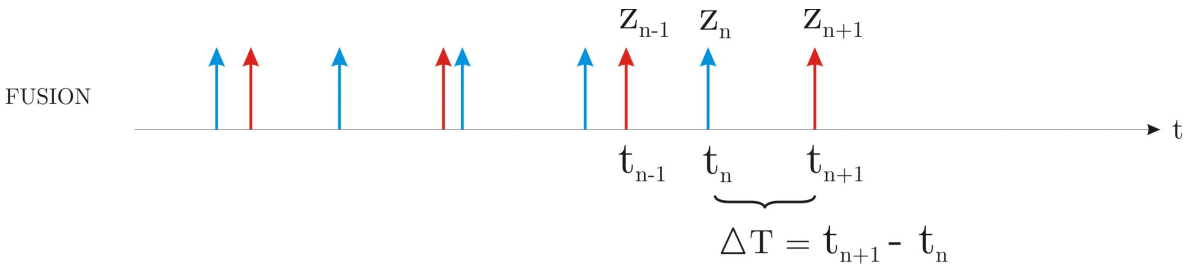

Figure 7.8: *Variable $\triangle T$*

What is done in this cases is not to project the state vector to the future as it is difficult to know when a new data is going to be produced. When a new incoming measurement arrives, our fusion filter is going to update the state–vector to present time and then it will do the measurement correction:

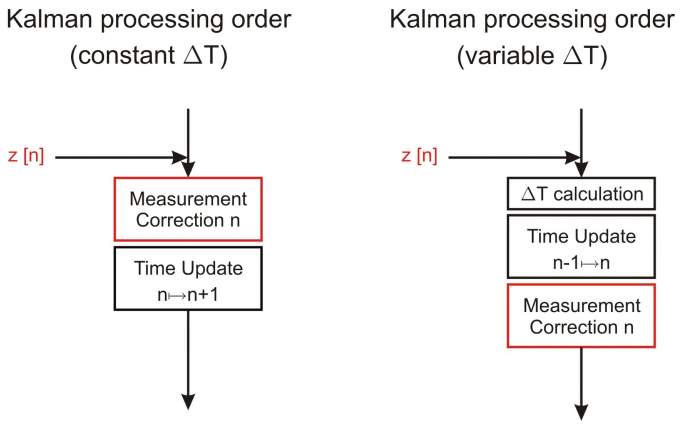

Figure 7.9: *Kalman processing order*
Our system works with variable ΔT (processing order on the right).

We should remark that at time t_n the system will have to compute the time-update before doing the measurement correction. With this solution, the lapse of time between the measurement arrival and the orientation estimation is larger comparing with a system with constant data rate. This extra time is produced by the processing time to compute the time update of the measure.

7.3 Fusion algorithm

The basis of our fusion algorithm is a unique Discrete Kalman Filter which has a common process and measurement model for both of the inertial and video trackers.

As explained in the Kalman filter theory (Chapter 4), when developing a Kalman Filter the first thing to do is to formulate these two models.

7.3.1 Process model

First of all, we will define the state vector which is the vector that summarizes the past of our idealized deterministic dynamic system. Our state vector contains six elements, the three elements (yaw, pitch and roll) of the orientation value to estimate, and their derivatives:

$$x_k = [\psi_k, \theta_k, \phi_k, \dot{\psi}_k, \dot{\theta}_k, \dot{\phi}_k]$$

Later on, once the state–vector is determined, we should fix the state transition matrix A. This matrix describes how the state x is expected to change from one timestep to the next.

$$x_{k+1} = Ax_k + w_k$$

We consider that the dynamic system of the head orientation $\Phi = [\psi_k, \theta_k, \phi_k]$ follows the next equation:

$$\Phi_{k+1} = \Phi_k + \dot{\Phi}_k \cdot \Delta T$$

And as a simplification we assume that during this period of time, the head's rotation velocity is constant:

$$\dot{\psi}_{k+1} = \dot{\psi}_k,\ \dot{\theta}_{k+1} = \dot{\theta}_k,\ \dot{\phi}_{k+1} = \dot{\theta}_k$$

This supposition is possible because the timestep between consecutive measures $\Delta T = t_{i+1} - t_i$ is very small, below the inverse of the rate of the fastest tracker. In our system, the fastest tracker is the inertial tracker

($InertiaCube^2$ nominal rate= $180Hz$ which means a timestep $\Delta T < 1/180 = 5.56 \cdot 10^{-3}$)

Now, the process model can be written in the following way:

$$\underbrace{\begin{bmatrix} \psi_{k+1} \\ \theta_{k+1} \\ \phi_{k+1} \\ \dot{\psi}_{k+1} \\ \dot{\theta}_{k+1} \\ \dot{\phi}_{k+1} \end{bmatrix}}_{x_{k+1}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta T_k & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta T_k & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta T_k \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \psi_k \\ \theta_k \\ \phi_k \\ \dot{\psi}_k \\ \dot{\theta}_k \\ \dot{\phi}_k \end{bmatrix}}_{x_k} + \underbrace{\begin{bmatrix} \Delta T_k^2/2\ddot{\psi}_k \\ \Delta T_k^2/2\ddot{\theta}_k \\ \Delta T_k^2/2\ddot{\phi}_k \\ \Delta T_k\ddot{\psi}_k \\ \Delta T_k\ddot{\theta}_k \\ \Delta T_k\ddot{\phi}_k \end{bmatrix}}_{w_k}$$

