Chapter 3

Software tools

In this section we are going to see all software tools used in the station and some tools that were study to include in the station but for one reason or another were finally discarded. Between these elements are Head-Tracking, 3D Sound and SPINE.

3.1 Head Tracking

3.1.1 Introduction

This section introduces head tracking system, defining its objective and motivations. It summarizes the state of the art in head tracking systems and presents the approach proposed in this section. The main advantages of the system with respect the state of the art are also detailed.

Objectives

The main objective of this section is to detail the implementation of a head tracking system suitable for its use in Teleoperation Stations, taking into account the limitations and constraints usually associated to these environments. The focus of the section will be centered on the multimodal technologies included in the human-machine interface for the ground command and control station. In particular, the section mainly describes the development of a robust head tracking system for its integration in Control Stations. Thus, the section firstly presents the last updates that have been carried out in the field of vision-based head tracking in the reporting period. Later, the section presents a new approach to integrate inertial sensors into the head tracking system in order to increase the reliability of the system.

Figure 3.1: *Left*: Tracker Pro (Madentec). *Center*: SmartNav 4 AT (NaturalPoint). *Right*: TrackIR 4 (NaturalPoint)

State of the art

Latest advances in technology and the growing computational capabilities of desktop computers have allowed the introduction of new devices for human-machine interaction. These devices usually provide functionalities to substitute the computer mouse. In the last years, new devices to estimate the real position of the user's head in real time under certain constraints have been developed.

Most of these devices rely on cameras and image processing algorithms. In general, they can be divided into two main application areas: 2DoF (Degrees of Freedom) and 6DoF head tracking. The 2DoF tracking products are focused on mouse emulation. They replace the standard computer mouse for people who cannot use their hands when controlling a computer or an augmentative communication device. On the other hand, 6DoF tracking is mainly oriented to gaming, allowing complete user immersion into different computer games.

2DoF head tracking applications and products are easy to find. Most of these products are based on image processing and marks/spots placed in the users' head (on a hat for instance). They also provide the two angles information used to move the mouse left/right and up/down. *Tracker Pro* [11] (see Figure 3.1) is a good example. This product is based on an USB camera and a software package. It is very reliable and has a wide field of view (about 45 degrees) and supports sunlight compatibility. Other examples are the *Headmouse Extreme* [16] or the *SmartNav 4 AT* [12] (see Figure 3.1).

6DoF head tracking moves one step forward and allows estimating the complete position and orientation of the user's head in real time. Most of the approaches can be divided into two groups: based on human face detection and based on visual pattern detection, both using image processing. Into the first group, several research works as [5], [4] or [7] have shown robust estimation processing stages of the position and orientation of the user's head. In those cases, model-based head detection is used to initialize the tracker and also to estimate the 6DoF localization of the head. The main concern is usually related to the reliability of the face detection stage. Actually, many recent works have been devoted to increase the robustness of this kind of approaches. For instance, in [2] particle filtering and complex

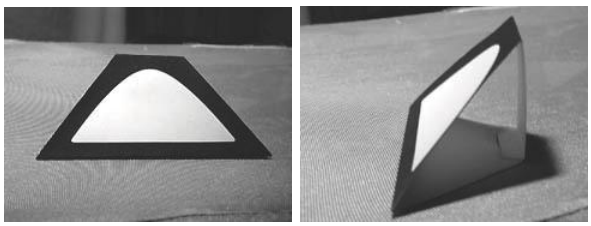


Figure 3.2: Visual pattern used in the Cachya head tracking system

tracking policies are used to implement a robust system.

The second group of 6DoF head tracking approaches makes use of some patterns/marks that allow simplifying the head detection process in the sequence of images. Thus, [8] uses infrared LEDs mounted on the user's head to localize it. A different approach is applied in [10], where the camera is mounted on the user's head and some landmarks are detected and used to localize the head. Nowadays, the commercial products are mainly focused in this kind of approaches and normally make use of camera and visual/IR patterns mounted on the head. *TrackIr* [13] is a good example of these systems (see Figure 3.1). It uses a 3D pattern visible in the infrared band to estimate the position and orientation of the user.

In our case, a new 6DoF head tracking system able to provide real time position and orientation of the head, minimizing the interferences with the user operations, is proposed. This is an aspect that differentiates Ivan and Fernando work from the above presented approaches. Thus, the previously introduced 6DoF products require a 3D pattern mounted on the user's head, as shown in Figure 3.2. They are normally attached to a hat that the user wears. Although it is common for gamers, operators are subject to hard constraints in terms of additional devices, i.e. they must be compatible with current systems like headphones, haptic systems, etc.

The design guidelines were focused on integration and robustness. To fulfill such constraints, the following system was proposed:

- A head tracking system based on the localization of an infrared pattern that the operator carries on the head. The reason to use infrared emission is that it is out of the visible band, so it is not perceived by the operator. Moreover, it is possible to use infrared filters to remove visual information, remaining only the infrared information and simplifying the pattern detection algorithms.

- The infrared pattern is integrated into the headphones used by the operator in order to avoid disturbing his working environment. Thus, the pattern is 2D and not 3D as usual in the previously described products and approaches.

To the best of our knowledge, this is one of the first implementations

of a 6DoF head-tracking system based on planar templates. The research is focused on a particular problem: the head-tracking in teleoperation stations. This environment poses very hard constraints in terms of robustness, usability and compatibility with already existing devices:

- Usability and compatibility are addressed by means of the proposed prototype based on an infrared planar template integrated into the user's headphones.

- Robustness is explicitly addressed in the approach by including marker tracking in the image space. This feature makes a difference with respect to the commercial devices in which environment disturbances such as sun light, reflections, halogen lamps or ir-remotes have a direct impact in the head-tracking estimation. The proposed tracking method allows rejecting such disturbances once the pattern has been detected. In addition, it allows decreasing the computational requirements for image processing because the filter prediction bounds the area in which the markers should be projected and hence, the processing can be applied only locally.

3.1.2 Head tracking system

This section details the design, taking into account several practical issues. The proposed head tracking process can be decomposed into the steps showed in Figure 3.3. First, an image of the environment is captured and processed by the system to prepare the detection of the infrared pattern. Then, the pattern is searched using two possible approaches: pattern detection or tracking considering previous information. The normal operation of the system will be to track the position of the pattern. If the tracking fails or there is not enough information to compute the tracking, then the system will try to detect it again.

Once the infrared pattern is detected in the image, the system will compute the homography matrix that relates the pattern and its projection, and this homography will be decomposed into the real position and orientation of the user's head.

The processing carried out in each step is further detailed in the next sections.

Image Capture

All the head tracking software developed is operating system independent up to the image capture level. For this purpose the `libdc1394` library for Linux has been used. This library provides a complete set of functions to manage any firewire camera that implements the DCAM protocol for machine vision,

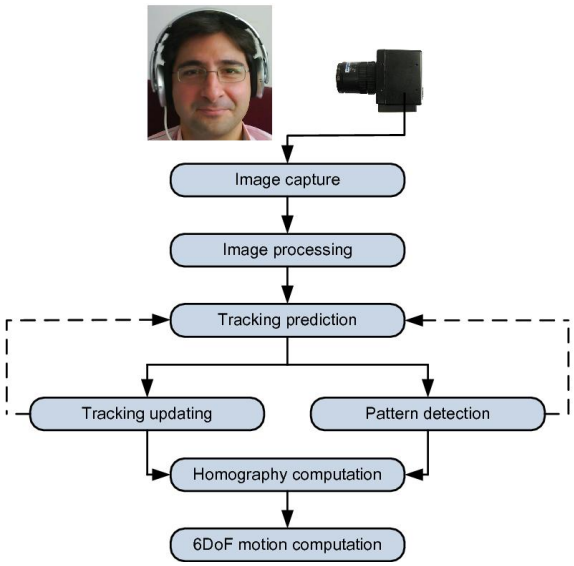


Figure 3.3: Different stages involved in the proposed head tracking system

from simple image capture to camera parametrization (shutter, exposure, gain, etc).

