FRÉDÉRIC NTAWINIGA

HEAD MOTION TRACKING IN 3D SPACE FOR DRIVERS

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en génie électrique
pour l'obtention du grade de maître ès sciences (M.Sc.)

DEPARTEMENT DE GENIE ELECTRIQUE ET DE GENIE INFORMATIQUE
FACULTE DES SCIENCES ET DE GENIE
UNIVERSITE LAVAL
QUEBEC

2008

Résumé

Ce travail présente un système de vision par ordinateur capable de faire un suivi du mouvement en 3D de la tête d'une personne dans le cadre de la conduite automobile. Ce système de vision par ordinateur a été conçu pour faire partie d'un système intégré d'analyse du comportement des conducteurs tout en remplaçant des équipements et des accessoires coûteux, qui sont utilisés pour faire le suivi du mouvement de la tête, mais sont souvent encombrants pour le conducteur. Le fonctionnement du système est divisé en quatre étapes : l'acquisition d'images, la détection de la tête, l'extraction des traits faciaux, la détection de ces traits faciaux et la reconstruction 3D des traits faciaux qui sont suivis. Premièrement, dans l'étape d'acquisition d'images, deux caméras monochromes synchronisées sont employées pour former un système stéréoscopique qui facilitera plus tard la reconstruction 3D de la tête. Deuxièmement, la tête du conducteur est détectée pour diminuer la dimension de l'espace de recherche. Troisièmement, après avoir obtenu une paire d'images de deux caméras, l'étape d'extraction des traits faciaux suit tout en combinant les algorithmes de traitement d'images et la géométrie épipolaire pour effectuer le suivi des traits faciaux qui, dans notre cas, sont les deux yeux et le bout du nez du conducteur. Quatrièmement, dans une étape de détection des traits faciaux, les résultats 2D du suivi sont consolidés par la combinaison d'algorithmes de réseau de neurones et la géométrie du visage humain dans le but de filtrer les mauvais résultats. Enfin, dans la dernière étape, le modèle 3D de la tête est reconstruit grâce aux résultats 2D du suivi et ceux du calibrage stéréoscopique des caméras. En outre, on détermine les mesures 3D selon les six axes de mouvement connus sous le nom de degrés de liberté de la tête (longitudinal, vertical, latéral, roulis, tangage et lacet). La validation des résultats est effectuée en exécutant nos algorithmes sur des vidéos préenregistrés des conducteurs utilisant un simulateur de conduite afin d'obtenir des mesures 3D avec notre système et par la suite, à les comparer et les valider plus tard avec des mesures 3D fournies par un dispositif pour le suivi de mouvement installé sur la tête du conducteur.

Abstract

This work presents a computer vision module capable of tracking the head motion in 3D space for drivers. This computer vision module was designed to be part of an integrated system to analyze the behaviour of the drivers by replacing costly equipments and accessories that track the head of a driver but are often cumbersome for the user. The vision module operates in five stages: image acquisition, head detection, facial features extraction, facial features detection, and 3D reconstruction of the facial features that are being tracked. Firstly, in the image acquisition stage, two synchronized monochromatic cameras are used to set up a stereoscopic system that will later make the 3D reconstruction of the head simpler. Secondly the driver's head is detected to reduce the size of the search space for finding facial features. Thirdly, after obtaining a pair of images from the two cameras, the facial features extraction stage follows by combining image processing algorithms and epipolar geometry to track the chosen features that, in our case, consist of the two eyes and the tip of the nose. Fourthly, in a detection stage, the 2D tracking results are consolidated by combining a neural network algorithm and the geometry of the human face to discriminate erroneous results. Finally, in the last stage, the 3D model of the head is reconstructed from the 2D tracking results (e.g. tracking performed in each image independently) and calibration of the stereo pair. In addition 3D measurements according to the six axes of motion known as degrees of freedom of the head (longitudinal, vertical and lateral, roll, pitch and yaw) are obtained. The validation of the results is carried out by running our algorithms on pre-recorded video sequences of drivers using a driving simulator in order to obtain 3D measurements to be compared later with the 3D measurements provided by a motion tracking device installed on the driver's head.