The error in the process model is a result of considering the head acceleration null during the period between two measures. The measurement covariance error $Q = E[w_k w_k^T]$ is:

$$Q = E\left[\begin{bmatrix} \Delta T_k^2/2\ddot{\psi}_k \\ \Delta T_k^2/2\ddot{\theta}_k \\ \Delta T_k^2/2\ddot{\phi}_k \\ \Delta T_k\ddot{\psi}_k \\ \Delta T_k\ddot{\theta}_k \\ \Delta T_k\ddot{\phi}_k \end{bmatrix} [\Delta T_k^2/2\ddot{\psi}_k \Delta T_k^2/2\ddot{\theta}_k \Delta T_k^2/2\ddot{\phi}_k \Delta T_k\ddot{\psi}_k \Delta T_k\ddot{\theta}_k \Delta T_k\ddot{\phi}_k]\right]$$

Assuming that the three angle's rotation acceleration is independent (cross correlations are null) and have the same variance,

$$E[\ddot{\psi}_k\ddot{\theta}_k] = E[\ddot{\psi}_k\ddot{\phi}_k] = E[\ddot{\phi}_k\ddot{\theta}_k] = 0$$

$$E[\ddot{\psi}_k\ddot{\psi}_k] = \sigma^2_{\ddot{\psi}_k}, E[\ddot{\theta}_k\ddot{\theta}_k] = \sigma^2_{\ddot{\theta}_k}, E[\ddot{\phi}_k\ddot{\phi}_k] = \sigma^2_{\ddot{\phi}_k}$$

$$\sigma^2_{\ddot{\psi}_k} = \sigma^2_{\ddot{\theta}_k} = \sigma^2_{\ddot{\phi}_k} = q$$

We can rewrite the Q matrix as follows:

$$Q = q \begin{bmatrix} (\Delta T_k^4/4) & 0 & 0 & (\Delta T_k^3/2) & 0 & 0 \\ 0 & (\Delta T_k^4/4) & 0 & 0 & (\Delta T_k^3/2) & 0 \\ 0 & 0 & (\Delta T_k^4/4) & 0 & 0 & (\Delta T_k^3/2) \\ (\Delta T_k^3/2) & 0 & 0 & (\Delta T_k^2) & 0 & 0 \\ 0 & (\Delta T_k^3/2) & 0 & 0 & (\Delta T_k^2) & 0 \\ 0 & 0 & (\Delta T_k^3/2) & 0 & 0 & (\Delta T_k^2) \end{bmatrix} \tag{7.1}$$

$$q = \sigma^2$$

Where σ is the standard deviation of the head acceleration.

But, later practical experiments (Section 8.2) have demonstrated that the assumption that all the angle's rotation acceleration are equal is wrong (movements of the head from right to left is normally sharper than movements from up to down, or inclinations of the head to the right and left shoulder). So each of the angle acceleration variances are going to be independent and are going to have different values.

$$\sigma^2_{\ddot{\psi}_k} \neq \sigma^2_{\ddot{\theta}_k} \neq \sigma^2_{\ddot{\phi}_k}$$

7.3.2 Measurement model

Once the process model has been designed, we should also concrete the measurement model. This model describes how the estimation space maps into the observation space, that is, it relates the measurement vector and the state vector by a linear equation:

$$z_k = Hx_k + v_k$$

In our design, the input measurements to the fusion filter is going to be the orientation measured by each of the trackers and not the raw data of the sensors (see Levels of sensor fusion, Section 7.1):

$$z_k = [\tilde{\psi}_k, \tilde{\theta}_k, \tilde{\phi}_k]$$

So relating the state vector with the measurement vector is done using an identity matrix:

$$\underbrace{\begin{bmatrix} \tilde{\psi}_k \\ \tilde{\theta}_k \\ \tilde{\phi}_k \end{bmatrix}}_{z_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} \psi_k \\ \theta_k \\ \phi_k \\ \dot{\psi}_k \\ \dot{\theta}_k \\ \dot{\phi}_k \end{bmatrix}}_{x_k} + \underbrace{\begin{bmatrix} v_{\psi_k} \\ v_{\theta_k} \\ v_{\phi_k} \end{bmatrix}}_{v_k}$$

where v_k is the measurement error. And the measurement error covariance,

$$R = E[v_k v_k^T] = E\left[\begin{bmatrix} v_{\psi_k} \\ v_{\theta_k} \\ v_{\phi_k} \end{bmatrix} [v_{\psi_k} v_{\theta_k} v_{\phi_k}]\right]$$

as v_k is assumed to be independent (of each other), white, and with normal probability distributions,

$$E[v_{\psi_k} v_{\theta_k}] = E[v_{\psi_k} v_{\phi_k}] = E[v_{\theta_k} v_{\phi_k}] = 0$$

$$E[v_{\psi_k} v_{\psi_k}] = E[v_{\theta_k} v_{\theta_k}] = E[v_{\phi_k} v_{\phi_k}] = r$$

we can rewrite the R matrix as follows:

$$R = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} \tag{7.2}$$

where r is the noise measurement energy of each orientation parameter (yaw, pitch and roll).