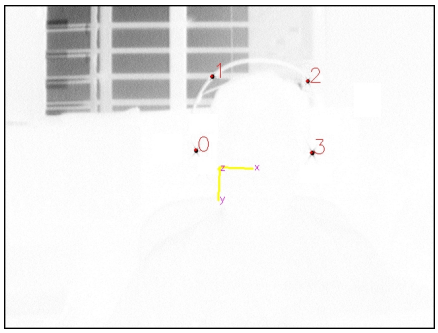
Thus, the firewire cameras ability of setting up image capture parameters such gain, shutter or iris allows implementing methods for camera self-configuration, making the system much more robust to changes in the lighting conditions.

The firewire image capture library is used in the software to capture images and to setup the following parameters: brightness, exposure, gamma, shutter and gain. All of them are set to zero in order to manually manage any image processing, so that our software can take the control of the entire image domain.

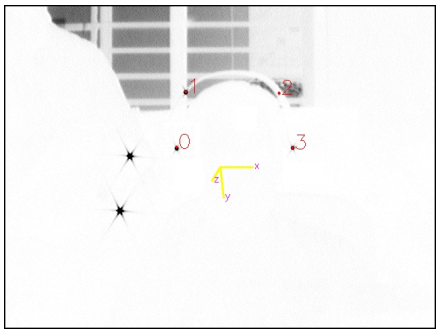
Regarding camera synchronization, an external digital signal can be used as trigger to ensure the image timing. However, given the low latency of the firewire bus triggering (lower than a microsecond) and assuming a static camera (which is the case), the firewire internal triggering is used in the implementation in order to simplify the camera setup. The camera is configured to capture images at 30 Hz, so the system will provide head tracking information every 33 ms.

Image Processing

The image processing stage is probably the most sensitive, but simplest stage, in the head tracking system. The goal of this stage is to process the image in order to detect the set of bright LEDs. This problem is particularly complex in the sense that depends on the camera environment.



(a) Detection in a room with a window in the background



(b) Detection in the same environment and two infrared remotes close to the headphones

Figure 3.4: Images showing the detection process during the operation of the system. The four infrared LEDs detected are marked by red dots over the raw images captured by the camera (notice that the colors are inverted). The computed coordinate frame is also shown as an overlay on the image captured.

As it was mentioned above, the camera integrates an infrared filter to suppress visible information from the image and pass the infrared pattern. However, infrared is present in many environments: daylight, lamps or infrared communication. The system is designed to be able to eliminate part of the disturbances induced by the environment.

Then, the camera, the filter and the infrared LEDs have to be carefully selected to obtain maximum gain into the bandwidth of interest. In these conditions, the images captured by the camera are similar to those presented in Figure 3.4. Four black spots, that represent the four emitting LEDs of the pattern, can be found (notice that the colors are inverted). Additional infrared information is also present in the images: sunlight from a window in the background and two ir-remotes.

The infrared detection is based on finding four maximums into the image (each maximum corresponding to an infrared LED). Additionally, these maximums will be subject to the following constraints:

- The grayscale of each maximum must be greater than a given threshold. Assuming that the infrared information provided by the LEDs is always greater than the infrared present in the environment, this threshold helps to separate between LED information and noise from the environment. In the current implementation, this threshold is set to 40 (15% of the maximum value that can be perceived by the camera). If there are no enough maximums greater than this threshold, the system drop off the image and cancel the head tracking estimation with that image.

- In a general case, the LED will be projected in the image as an ellipse, but not as a single pixel. The ellipse of each detected peak must have a minimal and maximal area. This information can be used to detect and eliminate potential outliers, allowing to reject mismatches produced by reflections or noise. However, given the elliptical nature of the detected peaks, the position of the LED must be computed as the centroid of such ellipse. Then, the position $[C_u, C_v]^t$ of the LED is finally given by

$$C_u = 1/N \sum_{k=0}^{N} p_u,\ C_v = 1/N \sum_{k=0}^{N} p_v\,, \tag{3.1}$$

  where N is the number of pixels that compose the ellipse, and p_u and p_v stand for the pixel column and row respectively.

- The distance among maximums must be greater than a given threshold. The idea behind this constraint is to avoid the selection of maximums too close to each other. This is very usual when the infrared emission is split due to the presence of objects such as hair, glasses, etc. This threshold has been set to 60 pixels. Notice that this threshold limits the distance at which the user can be located with respect to the camera. The current value allows standing at more than two meters from the camera.

These constraints are applied sequentially in the above order. Thus, a group of potential maximums from the first step will be obtained; they will be cut off depending on the size of the projected ellipse in the image and, finally, this sub-group of maximums is reduced to four taking into account the minimal distance constraint.

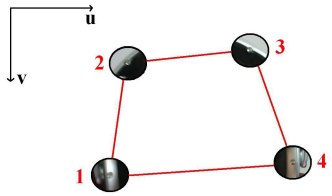
Figure 3.4 shows the result of applying this algorithm to different images. The four LEDs of the pattern (red spots) are detected and many outliers are rejected by the algorithm.

Infrared Pattern Matching

The previous step provides a set of points in the image that may correspond with the LEDs of the infrared pattern. This set of matches was filtered according to geometrical and gray scale constraints. However, the match between each LED and its projection in the image is still unknown. This section details how to match the infrared pattern with its projection in the image.

The matching between LEDs and projections is based on the geometrical constraints imposed by the pattern. Our pattern is composed of four LEDs mounted into the headphones of the operator, and has a trapezoidal shape (see Figure 3.5), i.e. the bottom LEDs (labelled as 1 and 4) are more separated than the two LEDs in the top (labelled as 2 and 3). This characteristic

(a) Infrared pattern integrated in the headphones



(b) Trapezoidal shape of the infrared pattern

Figure 3.5: Shape of the infrared pattern integrated in the headphones.

allows to easily sort out between bottom and top LEDs based on the relative distances.

Then, given the set of projections $\mathbf{p}_i = [u_i, v_i]^t$, $i = 1, \ldots, 4$, where u_i holds for the rows and v_i for the columns with the zero in the upper left corner of the image (see Figure 3.5), the first step consists on sorting the projections depending on the value of v_i from maximal to minimal values. Then, assuming that the pattern is rotated less than forty five degrees, the two first projections correspond to the bottom LEDs (labelled as 1 and 4 in Figure 3.5) and the other two to the upper projections (labelled as 2 and 3). Finally, it is possible to distinguish between the left and right projections depending on the value of u_i.

Later, constraints in u_i are used to verify the detection. Thus, given the matching between projections and LEDs in the pattern, and assuming a rotation less than forty five degrees, the projections belonging to LEDs 1 and 2 must have an u_i coordinate shorter than projections 3 and 4.

Then, this method is valid only when the pattern is rotated less than fortyfive degrees and tilted less than ninety degrees. It is easy to see that the LED/projection association is undetermined out of these limits. Nevertheless, this is not a hard constraint in the system because the operators's head will be normally within such limits. In addition, this limitation holds for the detection stage, but not for the tracking process detailed in next section.

Pattern Tracking in the Image Plane

Now, there is an initial estimation about the projection of the pattern into the camera and the next step is to track the position of these projections during the images sequence. The tracking of the projections allows to improve

the image processing and pattern detection thanks to the prediction phase. Thus, the predicted position of the projections will be used by the whole head tracking system to reject outliers and decrease the searching area in the image processing stage.