Acknowledgement

I would like to give special thanks to my supervisor, Dr. Denis Laurendeau, for accepting me to join his team, and funding me while working on this project throughout my graduate studies. His guidance, help and suggestions are highly appreciated. I can say without any doubt that this work would not have been possible without him.

I would like to say a big “thank you” to my Dad, who never ceased to encourage me and who offered to revise the English of my manuscript. My father always wanted a graduate education but because of first supporting us when we were little, he couldn't achieve his dream. Today, I am so proud to achieve that dream for him, and to give this gift to my father as a way of saying, thanks, Dad!

I would like to send special thanks to the Wecker family. I will always appreciate how they let me be apart of their family. Without their help, I could not have done my bachelor studies and go on to complete my master's degree.

I would also like to thank Ruban family, Mbonyumubanyi family and my friend Fabrice Désiré Kagame for their help in so many ways.

To all my teachers at Université Laval, computer laboratory colleagues, research professionals Denis Ouellet, Thierry Moszkowicz and Sylvain Comtois, I say thank you.

Finally, and above all, I am very thankful to my God for everything.

To Kim,

To my mum and dad,

To my brothers and sisters,

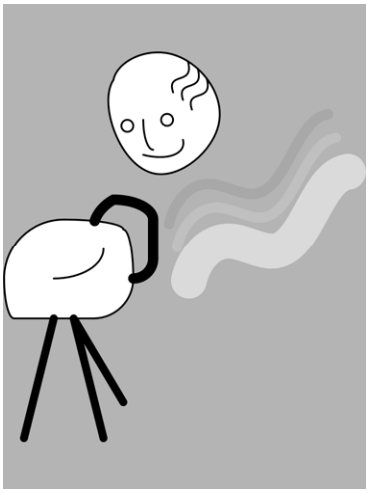
To my joint family and friends.

Table of content

List of tables

List of figures

CHAPTER 1: Introduction



Car driving has become a daily routine for most people in our society. As the number of drivers and cars increases, people are very concerned with public safety issues. There are two categories of “at-risk” drivers as far as driving is concerned: young or inexperienced drivers and older drivers with decreasing abilities. Since driving constitutes an important activity for drivers, it is important to detect “at risk” drivers and to propose retraining programs in order to improve their driving skills. This can be achieved in three steps: retraining the driver, optimizing the driving environment and optimizing the vehicle [1].

Identifying “at risk” drivers and retraining them in a safe environment can be achieved by using a driving simulator. In addition, making the vehicles more intelligent can help older drivers in facing potentially dangerous manoeuvres.

1.1 Project context and challenges

This project is part of a framework that is being developed by a multidisciplinary team in the context of a project called Cephalo-Ocular Behaviour and Visual Search Patterns of Drivers (COBVIS-D). The ultimate goal of COBVIS-D is to exploit driving simulators with realistic driving scenarios for evaluating the ability of elderly drivers and design experiments for retraining. The simulator uses a real automobile cockpit (Fig. 1.1, 1.2) with a steering wheel, a driver seat, rear view mirrors, brake and accelerator. A projector installed over the cockpit projects dynamic driving scenarios on a screen located in front of the driver and creates a reasonable level of immersion to the driver. Two devices are also used. A motion tracking device by Ascension Technology (Flock of Bird) is installed on the driver's head. The second device is positioned in front of the driver for tracking his eyes (Applied Science Laboratories (ASL) R6-HS eye-tracker).

Fig. 1. 1. Simulator cockpit with cameras, eye tracker and infrared source.

These devices provide information on the position of the head and the gazing direction of the eyes and are used for analysing the driver's cephalo-ocular behaviour (see Appendices A and B). The rationale for developing the computer vision-based solution proposed in this thesis is to provide a cost-effective software solution that can replace these costly devices (44,950 US $). The computer vision module uses two cameras for head and gaze tracking. A third monochromatic camera in the middle of the first two cameras is used for analysing the driver's facial expressions. Placed in the middle, this camera has two advantages: the whole driver's face is covered without needing the fusion of images from the first two

cameras to get full facial expression information. Furthermore, the facial expression analysis research can be done independently from this project. Infrared lighting is added to the setup in order to illuminate the drivers' faces without blinding them.