7.3.3 Filter Tuning: The Q_k and R_k Matrices

All the matrices needed to compute a Kalman filter have been fixed by simple theoretic inspection of the process and the measurement system. But, as

explained in the chapter of Kalman filter theory (Section 4.5), tuning is used to achieve a better result of filter's performance. Tuning consists in changing the variables Q and R to improve our filter execution and reduce the noise in measurements as much as possible.

In our algorithm design, neither the parameter q of the process covariance matrix Q (equation 8.1), nor the parameter r of the measurement noise covariance matrix R (equation 7.2) have been theoretically determined with a constant value.

When implementing a Kalman filter, a critical part of its design is fixing the elements of the matrices Q and R. Tuning has been employed for this task (see Section 8.2) that consists in changing the variables Q and R to get the best possible performance.

7.4 Properties of the fusion platform

The fusion framework developed has been designed to work under the requirements imposed by the tracker–modules and the posterior video–render. Since the vision and gyro sensors have different sample rates, we implement a complementary motion estimate filter. The benefits of this kind of filtering are:

1. Compensation of individual tracker weaknesses

   As described in previous sections, none of the existing trackers are completely prepared to generate enough accurate measures in all conditions. Every kind of sensor has fundamental limitations. This kind of data fusion system compensates for weaknesses in each component.

   Visual and inertial trackers exhibit a complementary behavior which our system takes advantage of. Our platform can dynamically trust more one or the other tracker depending on each situation; its way of weighting the data of each module is variable, and adaptable to achieve enough accurate orientation under any condition.

2. Admission of asynchrony and multiple data-rates

   The structure also admits asynchrony and accommodates the different sample rates of the sensors. No matter in which instant of time the sensor data has been produced, the fusion system will process it immediately and update the estimation of the actual pose.

   This immediate incorporation of the new tracker data reduces the end-to-end system latency. The algorithm does not need to get the data of all the modules before updating the orientation value. This means, each time a new value is read an improved estimation is computed.

3. Adaptable to any kind and number of trackers

   In future works it would be desirable to incorporate other kinds of trackers to substitute the original ones or to aid in the tracking to the basic ones. This structure is prepared to deal with any number and kind of trackers.

   The only condition required is to know the error variance of the measurements of each of the trackers.

4. Handle of incomplete/incoherent sensor information

Our data processing system handles incomplete information measurements. Lack of illumination or occlusion of landmarks are usual problems in visual processing systems. In these cases, the accuracy of the measures gets worse and untrusting results are produced.

In those situations our system supports the deactivation of those trackers that produce incoherent or undesirable data and the posterior reactivation when the problem has disappeared. The system will work in those periods with a unique tracker module.

Part III

Experiments and results

Chapter 8

Experiments and results

There have been two different kinds of experiments performed during our project. The first ones, have been done during the creation process of the fusion module and their goal has been to develop an optimal Kalman fusion filter design. The second kind have been completed in the last phase of the project, and they have been used for the fusion filter tuning and evaluation.

8.1 Design Experiments

When designing our fusion module we have considered some different options to formulate the Kalman equations of the fusion filter.

We have studied how other work groups have dealt with the problem of orientation formulating and we have observed that the Kalman state–vector is commonly expressed in two different ways: as a quaternion [28] and as euler angles [29]. To decide which one is better, we programmed one Kalman filter that works with quaternions and another that works with euler angles. But we have encountered some problems in both solutions because of the output data of our commercial inertial tracker ($InertiaCube^2$).

8.1.1 Euler Problem

In Figure 8.1 we can see the time evolution of user's head orientation. Specifically, the yaw rotation which represents turns of the head from left to right sides.

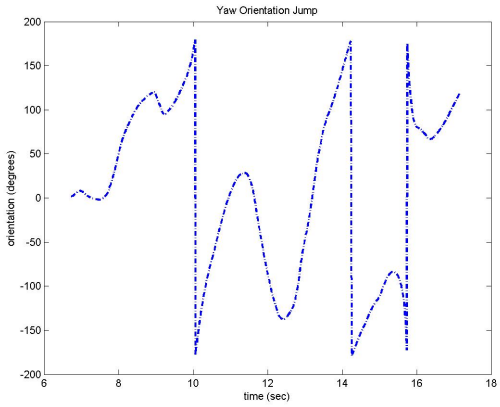

Figure 8.1: *Euler orientation jumps (yaw)*

These rotation measurements have been produced by the same commercial inertial tracker that we use in the tracking fusion, the $InertiaCube^2$ (Section 6.2.2). The reader should observe that when the head orientation exceeds +180 degrees a jump is produced, and the next orientation value is -180 degrees (in Figure 8.1, this phenomenon can be observed at 10 sec. and 14.2 sec.). This jump is also produced in the other sense of rotation (15.8 sec.).

These discontinuities in the Euclidean space produce a bad response of our filter. When we use a Kalman filter to process orientation data and this kind of jump is produced, the Kalman filter will perform an erroneous estimation (see Figure 8.2).