Then, the tracking process is divided into two basic parts: prediction and updating. The first step consists on predicting the position of the projections in the image and the second uses the current position of the projections (obtained by means of all the previous algorithms) to update the predictions.

It is proposed a Kalman Filter for each projection tracking. The filter will estimate the position and velocity of each projection in the image, whereas the covariance matrix associated to the projections will determine the searching areas and the candidates that can be used as projections. An independent Kalman Filter will be launched for each projection, being the state vector for projection i the following:

$$\mathbf{x}_i = [\mathbf{p}_i, \mathbf{v}_i]^t \tag{3.2}$$

where $\mathbf{p}_i$ and $\mathbf{v}_i$ are the position and velocity of the projection i in the current image expressed in pixels and pixels/s respectively.

During the prediction stage, it will be assumed that each Δt seconds, an instant perturbation in the velocity of the pixels will be produced ($\Delta\mathbf{v}$ holds for this perturbation). It will be assumed that the components of this velocity are independent Gaussians with zero mean and known standard deviation, so:

$$\Delta\mathbf{v} = \begin{bmatrix}\Delta v_u \\ \Delta v_v\end{bmatrix} = \begin{bmatrix}\mathbb{N}(0,\sigma_u^2) \\ \mathbb{N}(0,\sigma_v^2)\end{bmatrix} . \tag{3.3}$$

Then, the prediction model for the projection i, from time instant $k-1$ to k is given by

$$\begin{bmatrix}\mathbf{p}_i(k) \\ \mathbf{v}_i(k)\end{bmatrix} = \begin{bmatrix}\mathbf{p}_i(k-1) + (\mathbf{v}_i(k-1) + \Delta\mathbf{v})\Delta t \\ \mathbf{v}_i(k-1) + \Delta\mathbf{v}\end{bmatrix} . \tag{3.4}$$

The Kalman filter proposes the following equations to predict the new state and its covariance matrix:

$$\mathbf{x}_i^-(k) = \mathbf{A}_i\mathbf{x}_i(k-1) + \mathbf{B}_i\mathbf{u}_i(k-1) \tag{3.5}$$

$$\mathbf{P}_i^-(k) = \mathbf{A}_i\mathbf{P}_i(k-1)\mathbf{A}_i^T + \mathbf{Q}_i \tag{3.6}$$

By comparing with (3.4), it is easy to identify the matrices $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{Q}_i$ as

$$\mathbf{A}_i = \begin{bmatrix}1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1\end{bmatrix}, \quad \mathbf{B}_i = 0, \quad \mathbf{Q}_i = \begin{bmatrix}(\Delta t\sigma_u)^2 & 0 & \Delta t\sigma_u^2 & 0 \\ 0 & (\Delta t\sigma_v)^2 & 0 & \Delta t\sigma_v^2 \\ \Delta t\sigma_u^2 & 0 & \sigma_u^2 & 0 \\ 0 & \Delta t\sigma_v^2 & 0 & \sigma_v^2\end{bmatrix} . \tag{3.7}$$

In the system implementation, the value of Δt is dynamically computed during the program execution. A timer is launched when the filter updating is done and stopped when the filter prediction is computed, being Δt the elapsed time. This value is usually in the order of 0.33 ms.

Another important part in the prediction is the value estimation of σ_u and σ_v. In order to have a balance between conservativeness and efficiency in the prediction, this value is composed of two terms: a constant and a term that depends on the current filter state. Thus, such values are computed as follows:

$$\sigma_u = 1 + 0.2\hat{v}_u(k-1) \tag{3.8}$$
$$\sigma_v = 1 + 0.2\hat{v}_v(k-1) \tag{3.9}$$

On the other hand, the updating model considers the measurement provided by the detection algorithms. Then, the estimated position of the projection i will be directly mapped into the state vector $\mathbf{x}_i$. If $\mathbf{z}_i(k)$ is the measurement corresponding to the i-th projection, the updating equation can be expressed as

$$\mathbf{z}_i(k) = \mathbf{p}_i(k)\,. \tag{3.10}$$

The new measurement $\mathbf{z}_i(k)$ is considered in the Kalman Filter by means of the following equations:

$$\mathbf{K}_i(k) = \mathbf{P}_i^-(k)\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_i^-(k)\mathbf{H}_i^T + \mathbf{R}_i)^{-1} \tag{3.11}$$
$$\mathbf{x}_i(k) = \mathbf{x}_i^-(k) + \mathbf{K}_i(k)(\mathbf{z}_i(k) - \mathbf{H}_i\mathbf{x}_i^-(k)) \tag{3.12}$$
$$\mathbf{P}_i(k) = (\mathbf{I} - \mathbf{K}_i(k)\mathbf{H}_i)\mathbf{P}_i^-(k) \tag{3.13}$$

Considering (3.10), the following updating matrices can be derived:

$$\mathbf{H}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{R}_i = \begin{bmatrix} \sigma_{iu}^2 & 0 \\ 0 & \sigma_{iv}^2 \end{bmatrix} \tag{3.14}$$

where σ_{iu}^2 and σ_{iv}^2 stand for the variance associated to the detected projection i. Assuming that every projection includes a standard deviation of 2 pixels in the detection due to image saturation and binarization, this leads to $\sigma_{iu}^2 = 4$ and $\sigma_{iv}^2 = 4$.

6DoF Pose Estimation

This section details the computation of the user's head position and orientation by using the projection provided by the tracking system. Two basic steps are carried out: first, the mathematical model that relates the projection with the infrared pattern is computed (this model is a homography),

and then, the computed homography is decomposed into rotation and translation.

The following notation will be used: A 2D point in the image plane is denoted by $\mathbf{m} = [u, v]^t$ and a 3D point in the world system reference is denoted by $\mathbf{g} = [x, y, z]^t$. This system reference is the frame in which the position of the infrared pattern will be finally expressed. In the case of the application presented in the Chapter, this frame is attached to the main screen of the operation station. It will be used the symbol ˜ to denote the augmented homogeneous vector generated by adding 1 as the last element: $\tilde{\mathbf{m}} = [u, v, 1]^t$ and $\tilde{\mathbf{g}} = [x, y, z, 1]^t$.

Homography Computation

Knowing the matching between pattern and image projections, it is necessary to fit them a motion model, minimizing the error. The model used is the homography. Thus, assuming that the pattern is planar (which is our case), the projection in the image is related with the pattern through the following expression:

$$k\tilde{\mathbf{m}}' = \mathbf{H}\tilde{\mathbf{m}} \quad \text{with} \quad \tilde{\mathbf{m}}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{m}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{3.15}$$

where $\mathbf{H}$ is a $3x3$ non-singular matrix called homography and which is defined up to a scale factor k. This means that the homography matrix depends on 8 parameters. Four correspondences are needed to determine one homography because each correspondence gives two equations to solve the system.

If a set of four matches between the pattern and the image are given by the image processing algorithms, the following equations can be extracted for each match:

$$(h_{31}u + h_{32}v + h_{33})u' = h_{11}u + h_{12}v + h_{13} \tag{3.16}$$

$$(h_{31}u + h_{32}v + h_{33})v' = h_{21}u + h_{22}v + h_{23} \tag{3.17}$$

and reordering the expression we have

$$\begin{bmatrix} u & v & 1 & 0 & 0 & 0 & -uu' & -vu' & -u' \\ 0 & 0 & 0 & u & v & 1 & -uv' & -vv' & -v' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3.18}$$

Finally, it is possible to stack the two equations provided by each projection, building a 9×9 matrix $\mathbf{A}$, and compute the parameters of the homography $\mathbf{H}$. However, it is important to note that the corresponding system of equations is homogeneous, so it cannot be solved using the classical least squares approach. In this case, the solution to the system of equations is given by the right singular vector of $\mathbf{A}$ associated with the smallest singular value.