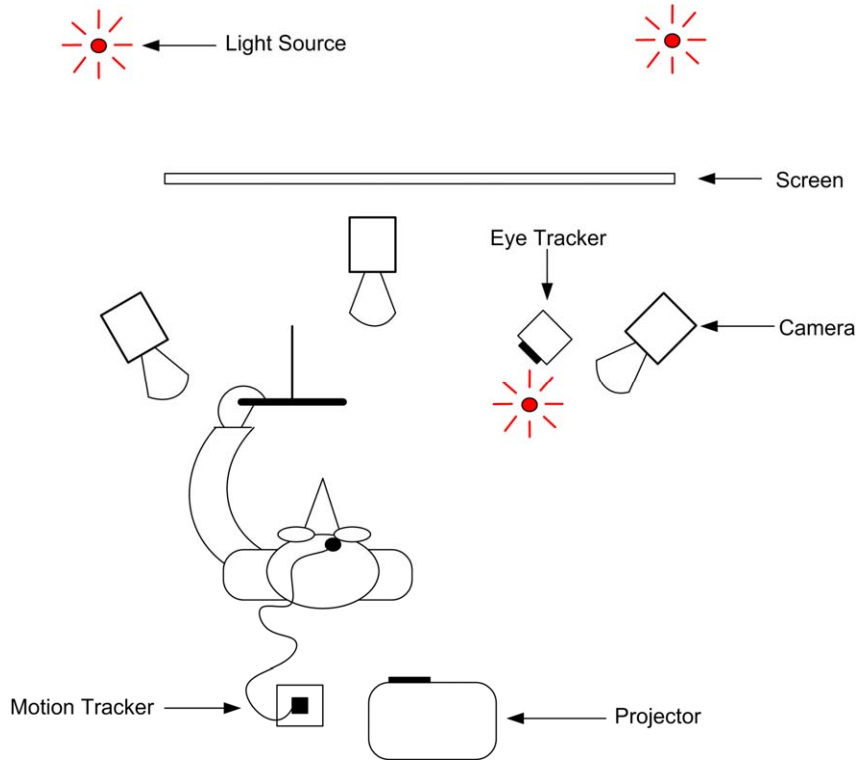


Fig. 1. 2. Cockpit Setup Diagram

1.2 Computer vision overview

Computer vision is becoming more popular as the power of computers increases. This is confirmed by the fact that computer vision applications are not restricted to academic laboratories but can now be found in numerous industrial systems: industrial robot, autonomous vehicle, visual surveillance, information organization, industrial inspection, medical analysis, topographical modeling, and computer-human interaction to name a few [2].

Most of the applications can be divided in two categories: image analysis/understanding applications which focus on still images and tracking applications for which a video sequence is also being analyzed for understanding its contents but for which motion is also of interest (e.g. the dynamic content of the sequence is equally relevant). The project discussed in this thesis belongs to the second category with a focus on object tracking and pose estimation (e.g. detecting the head of the driver in the simulator, tracking the head with respect to time, and estimating its position and orientation in Cartesian space). The following section presents an overview of the techniques that are commonly used for head tracking and pose estimation.

1.3 Head Pose Tracking

Various techniques have been proposed in the literature for head tracking and head pose estimation in video sequences. The systems developed from these techniques have a great range of applicability such as virtual reality, entertainment, physiological sciences studies, transportation, and computer-human interaction systems.

The majority of head tracking systems can be classified in two categories: systems that rely mostly on hardware such as wearable devices carried on the subject's head to help the tracking [12-15] and computer vision based systems exploiting off-the-shelf video cameras and image analysis software. [5-11]. Although the systems in the first category are usually very accurate, they are often cumbersome for the user and/or very expensive. It is expected

that a computer vision system made of off-the-shelf components will provide a cost-efficient non-contact solution for estimating reliable head pose information.