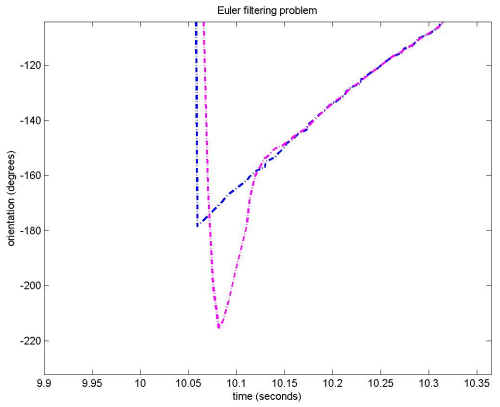

Figure 8.2: *Jump filtering problem*

The source of the problem is that the Kalman filter uses a linear process model to predict the next value. When this abrupt change in orientation is produced, our linear estimation will predict following orientation values as a continuation of this jump. Latter, after some number of new measurements, the estimation is corrected. In Figure 8.2 the reader can observe this anomalous behavior.

8.1.2 Quaternion Problem

Figure 8.3 shows the time evolution of one of the parameters of the user's head orientation using quaternions. This parameter is the element Y of the quaternion (see quaternion theory in Section 2.2).

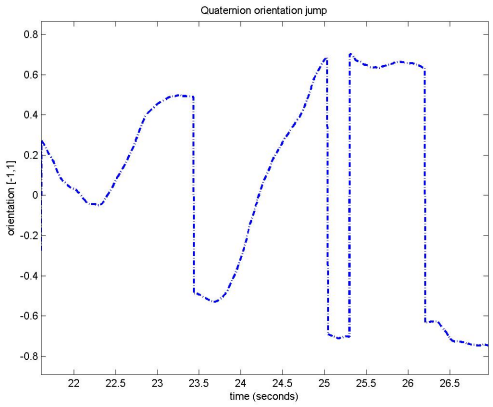

Figure 8.3: *Quaternion orientation jumps (Y component)*

There is also a problem with the quaternion output measurements of our inertial tracker ($InertiaCube^2$). We can observe that from time to time, there is an unpredictable jump of quaternion parameters (jumps are produced in the graphic at time: $23.3, 25, 25.3$ and 26.2 seconds).

There is a signal inversion of the four quaternion components (W, X, Y, Z) of the inertial tracker output at the same time. Mathematically, a quaternion and its inverse represents the same orientation ($q = -q$). So no wrong measurements are given by the tracker device (the orientation is sometimes represented by an inverted quaternion). But this rapid changes alters the functioning of our Kalman filter.

When trying to filter these orientation jumps, the same error (Figure 8.2) analyzed in the previous section is produced.

8.1.3 Final decision

This kind of jumps make our Kalman fusion filter useless because after their appearance, erroneous orientation estimations are computed. The main difference between quaternion and euler jumps is that quaternion's discontinuities are unpredictable and euler's are not. Euler jumps just appear when an euler angle overpasses +180 or -180 degrees, so they can be predicted and corrected. To do so, it exists the phase unwrapping technique. This function unwraps angular time series such that they are continuous.

However, we do not apply any of these technique but rather assume that head orientation is always smaller than ± 180 degrees in angle.

8.2 Tuning and Evaluation Experiments

Once we developed the fusion framework, we have done some experiments to test the functioning of our filter. The goals of these experiments have been tuning our fusion filter for a better performance, and moreover evaluating its properties.

8.2.1 Evaluation experiments with synthetic data

In order to evaluate the performance of our particular fusion particle filter, we first implemented it with purely synthetic data (Figure 8.4). So, a synthetic orientation of user's head is described, and synthetic inertial and video tracker data are generated from it. A synthetic white Gaussian random sequence is added to the measurements with variance 0.5 for the inertial tracker and 1.5 for the vision tracker (Figure 8.5). In addition, the process and measurement noises, w and v respectively, are assumed gaussian. The data rate of the sensors is $\Delta T = 1/180Hz$ for the inertial tracker and $\Delta T = 1/30Hz$ for the visual tracker.

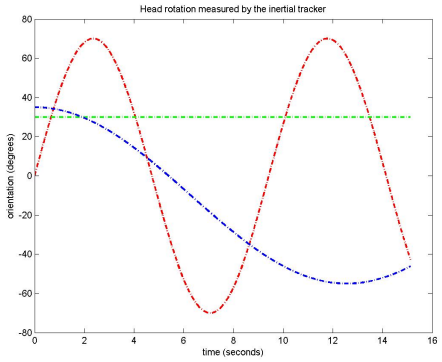

Figure 8.4: *Synthetic head rotation (yaw in red, pitch in green and roll in blue)*

In Figure 8.4, yaw is represented in red, pitch in green and roll in blue. And in Figure 8.5, blue stars and red points, represents synthetic video and inertial tracker measurements respectively.