Motion Estimation

If the camera is modelled by the usual pinhole model, the relationship between the augmented homogeneous vectors of a 3D point $\tilde{\mathbf{g}}$ and its image projection $\tilde{\mathbf{m}}$ is given by:

$$k\tilde{\mathbf{m}} = \mathbf{C}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{g}} = \mathbf{C}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{t}]\tilde{\mathbf{g}} \tag{3.19}$$

where k is an arbitrary scale factor, matrix $[\mathbf{R}|\mathbf{t}]$ (called the extrinsic parameters) contains the rotation matrix $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ and the translation vector $\mathbf{t}$ which relates the world coordinate system to the camera coordinate system, and $\mathbf{C}$ is the camera calibration matrix.

Without loss of generality, it is assumed that the pattern plane is on $z = 0$ of the world coordinate system, allowing us to derive the following expression:

$$k \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{C}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{t}] \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \mathbf{C}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} . \tag{3.20}$$

In turn, it can be assumed that $\tilde{\mathbf{g}} = [x, y, 1]^t$ while $\mathbf{r}_3$ is computed as the cross product of the computed $\mathbf{r}_1$ and $\mathbf{r}_2$. Therefore, a pattern point $\tilde{\mathbf{g}}$ and its projection $\tilde{\mathbf{m}}$ are related by a homography $\mathbf{H}$ according to

$$k\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{g}} \quad \text{with} \quad \mathbf{H} = \mathbf{C}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}] . \tag{3.21}$$

Then, if the homography $\mathbf{H}$ that relates the infrared pattern and the projection into the camera is known, it is possible to recover the full rotation and translation of the pattern with respect to the camera. Thus, it can be easily computed from

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3] \tag{3.22}$$
$$\mathbf{r}_1 = \lambda \mathbf{C}^{-1}\mathbf{h}_1 \tag{3.23}$$
$$\mathbf{r}_2 = \lambda \mathbf{C}^{-1}\mathbf{h}_2 \tag{3.24}$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{3.25}$$
$$\mathbf{t} = \lambda \mathbf{C}^{-1}\mathbf{h}_3 \tag{3.26}$$

with $\lambda = 1/\|\mathbf{C}^{-1}\mathbf{h}_1\| = 1/\|\mathbf{C}^{-1}\mathbf{h}_2\|$. In general, due to the noise in the data, the so-computed matrix $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ does not satisfy the properties of a rotation matrix. It can be demonstrated that the closest (in mean squares terms) rotation matrix to the above solution will be determined by $\mathbf{R} = \mathbf{U}\mathbf{V}^t$, where $\mathbf{U}$ and $\mathbf{V}$ come from the singular value decomposition of the above estimation $[\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3] = \mathbf{U}\mathbf{S}\mathbf{V}^t$.

Further mathematical details and demonstrations can be found in [21].

3.1.3 Integration of inertial sensors into head tracking system for robust estimation

In the previous Section, the updates carried out on the vision-based head tracking system have been described. But additionally, the vision-based system has been extended with inertial sensors in order to increase its robustness. Then, this Section describes the details of the improved head-tracking system.

Introduction

A vision-based head tracking system using a single camera and a planar IR pattern has been designed, implemented and tested. It was described in [1] (previous reporting period) and also in Section 3.1 of this Section. This system was able to compute the full position and orientation of an operator in real time at 30 Hz. The experimental results showed that the accuracy of the approach was good enough to be usable in a Ground Control Station to provide the state of the operator in real time. Also, new modalities to communicate with the computer were described, such as the so-called Virtual Head Mounted Display.

Nevertheless, the system presented in [1] has some limitations. Due to the field of view of the camera installed in the Control Station, the operator's position and orientation computation is limited to a certain volume in front of the station.

This section describes how the integration of the vision-based head tracking measurements with inertial and gyroscope sensors can be used to increase the reliability, robustness and operational space. Thus, the system will be able to provide full position and orientation information even if the operator is out of the field of view of the camera thanks to the integration of the sensors.

Approach overview

The approach can be divide into the following three steps (depicted in Fig. 3.6):

1. *Position and orientation estimation using visual information.* The techniques described in Chap. 3.1 are used to provide an estimation of the head position and orientation using visual means.

2. *Orientation correction.* Once a first guest of the operator orientation is provided, gyroscopes can be used to make a short time prediction based on the rotation rate of each axis. In this step, the accelerometers will be also used to have a ground truth in roll and pitch angles. The yaw ground truth is provided by the visual system.

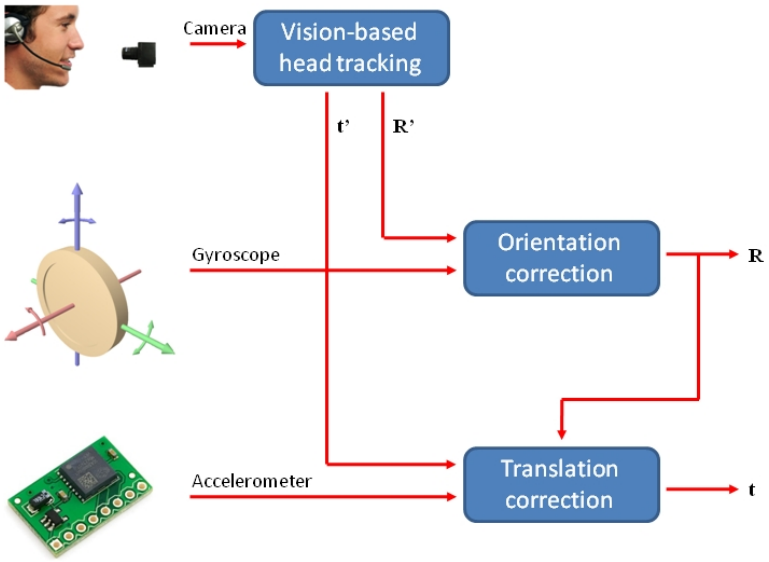


Figure 3.6: Approach overview. The output of the system are the operator's head orientation $\mathbf{R}$ and translation $\mathbf{t}$. $\mathbf{R}'$ and $\mathbf{t}'$ stand for the orientation and translation computed by the vision-based head tracking system

3. *Translation correction.* From the previous steps we have an estimation of the translation and a stabilized orientation. Now, this orientation can be used to rotate the measured acceleration to align it with the operator system reference. Then, the acceleration can be integrated in time to have a short time prediction of the translation. Again, the information provided by the visual system is used as a ground truth for the estimation.

Next sections will describe the second and third steps. A Kalman filter will be proposed for orientation and translation correction. This type of filters will allow to integrate the measurements taking into account the intrinsic noise in both inertial sensors and visual system.

Orientation correction

Assuming that an initial estimation of the orientation with respect to the Ground Control Station is provided by the vision-based head tracking system, this step will refine the orientation by means of the integration of gyroscopes for a short term prediction and accelerometers for long term estimation in roll and pitch.