To describe the cephalo-ocular behaviour of drivers, the pose of the head in Cartesian space must be computed. The pose consists of the position (3 degrees-of-freedom) and orientation (three degrees-of-freedom) of an object in space. In computer vision, the use of two cameras observing an object from two different non-collinear vantage points (called a stereoscopic arrangement) allows to recover the three dimensional information required for pose estimation, provided that the images collected by the cameras are synchronized. Not all systems use stereo information and rather track the head in 2D [9-11].

Other implementations use 3D models for the tracking [7-8]. However, since model-based systems are not as accurate as stereo systems, both approaches are sometimes combined. In [6], stereo was used with colour cameras. In the system presented in this thesis, only near-infrared monochromatic cameras were used because they are sensitive to the infrared illumination that does not blind drivers. Gorodnichy [5] used stereo combined with epipolar geometry for increasing tracking robustness but the automatic detection of facial features was not considered in his implementation.

The system proposed in this thesis combines all the advantages enumerated above with the additional feature that it can be operated in real-time. As most computer vision systems, it consists of several processing stages: image acquisition, head detection, facial features detection and extraction using epipolar geometry, and finally, 3D reconstruction (for pose estimation) of the facial features that are being tracked (see Fig. 1.3).

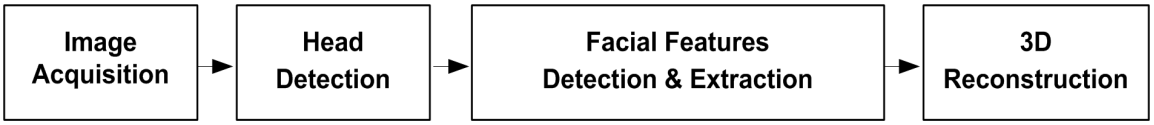


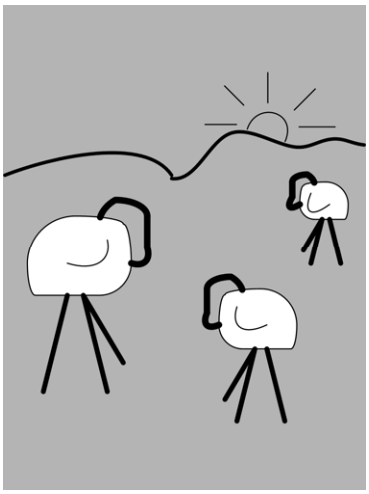
Fig. 1. 3. Processing steps of the proposed system for head pose estimation.

This thesis consists of six chapters. Chapter 2 presents the image acquisition system that was briefly described in the preceding paragraphs. Chapter 3 describes the detection of the head in the images of the stereo pair in order to limit the search space that will be used for finding facial features. Chapter 4 and 5 cover Facial features detection and extraction

modules. Chapter 6 describes the procedure for the 3D reconstruction of the facial features and the filter used for accurate tracking. The results of experiments are presented and discussed in Chapter 7. Finally Chapter 8 provides an overview of the presented implementation and recommendations for future work.

The main original contributions of the present research are: 1) the development of the software for head motion tracking in 3D space and eyes tracking, 2) the development of the Vertical Frequency Filter (VFF) which helps in providing a solution of the problem of eye tracking for people wearing glasses.

CHAPTER 2: Description of the Image Acquisition Module



To establish a relationship between the scene and the computer system, image acquisition is needed to collect and store the data as images for a later analysis.

2.1 Introduction

Image acquisition is a common stage for all computer vision systems. Digital images are obtained by using devices such as digital cameras and scanners. The light scattered by a scene is collected by a discrete sensor as proposed by Lally as early as 1961 [3]. The resulting signal is sampled and stored in computer memory for a later use. In the system presented in this thesis, standard off-the-shelf digital video cameras are used since, with proper image processing software, they allow the capture of 3D information on a scene, in our case, the driving simulator environment shown in Fig. 1.1.