Finally, a mean square error (MSE) is calculated to evaluate the filter quality. The MSE is given by:

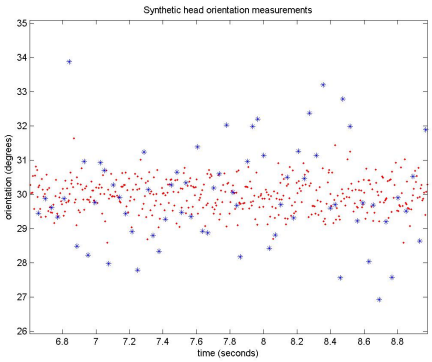

Figure 8.5: *Synthetic Inertial and Video measurements (red points and blue stars, respectively)*

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \|x(i) - \hat{x}(i)\|^2$$

where $x(i)$ is the real value of the orientation, and $\hat{x}(i)$ is the estimation done from the noisy measurements.

We tuned our Kalman filter adjusting Q (process covariance matrix) and R (measurement error matrix) to get the best possible MSE (see Sections 4.5 and 7.3.3). The parameter r of R matrix (in Equation 7.2) is fixed with the real values of the synthetic tracker's measurement error.

$r = 0.5$ when an inertial measurement is filtered
$r = 1.5$ when a video measurement is filtered.

The measurement filtering has been tested with several q values from 100 to 500000 (q is an undetermined parameter of the Q matrix, see Equation 8.1).

As the reader can observe in Table 8.1, our orientation **estimation reduce the measurement error considerably** (see Figure 8.6). For instance, setting a correct q value, **error is reduced by 20** in yaw estimation. In Table 8.1, one can see that if $q = 50000$ we get an MSE of just 0.027 $degrees^2/sec$ from measurements with noise variance of 0.54 $degrees^2/sec$ (the measurement error variance is divided by 20 after the filtering). We achieve even better results filtering roll and pitch measurements because they represent more static movements which fit better with our process model (Section 7.3.1).

Estimation MSE(q)			
q	**Yaw**	**Pitch**	**Roll**
100	2.0494	**0.0043**	0.0151
500	0.3977	0.0063	**0.0104**
1000	0.1977	0.0075	0.0107
5000	0.0469	0.0115	0.0139
10000	0.0316	0.0140	0.0161
50000	**0.0270**	0.0221	0.0235
100000	0.0305	0.0267	0.0279
500000	0.0441*1e+4	0.0399*1e+4	0.0419*1e+4
Measurement error variance	***0.5400***	***0.5188***	***0.5376***

Table 8.1: *Estimation MSE using different q values*

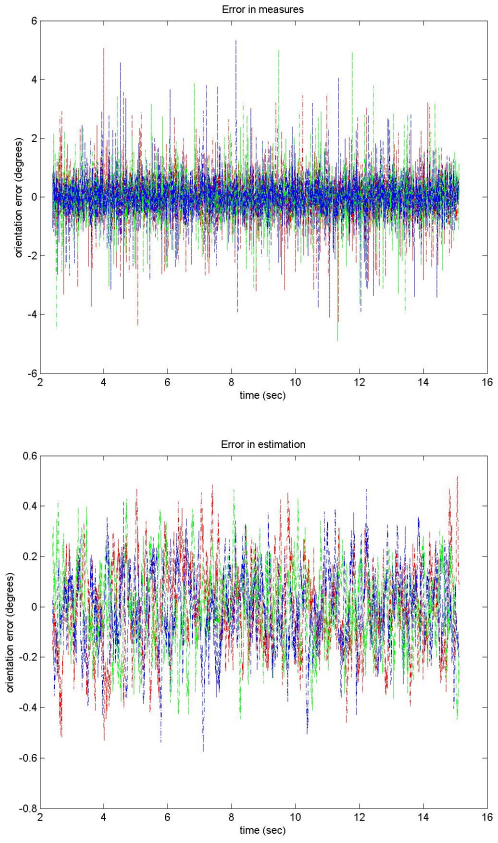

Figure 8.6: *Measurement noise (up) and estimation error (down) (orientation yaw filtering using $q = 50000$)*
Measurement noise is significantly reduced with the Kalman fusion filter (compare dimensions of measurement and estimation errors).

When the parameters are well fixed, we note that the implemented filter succeeds to follow the true trajectories very closely (Figure 8.7), which implies a robust head motion prediction.