This document proposes to filter the above information considering the intrinsic errors of the sensors. A Kalman Filter will be used for this purpose. This filter can be divided in two stages:

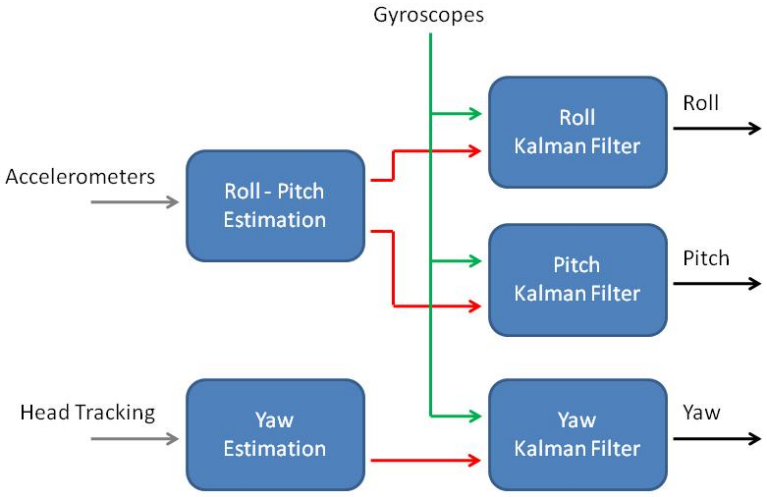


Figure 3.7: Orientation correction system. The approach processes the gyroscopes, accelerometers and visual information to produce a compensated estimation of the operators's Roll, Pitch and Yaw. Notice that the Kalman Filters have the same formulation but with different data-feeds.

1. *Prediction.* The gyroscope information will be used to predict the state of the orientation based on the angle rates in each axis and the time horizon. This prediction will not only provide an estimation of the orientation at a very high rate, but also an estimation of its uncertainty. This uncertainty will be used later in the filter updating to optimally integrate the accelerometer and vision-based measurements.

2. *Updating.* This stage will allow to integrate sensor readings provided by the accelerometers and the vision-based system. These measurements, although noisy, provide a global estimation of the orientation of the operator's head. Particularly, accelerometers enable a stable computation of roll and pitch angles, whereas vision-based head tracking will provide the yaw angle of the operator's head.

Assuming that the operator's roll, pitch and yaw orientation angles are statistically independent, a separate filter can be implemented per axis. Figure 3.7 shows a scheme of the proposed approach for the orientation correction in which the three filters can be seen. The Kalman Filters used in the orientation correction have the same formulation but with different sensor feeds. In the next paragraphs this Kalman Filter will be detailed.

State vector

The measurements of the gyroscopes are composed by the addition of two factors; the first one is proportional to the rotation rate in the measured

axis and the second is an unknown bias value. The bias evolves smoothly with time and temperature, so it should be computed at the beginning and periodically updated by the filter.

Thus, the state vector will be composed by the estimated orientation angle (roll, pitch or yaw) and the associated gyroscope bias. It can be written as follows:

$$\mathbf{x} = [\theta, b]^T, \tag{3.27}$$

where θ is the angle expressed in degrees and b the estimated gyro bias.

The associated covariance matrix $\mathbf{P}$ will be the following 2×2 matrix:

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} \\ p_{01} & p_{11} \end{bmatrix}. \tag{3.28}$$

Notice that it has been considered the symmetry of the covariance, so $p_{01} = p_{10}$.

Prediction stage

The proposed filter will provide predictions based on the gyro values every Δt seconds. Thus, given a gyro measurement g over a particular axis at time t, the predicted orientation in the next time step $t + \Delta t$ will be given by the following expression:

$$\theta = \theta + (g - b)\Delta t. \tag{3.29}$$

Considering the definition of the state vector, this equation can be re-written in matrix form as follows:

$$\mathbf{x} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} g. \tag{3.30}$$

Then, according to the prediction equations of the Kalman Filter, the covariance matrix $\mathbf{P}$ associated to the state vector $\mathbf{x}$ will be modified as follows:

$$\mathbf{P} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \mathbf{P} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \sigma_g^2 \begin{bmatrix} \Delta t & 0 \end{bmatrix} \tag{3.31}$$

where σ_g^2 stands for the variance of the additive errors included in the gyroscope estimation.

This stage will be repeated every Δt seconds with the latest value of the gyroscope sensor. It can be seen how this prediction cannot be maintained for a long period of time because the additive errors inherent to the gyroscope measurement will eventually make diverge the angle θ estimation. To avoid this problem, global measurements provided by accelerometers and vision-based head tracking will be used.

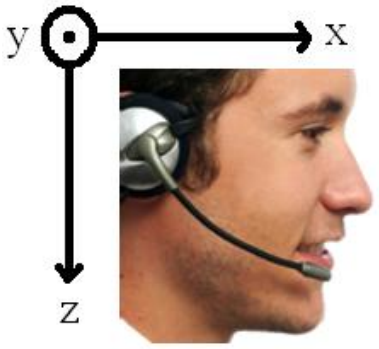


Figure 3.8: Reference system attached to the operator's head

Updating stage

Measurements

The information of the accelerometers could be also used to estimate the orientation of the operator's head if a smooth motion of the head is assumed (in this case the measured acceleration matches with the projection of the gravity vector). Under this assumption, the values of the roll and pitch angles can be computed from a 3-axis accelerometer according to the following formulae:

$$roll = \arctan(a_y/\sqrt{a_x^2 + a_z^2}) \tag{3.32}$$

$$pitch = \arctan(a_x/\sqrt{a_y^2 + a_z^2}), \tag{3.33}$$

where a_x, a_y and a_z represent the components of the acceleration vector. The reference frame is attached to the operators' head following the usual right-handed aeronautics system represented in Figure 3.8. *roll* and *pitch* stand for the rotation around the x and y axes respectively.

Notice how the accelerometers provide a global measurement of roll and pitch angles, so they are not subject of additive errors as gyroscopes. Nevertheless, accelerometers can only be used to estimate the orientation if and only if the operator's head is moving smoothly, otherwise, the accelerometer will include the acceleration of the gravity plus the acceleration in the movement axis.

To avoid erroneous estimations of roll and pitch based on accelerometers, the module of the acceleration can be used to distinguish whether the operator's head motion is smooth or not. Thus, the acceleration module $\| a \|$ is computed as:

$$\| a \| = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3.34}$$

In general, the acceleration will be included into the filter if the module satisfies the constraint $0.9 \leq \| a \| \leq 1.1$.

On the other hand, accelerometers cannot provide measurements on the yaw angle. The yaw angle estimated by the vision-based head tracking system will be used as global measurement in this case.

Filter updating

Now that we have an estimation of the angle ground-truth (wherever it cames: accelerometers or head tracking), this measurement has to be incorporated into the filter to correct the gyro-based estimation.

Assuming a measurement (z_θ, σ_z^2) of the orientation angle is provided by the sensors, the filter must be updated according to the equation:

$$z_\theta = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}. \tag{3.35}$$

From this equation it is possible to compute the Kalman Gain and to optimally update the filter according to its state and the measurement z_θ, and we obtain:

$$\mathbf{x} = \mathbf{x} + \begin{bmatrix} p_{00}(z_\theta - \theta)/(p_{00} + \sigma_z^2) \\ p_{01}(z_\theta - \theta)/(p_{00} + \sigma_z^2) \end{bmatrix} \tag{3.36}$$

$$\mathbf{P} = \mathbf{P} - \frac{1}{p_{00} + \sigma_z^2} \begin{bmatrix} p_{00}p_{00} & p_{00}p_{01} \\ p_{00}p_{01} & p_{01}p_{01} \end{bmatrix}. \tag{3.37}$$

The value of the variance σ_z^2 associated to the measurement z_θ will be different depending on the source of information. Normally, the errors associated to the vision-based orientation will be smaller than the ones provided by the accelerometers.

Filter initialization

Finally, this section deals with the initialization of the three filters (roll, pitch and yaw). Thus, the initial value of the state vector and the covariance matrix will be detailed.

In the case of the state vector, two are the variables to be initialized: angle and bias. The procedure is the following:

- The filter is initialized when the first angle measurement arrives. The angle θ of the state vector is forced to be equal to the angle measurement z_θ.
- Once the angle is initialized, the bias value is set to the current value of the gyroscope sensor g.