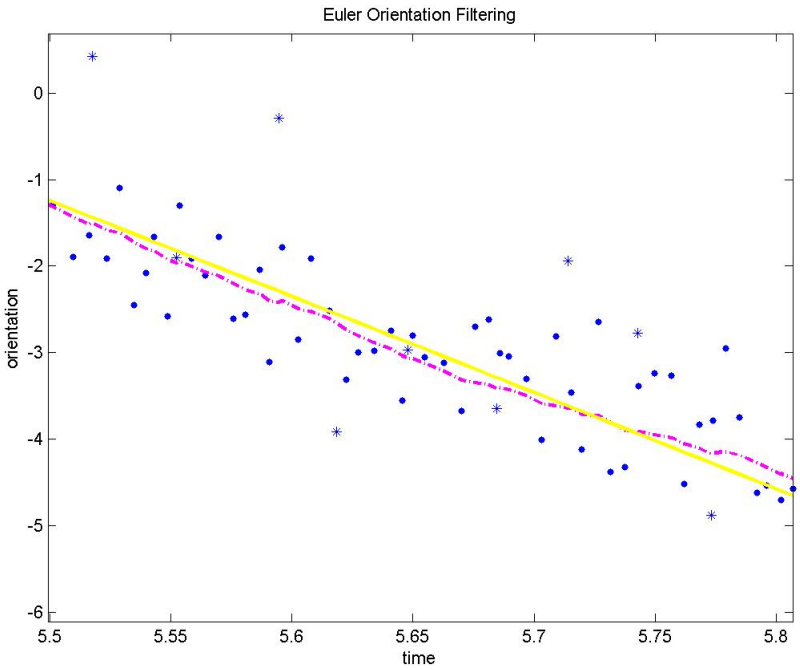

Figure 8.7: *Data fusion and estimation of the head orientation (process to estimate in yellow, estimation in magenta, and inertial and video measurements represented by blue points and stars, respectively).* respectively).

We should remark that for this synthetic head movement, we have to use a greater value of q to filter yaw measurements than for roll or pitch. The reason is that yaw synthetic movement is faster (see Figure 8.4) and more difficult to predict.

8.2.2 Evaluation experiments with real data

Once the fusion filter was tested with synthetic data, we evaluated its functioning using inertial tracker real measurements as entry parameter and the filtering was done in real time. In Figure 8.8 is represented the euler orientation data of a typical head movement.

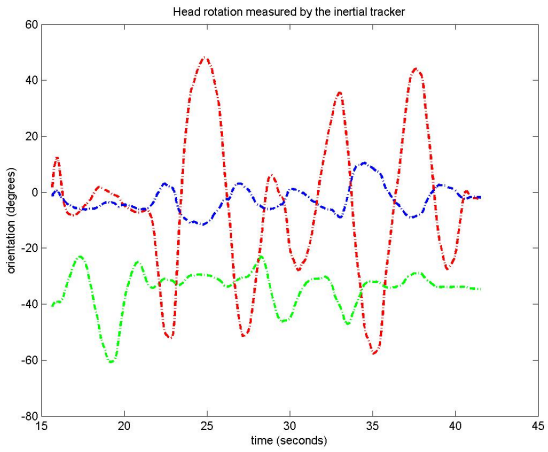

Figure 8.8: *Head orientation measured by the inertial tracker (yaw in red, pitch in green and roll in blue)*

We have done some tests to design a better Kalman process model and to fix the process covariance matrix (Q) with real data. These data was obtained from measurements of a real process and not of a synthetic one as we did before. These tests consisted in tuning our filter using MSE as performance factor for different head movements.

After some filtering performances of real head movements, we realized that one of the assumptions we made in our theoretic study of the process model was false. The assumption that all the angle's rotation acceleration variance are equal is wrong (see Equation 8.1). Movements of the head from right to left (yaw) are normally sharper than movements from up to down (pitch), or inclinations of the head to the right and left shoulder (roll) (see Figure 8.8).

We can observe in this Table 8.2 that the best performance of our filter is achieved with a different q value for each euler angle. While the minimum estimation MSE is measured when $q = 25000$ in pitch and roll, this minimum is produced in yaw when $q = 75000$. The explanation of this result is that,

Estimation MSE(q)			
q	**Yaw**	**Pitch**	**Roll**
100	33.1226	1.8276	1.5809
500	8.2672	0.5951	0.5478
1000	4.5573	0.3781	0.3596
5000	0.9706	0.1613	0.1497
10000	0.0316	0.1578	0.1439
25000	0.3580	**0.1496**	**0.1278**
50000	0.2633	0.1521	0.1320
75000	**0.2331**	0.1580	0.2062
100000	0.2447	0.2249	0.0279
500000	0.5717*1e+4	0.5220*1e+4	1.5919*1e+4
Measurement error variance	***1.0302***	***0.9999***	***1.0075***

Table 8.2: *Estimation MSE using different q values for a real head movement*

as mentioned before, yaw represents faster rotations than pitch and roll. The faster the movement is, the greater the matrix Q should be because this parameter fixes the size of the prediction steps in the estimation (see tuning theory in Section 4.5). So a greater Q, means that the filter adjusts the parameters faster to track changes (but it is also more vulnerable to measurement errors).

Our wrong assumption was to consider that angle's rotation acceleration variance are equal in a head movement:

$$\sigma^2_{\ddot{\psi}_k} = \sigma^2_{\ddot{\theta}_k} = \sigma^2_{\ddot{\phi}_k} = q$$

Our process covariances matrix was initially:

$$Q = q \begin{bmatrix} (\Delta T_k^4/4) & 0 & 0 & (\Delta T_k^3/2) & 0 & 0 \\ 0 & (\Delta T_k^4/4) & 0 & 0 & (\Delta T_k^3/2) & 0 \\ 0 & 0 & (\Delta T_k^4/4) & 0 & 0 & (\Delta T_k^3/2) \\ (\Delta T_k^3/2) & 0 & 0 & (\Delta T_k^2) & 0 & 0 \\ 0 & (\Delta T_k^3/2) & 0 & 0 & (\Delta T_k^2) & 0 \\ 0 & 0 & (\Delta T_k^3/2) & 0 & 0 & (\Delta T_k^2) \end{bmatrix} \tag{8.1}$$

and has to be reformulated in the next way:

$$Q = \begin{bmatrix} (\Delta T_k^4/4)\sigma^2_{\dddot{\psi}_k} & 0 & 0 & (\Delta T_k^3/2)\sigma^2_{\dddot{\psi}_k} & 0 & 0 \\ 0 & (\Delta T_k^4/4)\sigma^2_{\dddot{\theta}_k} & 0 & 0 & (\Delta T_k^3/2)\sigma^2_{\dddot{\theta}_k} & 0 \\ 0 & 0 & (\Delta T_k^4/4)\sigma^2_{\dddot{\phi}_k} & 0 & 0 & (\Delta T_k^3/2)\sigma^2_{\dddot{\phi}_k} \\ (\Delta T_k^3/2)\sigma^2_{\dddot{\psi}_k} & 0 & 0 & (\Delta T_k^2)\sigma^2_{\dddot{\psi}_k} & 0 & 0 \\ 0 & (\Delta T_k^3/2)\sigma^2_{\dddot{\theta}_k} & 0 & 0 & (\Delta T_k^2)\sigma^2_{\dddot{\theta}_k} & 0 \\ 0 & 0 & (\Delta T_k^3/2)\sigma^2_{\dddot{\phi}_k} & 0 & 0 & (\Delta T_k^2)\sigma^2_{\dddot{\phi}_k} \end{bmatrix}$$

Different values have to be fixed for each angle acceleration variance to achieve the best estimation result.

Part IV

Conclusions and future work

Chapter 9

Conclusion and Future Work

The design and implementation of our hybrid tracking system has offered us the opportunity to study the present situation of tracking systems in AR and a view of the different existing data fusion techniques. First, it gave us an interesting insight into the Kalman filtering theory. This theory brought us to the final implementation of our fusion framework. Finally, we built a Kalman fusion filter capable of combining tracker's orientation data, and we evaluated it.

This chapter exposes a summary of the achieved results. To conclude this work, we propose extensions and improvements of the theoretical design and performance of our fusion framework.

9.1 Conclusions

In this work, we designed and implemented a Kalman fusion filter framework to combine orientation measurements of different tracker devices. It has been constructed to work with a markeless video tracker subsystem and an inertial tracker, but its flexibility permits the fusion of orientation data of any kind and number of trackers, no matter their rate or they physical configuration. The fusion is based on a Kalman filter with a common process and measurement model for all the trackers. Our system weights each new measurement according to its quality (measurement noise covariance) and this measurement is incorporated to the orientation estimation in real time.

The main limitation of this fusion filter is that measurement noise of

tracker devices must be gaussian and zero mean. Moreover our filter can not correct drift, so at least one of the tracker devices of the hybrid system must produce global orientation measurements without bias.

9.2 Possible extensions and improvements

This section shows how our fusion module could be extended and improved. Two kinds of improvements have been considered, improvements of our fusion filter and improvements of our global AR tracking platform.

From the point of view of filter design, we consider some possibilities of extension. First of all, our Kalman fusion filter could obtain superior filtering results if we design a more complex process model. We could implement a new model that approaches head's user rotations better. This can be achieved including angle's acceleration in our state–vector (see process model in Section 7.3.1) or using quaternions and an extended Kalman filter (EKF). The slight drawback of this filter improvement is that, more complex models mean longer time processing operations and consequently, the final latency of the system will increase.

In addition, our fusion module bases the weighting of new input measurements on its measurement noise covariance R. Measurement noise not only depends on which tracker produces this data but also it depends on other variables that condition the quality measurements of the tracker. For instance, confidence in visual tracker measurements varies according to light conditions or speed of the camera movements. A method to dynamically evaluate and estimate the measurement noise variance of each tracker should be found.

Finally, we could modify our filter to be able to dynamically change the process covariance Q according to the head's movement. Our filter should reduce the magnitude of Q if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. This solution will improve our filtering performance but it needs a model of the user's intention that is not easy to formulate.

When thinking of the global AR tracking platform, diverse means to improve have also been considered. Deepening on system's performance, we have realized that neither the commercial inertial tracker device ($InertiaCube^2$) nor the markerless video–based tracking subsystem measures the global ori-

entation directly. They calculate the global orientation of the head by integrating relative rotation measurements, so both system accumulate errors. The increasing drift will make our hybrid tracking device inoperative after a period of time. This drift should be corrected, and we may modify our video tracker to work also as a drift corrector (You et al. (1999) [18] and Chen and Pinz (2004) [19] have worked in this approach).

Pablo Berges del Arco

Lausanne, 22nd April 2005

Bibliography

[1] T. B. . M. of Thought, "Siggraph 2001 course 11."

[2] B. Martin, "Quaternion fractal animator," Master's thesis, 1999.

[3] Y. Abdeljaoued, D. Marimon, and T. Ebrahimi, *3D Videocommunication*, ch. Tracking and User Interface for Mixed Reality. WILEY & SONS, 2005.