On the other hand, the covariance matrix P is initialized as follows: Angle and bias are assumed statistically independent in the initialization, so the covariance matrix is a diagonal matrix ($p_{01} = p_{10} = 0$). The remaining two elements are given by:

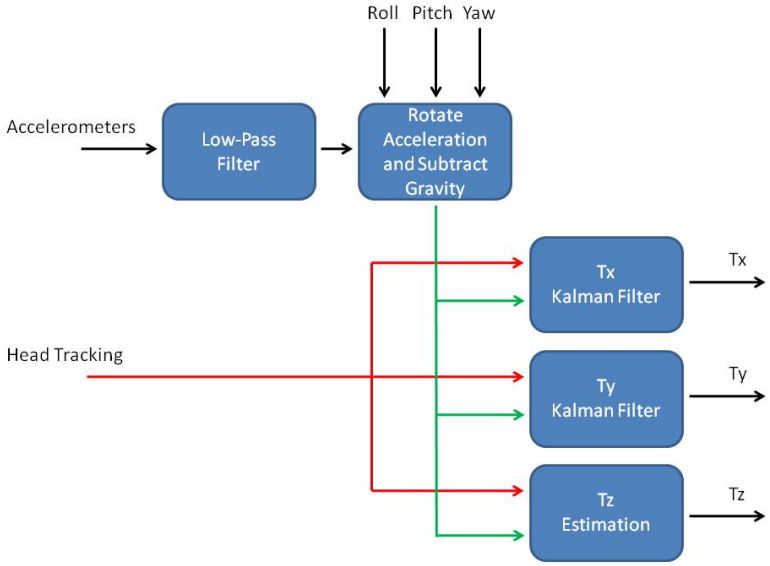


Figure 3.9: Approach for accelerometers filtering. Notice that the Kalman Filter has the same formulation for each translation but with different data feeds.

- p_{00}: It is set to the uncertainty of the measurement σ_z^2.
- p_{11}: It is set to a small value that guarantees a slow convergence to the real bias value during the data filtering. It is very usual to set it as $p_{11} = 0.001$

Translation correction

Once the orientation of the operator's head has been estimated, the translation with respect to the Ground Station can be refined based on the accelerometers. The idea behind is similar to the approach adopted with the Strapdown Inertial Measurement Units [19], that is, the stabilized orientation is used to rotate the accelerometers to a known reference system and then the acceleration can be integrated through time to provide a short term estimation of the translation.

The approach is depicted in Figure 3.9. Prior to integration, the accelerations are filtered to eliminate high frequency noise and then rotated according to the current system orientation. Once rotated, the projection of the gravity acceleration vector is known and can be removed from the measured acceleration, remaining the undergo motion of the operator's head. Finally, this filtered acceleration can be integrated through time to provide an estimation of the operator's head translation.

The integration of accelerations will be carried out by means of a Kalman

Filter. As in the orientation correction, this filter will have the same formulation in each axis but with different sensor feeds.

Low-pass filtering

Solid state accelerometers usually have high frequency noise associated to their estimations. Considering that the accelerations will be used to integrate the user position, this noise must be removed from the sensor because otherwise, the impact in the estimation could be relevant.

Several approaches allow to eliminate such noise, but taking into account the characteristics of the environment in which the sensor will gather the information, a low-pass filter seems to be the better solution. Thus, assuming a common behavior in the movements of the operator's head, it is possible to filter the accelerometers signal to remove/attenuate frequencies higher than expected.

A very simple first order low-pass filter is proposed in this case. Considering that the dynamics of the operator's head will be mainly below frequency F_c, the filter can be computed as follows:

$$y = (\Delta t a_1 + \Delta t a - \Delta t y_1 + 2y_1/F_c)/(\Delta t + 2/F_c); \tag{3.38}$$

where a and a_1 are the current and previous time step accelerations respectively. y and y_1 stand for the current and the previous time step filter outputs respectively. Finally, Δt is the time step in seconds.

For head tracking, it has been found that a value $F_c = 100\,\text{Hz}$ can properly filter the noise of the accelerometers without compromising the accelerations.

Acceleration rotation and gravity cancelation

Once the accelerations are filtered, they have to be rotated to a known reference frame in order to subtract the gravity acceleration vector components. For this purpose, the output of the orientation correction will be used.

The output of the orientation correction stage are the stabilized roll, pitch and yaw that transform the system reference showed in Figure 3.8 to the reference frame attached to the operator's head. These angles are called "Tait-Bryan Angles" and can be used to compute the associated rotation matrix:

$$\mathbf{R}' = \begin{bmatrix} cos(p)cos(y) & -cos(p)sin(y) & sin(p) \\ sin(r)sin(p)cos(y) + cos(r)sin(y) & -sin(r)sin(p)sin(y) + cos(r)cos(y) & -sin(r)cos(p) \\ -cos(r)sin(p)cos(y) + sin(r)sin(y) & cos(r)sin(p)sin(y) + sin(r)cos(y) & cos(r)cos(p) \end{bmatrix} \tag{3.39}$$

where r, p and y stand for the roll, pitch and yaw values respectively.

However, we are interested in the rotation from the frame attached to the operator's head to the reference system shown in Figure 3.8. Then, we

need to invert the rotation matrix $\mathbf{R}'$ (which is equal to its transpose), and the final rotation matrix is given by

$$\mathbf{R} = \mathbf{R'}^T = \begin{bmatrix} cos(p)cos(y) & sin(r)sin(p)cos(y) + cos(r)sin(y) & -cos(r)sin(p)cos(y) + sin(r)sin(y) \\ -cos(p)sin(y) & -sin(r)sin(p)sin(y) + cos(r)cos(y) & cos(r)sin(p)sin(y) + sin(r)cos(y) \\ sin(p) & -sin(r)cos(p) & cos(r)cos(p) \end{bmatrix}. \tag{3.40}$$

With the rotation matrix $\mathbf{R}$, the accelerometers can be rotated to a reference system in which the value of the gravity vector is known. According to Figure 3.8, the gravity vector will be given by $\mathbf{g} = [0, 0, 9.8]^T$ m/s^2 and the gravity free acceleration in the fixed reference system can be computed as:

$$\mathbf{a}_{gf} = \mathbf{R}\mathbf{a} + \mathbf{g}, \tag{3.41}$$

where $\mathbf{a}$ represents the original acceleration vector and $\mathbf{a}_{gf}$ is the gravity free acceleration vector.

The new acceleration vector can be used now to integrate the velocity and the position in a Kalman Filter in order to interpolate the estimation of the vision-based head tracking.

Kalman Filter for acceleration integration

This section describes the proposed Kalman Filter for acceleration integration. As with the orientation, the filter will be split into three filters (one per axis), each with the same formulation but different data feeds.

State vector

The state vector should consider the following constraints:

- Position. The filter must account for the current position of the operator's head. This variable will accumulate the integration provided by the accelerations and the position information given by the vision-based head tracking system.
- Velocity. Given that the acceleration must be integrated two times to compute the translation, accounting for the velocity into the filter will help to estimate the errors and propagate information.
- Bias. The accelerometers will always be misaligned with respect to the vision-based measurements. If this error is not compensated, it will be included into the position integration and will lead to permanent errors in the translation estimation. To avoid this, the bias will be estimated into the filter and compensated.

According to these constraints, the state vector for a single axis will be structured as follows:

$$\mathbf{x} = [t, v, b]^T, \tag{3.42}$$

where t, v and b stand for the current translation, velocity and bias in a particular axis respectively.