[4] G. Baratoff and S. Blanksteen, "Tracking devices.." http://www.hitl.washington.edu/scivw/EVE/I.D.1.b.TrackingDevices.html.

[5] P. Fischer, R. Daniel, and K. V. Siva, eds., *Specification and Design of Input Devices for Teleoperation*, vol. vol.1, IEEE, International Conference of Robotics and Automation, 1990.

[6] G. Schwann, "Physically based animation in an augmented reality environment."

[7] R. Azuma and G. Bishop, "Improving static and dynamic registration in an optical see-through hmd," in *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 197–204, ACM Press, 1994.

[8] T. Auer and A. Pinz, "Building a hybrid tracking system: Integration of optical and magnetic tracking," in *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, (Washington, DC, USA), p. 13, IEEE Computer Society, 1999.

[9] G. Welch and G. Bishop, "An introduction to the kalman filter," tech. rep., Chapel Hill, NC, USA, 1995.

[10] L. Kleeman, "Understanding and applying kalman filtering."

[11] P. S. Maybeck, *Stochastic models, estimation and control*. Academic Press, 1979.

[12] R. Azuma, "Tracking requirements for augmented reality," *Commun. ACM*, vol. 36, no. 7, pp. 50–51, 1993.

[13] A. State, G. Hirota, D. T. Chen, W. F. Garrett, and M. A. Livingston, "Superior augmented reality registration by integrating landmark tracking and magnetic tracking," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 429–438, ACM Press, 1996.

[14] S. Emura and S. Tachi, "Multisensor integrated prediction for virtual reality," vol. 7, pp. 410–422, 1998.

[15] S. Feiner, B. MacIntyre, and T. Höllerer, "A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment," (Washington, DC, USA), p. 74–81, IEEE Computer Society, 1997.

[16] R. Behringer, "Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors," in *VR '99: Proceedings of the IEEE Virtual Reality*, (Washington, DC, USA), p. 244, IEEE Computer Society, 1999.

[17] R. Azuma *et al.*, "Tracking in unprepared environments for augmented reality systems," *Computers and Graphics*, vol. 23, no. 6, pp. 787–793, 1999.

[18] S. You, U. Neumann, and R. Azuma, "Hybrid inertial and vision tracking for augmented reality registration," in *VR '99: Proceedings of the IEEE Virtual Reality*, (Washington, DC, USA), p. 260, IEEE Computer Society, 1999.

[19] J. Chen and A. Pinz, "Structure and motion by fusion of inertial and vision-based tracking," in *Digital Imaging in Media and Education* (W. Burger and J. Scharinger, eds.), vol. 179 of *Schriftenreihe*, pp. 55–62, OCG, 2004. Proceedings of the 28^{th} ÖAGM/AAPR Conference.

[20] Y. Yokokohji, Y. Sugawara, and T. Yoshikawa, "Accurate image overlay on video see-through hmds using vision and accelerometers," in *VR '00: Proceedings of the IEEE Virtual Reality 2000 Conference*, (Washington, DC, USA), p. 247, IEEE Computer Society, 2000.

[21] S. You and U. Neumann, "Fusion of vision and gyro tracking for robust augmented reality registration," in *VR '01: Proceedings of the Virtual Reality 2001 Conference (VR'01)*, (Washington, DC, USA), p. 71, IEEE Computer Society, 2001.

[22] P. Lang, M. Ribo, and A. Pinz, "A new combination of vision-based and inertial tracking for fully mobile, wearable and real-time operation," in *Proc. of 26th Workshop of the Austrian Association for Pattern Recognition(ÖAGM/AAPR)*, vol. 160, (Graz, Austria), pp. 141–148, September 2002.

[23] F. eddine Ababsa and M. Mallem, "Inertial and vision head tracker sensor fusion usin particle filter for augmented reality systems," in *Circuits and Systems*, pp. 861–4, IEEE Computer Society, 2004.

[24] InterSense, Inc., 1 North Avenue Burlington, MA 01803, *Product Manual for use with InertiaCube2 Serial and USB Interfaces*, 2003.

[25] E. Foxlin, "Inertial head-tracker sensor fusion by a complimentary separate-bias kalman filter," in *VRAIS '96: Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS 96)*, (Washington, DC, USA), p. 185, IEEE Computer Society, 1996.

[26] R. Azuma, B. Hoff, H. N. Iii, and R. Sarfaty, "A motion-stabilized outdoor augmented reality system," in *VR '99: Proceedings of the IEEE Virtual Reality*, (Washington, DC, USA), p. 252, IEEE Computer Society, 1999.

[27] T. Bak, "Lecture notes - estimation and sensor information fusion." http://www.control.auc.dk/~tb/Teaching/Courses/Estimation/sensfusion.pdf, 11 2000.

[28] J. S. Goddard, *Pose and motion estimation from vision using dual quaternion-based extended kalman filtering*. PhD thesis, 1997. Major Professor-Mongi A. Abidi.

[29] M. D. Cordea, D. C. Petriu, E. M. Petriu, N. D. Georganas, and T. E. Whalen, "3d head pose recovery for interactive virtual reality avatars," in *Instrumentation and Measurement Technology Conference*, vol. 1, pp. 72 – 77, IEEE Computer Society, 2001.