Prediction stage

As with the orientation correction, the sensor readings (in this case the acceleration in one axis) will be used to predict the state of the filter. The acceleration will be integrated two times to estimate both velocity and position. Thus, the corresponding prediction equations will be given by:

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}a; \tag{3.43}$$

where a is the measured acceleration in the corresponding axis and matrices $\mathbf{A}$ and $\mathbf{B}$ are the following:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & -\Delta t^2 \\ 0 & 0 & -\Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \Delta t^2 \\ \Delta t \\ 0 \end{bmatrix} \tag{3.44}$$

The covariance matrix $\mathbf{P}$ is updated following the usual prediction equations of the Kalman Filter, that is

$$\mathbf{P} = \mathbf{A}\mathbf{P}\mathbf{A}^T + \mathbf{Q}, \tag{3.45}$$

where $\mathbf{Q}$ is the process covariance matrix that contains the noise added by the sensor in the prediction equation. Assuming that the sensor is affected by noise whose variance is given by σ_a^2, matrix $\mathbf{Q}$ takes the following value

$$\mathbf{Q} = \begin{bmatrix} \Delta t^4 \sigma_a^2 & \Delta t^3 \sigma_a^2 & 0 \\ \Delta t^3 \sigma_a^2 & \Delta t^2 \sigma_a^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.46}$$

Updating stage

The filter updating will be carried out based on the vision-based measurements. This system provides position estimation in each axis at a lower rate than accelerometers do, but with a bounded error. Thus, the vision-based computed translation allows to fix the estimated translation and, indirectly, allows also to refine the estimated velocity and bias.

Then, the measurement will be the actual head translation in a particular axis and its variance, (z_t, σ_z^2). The updating equation can be written easily in matrix form as follows:

$$z_t = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{x}. \tag{3.47}$$

From this equation it is easy to obtain the Kalman Gain that optimally computes the state vector and covariance matrix updating.

Filter initialization

Finally, this section deals with the initialization of the state vector and the covariance matrix. Thus, the state vector is initialized with the first vision-based measurement (z_t, σ_z^2) as follows:

- The translation t is set to the mean value of the measurement z_t.
- The velocity v is set to zero.
- The bias b is also initialized to 0.

The covariance matrix also is initialized with the first measurement. The result is a diagonal matrix in which the variance of the translation t is set to σ_z^2, the variance of the velocity is set to 0.1 and the variance of the bias to 0.01.

3.2 SPINE

3.2.1 Introduction

Was raised using a biometric sensor network on the user of the station, in that way the system could know the state of the user all the time. That could give us some advantages, like to know if the user is stressed or if the user is too much relaxed (maybe slept) for example. For this reason was decided to study the option to include a Software tool that allows us to monitor and control the sensor network. This tool was SPINE [9].

SPINE (Signal Processing in Node Environment) is a software Framework for the design of Wireless Sensor Network applications. SPINE enables efficient implementations of signal processing algorithms for analysis and classification of sensor data through libraries of processing and utility functions and protocols. SPINE allows decrease development time and improves interoperability among applications through libraries of components of typical WSN systems specified in nesC and developed in TinyOS environment. Libraries include:

- a library of features computing parameters of the sensor data such as variance, mean or range of the sensor data
- an over-the-air protocol that allows the coordinator of a WSN to dynamically request the computation of specific features to the sensor nodes and obtain the result
- a set of utility functions such as a circular buffer and a sorting algorithm

SPINE enables efficient implementations of signal processing algorithms by providing a flexible way to allocate tasks among the WSN nodes. For example, SPINE allows compute features on the sensor nodes and uses an over-the-air protocol to send the computation results to the WSN coordinator. This implementation allows the coordinator to request the computation of features only when needed.

The objective of the SPINE open source project is to build a community to further develop the Framework and make it a valuable tool for the design of signal processing intensive WSN applications.

3.2.2 Technical description

The SPINE Framework relies on a WSN architecture including one or more sensor nodes and a WSN coordinator. The WSN coordinator typically performs functions such as managing the WSN nodes, collecting and analyzing the data received from the sensor nodes, and connecting as a gateway the WSN with a wide area network (WAN) for remote data access.

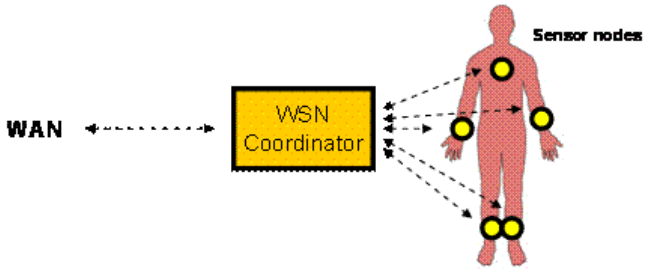


Figure 3.10: Schematic SPINE

SPINE has two main SW components: one to be executed on the sensor nodes and the other on the BSN coordinator.

The sensor node component, designed in TinyOS environment and written in nesC language, includes several utilities for signal processing such as data storage buffers, mathematical function libraries and common feature extractors used in signal processing. Furthermore, it includes an over-the-air communication protocol to transfer data from the sensor nodes to the WSN coordinator.

The coordinator component consists of a Java-based interface that an application running on the gateway itself or on a remote server can use to manage the sensor nodes or make service requests. This lightweight Java API is easily portable to devices of various capabilities, such as a PC or a mobile phone.

The SPINE framework offers developers great flexibility in the implementation of distributed signal processing algorithms for the analysis and the classification of sensor data. Some applications are based on complex algorithms that require an implementation on nodes such as gateway devices with sufficient computational resources. Other applications are based on rather lightweight algorithms that can be implemented in a distributed manner, i.e. with some functions executed on the sensor nodes.

SPINE supports both centralized and distributed implementations and therefore offers designers the flexibility to select the implementation approach that is most suitable to meet the requirements of the application. It includes a Feature Selection Protocol (FSP) that can be used by the gateway to request the sensor nodes to send back the result of locally computed features. Computing features locally and sending the results instead of transmitting the raw sensor data offers several advantages such as a more efficient utilization of the bandwidth of the wireless medium and savings of the energy of the nodes. In addition to requesting a node to send the results of features computed locally, FSP can also be used to specify:

- The interval over which a feature is to be calculated.

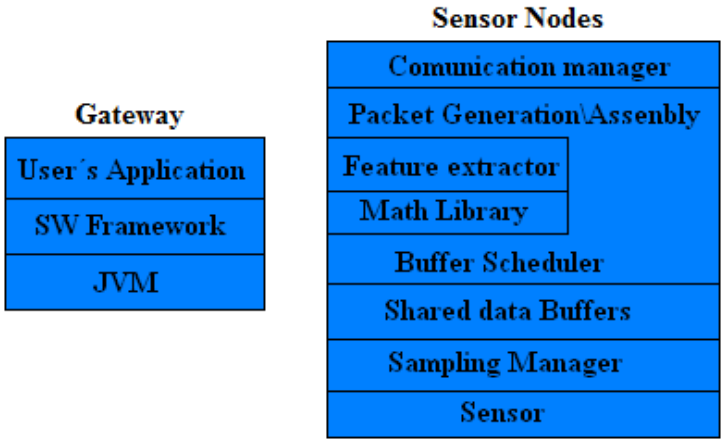


Figure 3.11: SPINE System

- The frequency with which the nodes should send the data to the gateway, e.g. 1) at regular intervals, 2) upon request, 3) when the values reach a specified threshold

Another important feature of SPINE is the reusability of nodes in different application scenarios. The service discovery function at the WSN coordinator allows it to recognize the functions of sensor nodes that have already been configured. This flexibility allows the same node to be deployed in many different application scenarios without reconfiguration of the embedded code.

The first release of SPINE has been used for the design of assisted living services such as the monitoring of limb movements of a person and more in general the recognition of a person's activities and postures. The system is based on a sensor board with 3-axis accelerometers and gyroscopes. The classification algorithm is implemented in TinyOS environment using the SPINE libraries.

The current libraries include features relevant to the analysis of data from accelerometers and gyroscopes (Figure 3.12), as well as on-mote implementation of code that interfaces accelerometer and gyroscope sensors with internal buffers. However, the framework is not limited just to applications based on accelerometers, but can be easily extended, adding libraries with new features and interfaces, and used also in WSN applications based on other types of sensors.

In our particular case we were interested in a sensor network with biometric sensors (like we can see in the figure 3.12). In fact, SPINE can be used to monitor the state of patients and we can use that in our profit.

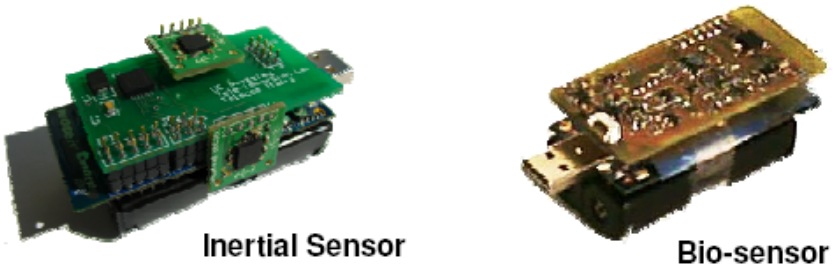


Figure 3.12: SPINE Compatible Sensors

But, finally, because the difficulties involved, the feeling that could prove cumbersome for the user of the station take over as many elements (remember that we are already using the Headphones and the vibrators) and the fact that the contribution of such biometric sensors is not so valuable. It was decided to discard this option. But if it would be necessary, this tool could be implemented in the future.

3.3 3D Sound

In a station is very important to combine the visual information with the auditory information. In that way the user can process more information given by the station. Some theories related to the Theory of Attention speak about the viability of performing two or more tasks simultaneously. As the theory of Multiples Resources, it is possible to carry out some tasks simultaneously if these tasks require different attention resources. So, it is possible to be looking at a screen for warnings and listen a warning and react to it. But, for example, it is rather difficult to perform a linguistic task (repeating a text, counting in reverse order, etc.) and a spatial task (detect a target on a video screen, etc.) simultaneously. And it is still more difficult to perform two linguistic tasks simultaneously (listening and reading), or two spatial tasks, especially if these tasks concern information presented in the same modality.

Taking account all above it was decided to use auditory warnings and messages in the station and for that aim we decided to use the Headphones. And, when the new functionality was installed in the Headphones (Head-Tracking) we can use it in our profit and exploit it building a new Software tool, 3D Sound. But, what is the 3D Sound?

Surround sound (or 3D Sound) encompases a range of techniques for enriching the sound reproduction quality of an audio source with audio channels reproduced via additional, discrete speakers. The three-dimensional (3D) sphere of human hearing can be virtually achieved with audio channels above and below the listener. To that end, the multichannel surround sound application encircles the audience (left-surround, right-surround, back-surround), as opposed to "screen channels" (center, [front] left, and [front] right), i.e. ca. 360ř horizontal plane, 2D).

Surround sound technology is used in cinema and home theater systems, video game consoles, personal computers and other platforms. Commercial surround sound media include videocassettes, Video DVDs, and HDTV broadcasts encoded as Dolby Pro Logic, Dolby Digital, or DTS. Other commercial formats include the competing DVD-Audio (DVD-A) and Super Audio CD (SACD) formats, and MP3 Surround. Cinema 5.1 surround formats include Dolby Digital and DTS. Sony Dynamic Digital Sound (SDDS) is a 7.1 Cinema configuration which features 5 independent audio channels across the front with two independent surround channels, and an LFE.

Most surround sound recordings are created by film production companies or video game producers; however some consumer camcorders have such capability either built-in or available separately. Surround sound technologies can also be used in music to enable new methods of artistic expression. After the failure of quadraphonic audio in the 1970s, multichannel music has slowly been reintroduced since 1999 with the help of SACD and DVD-Audio formats. Some AV receivers, stereophonic systems, and computer soundcards

contain integral digital signal processors and/or digital audio processors to simulate surround sound from a stereophonic source.

The 3D simulation is the most advanced group of 3D audio effects. Using head-related transfer functions and reverberation, the changes of sound on its way from the source (including reflections from walls and floors) to the listener's ear can be simulated. These effects include localization of sound sources behind, above and below the listener.

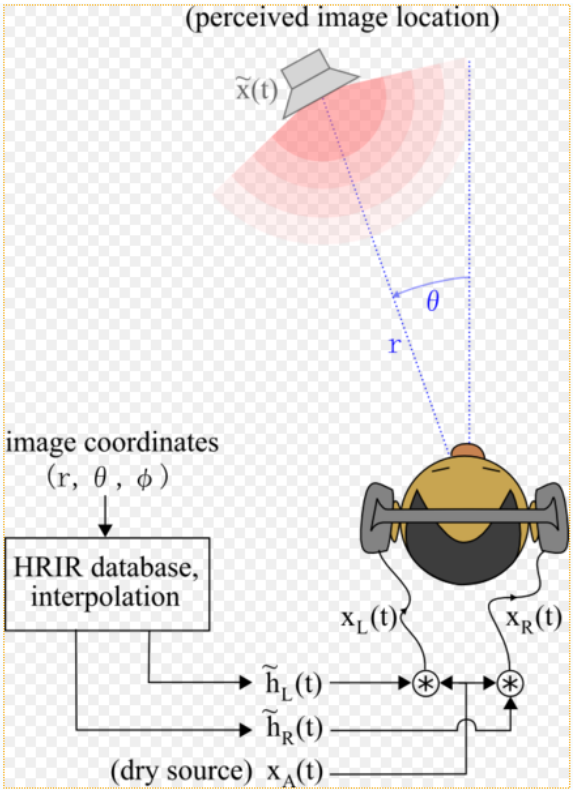


Figure 3.13: 3D Sound System

As we can see above, this is an emerging technology especially in the leisure field, but there are not reasons not to apply this technology in the industrial field. In our case we want to exploit the Head-Tracking system to get a more accurate 3D Sound system. The main advantage we can get with the implementation of 3D Sound in our system is the ability to locate a sound in the 3D space; initially it does not look like much, but as our system may have more than a single screen and due we always know the position of user's head (thanks to Head-tracking System), we can use it to generate a sound that seems to come from one of the screens. In that way, we can get that user does not miss important visual information. For example, we suppose that we have three screens, and a warning appear in the left screen

but, for any reason, the user is looking at the right screen; maybe user do not see the warning. With 3D Sound it would be possible to generate a sound in the Headphones that make the user believe that the sound comes from the left screen. These and other utilities will be seen further in the next section: **Possible applications** 4.

In order to schedule the 3D Sound, the smartest option looks to be to use OpenAL. OpenAL (for "Open Audio Library") is a software interface to audio hardware. The interface consists of a number of functions that allow a programmer to specify the objects and operations in producing high-quality audio output, specifically multichannel output of 3D arrangements of sound sources around a listener.

The OpenAL API is designed to be cross-platform and easy to use. It resembles the OpenGL API in coding style and conventions. OpenAL uses a syntax resembling that of OpenGL where applicable.

OpenAL is foremost a means to generate audio in a simulated three-dimensional space. Consequently, legacy audio concepts such as panning and left/right channels are not directly supported. OpenAL does include extensions compatible with the IA-SIG 3D Level 1 and Level 2 rendering guidelines to handle sound-source directivity and distancerelated attenuation and Doppler effects, as well as environmental effects such as reflection, obstruction, transmission, and reverberation.

Next step will be to transform the reference matrix from Head-Tracking to OpenAL matrix reference, in order to locate the user's head in the 3D Sound System and to schedule the sound that must be generated.