This chapter is organized as follows: section 2.2 describes the basic concepts for image acquisition, section 2.3 presents the image acquisition analysis and finally section 2.4 provides a summary of the chapter.

2.2 Basic concepts for image acquisition

Image processing depends on the quality of the images obtained at the acquisition stage [4]. Every loss of information at this stage is difficult to recover at later stages of the processing chain. To achieve good image acquisition performance, many concepts such as lighting control in the environment, spectral component selection, camera colour space selection, camera position with respect to the scene of interest, camera frame rate, and camera synchronization must be taken under consideration during system design.

A robust image acquisition system should be able to adapt to a change in illumination as the human visual system does almost effortlessly. Robustness and adaptive behaviours allow the system to be dynamic and flexible at the cost of increased complexity. This complexity can be reduced considerably for a static environment for which lighting can be controlled by the user.

The selection of the spectral band for acquiring the images is also of great importance in the design of a computer vision system. In our case, system design must take into account that the driving scenario is displayed to the user using a projector and that good immersion precludes the use of controlled lighting in the visible part of the spectrum since this would bleach the images projected on the screen and may even blind the user of the driving simulator. However, reliable detection of facial features used for tracking would greatly benefit from using a lighting system that provides a larger dynamic range in the acquired images. A compromise must thus be found between the quality of immersion and the reliability and robustness of the detection of the facial features used for tracking.

In relation with controlled lighting (and spectral content), the sensitivity of the camera as well as the frame rate must be chosen so as to provide the best image possible for the application.

2.3 Image acquisition system analysis

In this context, the simulator environment into which the driver is placed is illuminated with infrared illuminators, located sideways in front of the cockpit (see Fig. 1.3). Two synchronized monochromatic cameras without infrared cut-off filter are used to allow stereoscopic capture of facial features. As shown in Fig. 2.1, the cameras are sensitive in the near infrared region of the light spectrum, by using infrared illuminators with the wavelength of 840nm and 940 nm, and the power varying from 6W to 55W (see Appendix D), it is safe to illuminate the driver's face and acquire good images of facial features without blinding him nor bleaching the images projected on the screen with the visible light as long as the infrared illuminators are placed at a distance where the driver does not feel the heat. This is made possible by the fact that human vision system presents a low sensitivity to infrared radiation while the monochromatic cameras used are sensitive to near infrared light spectrum.

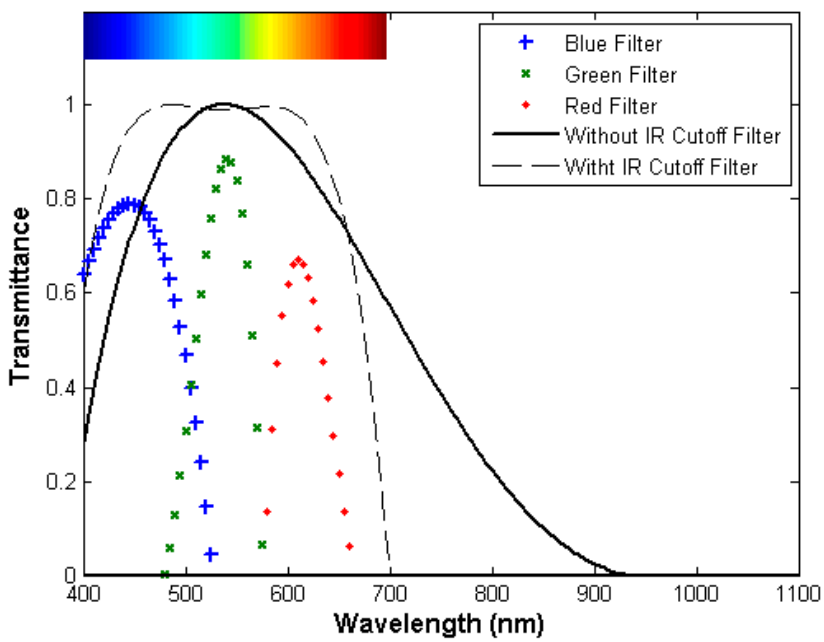
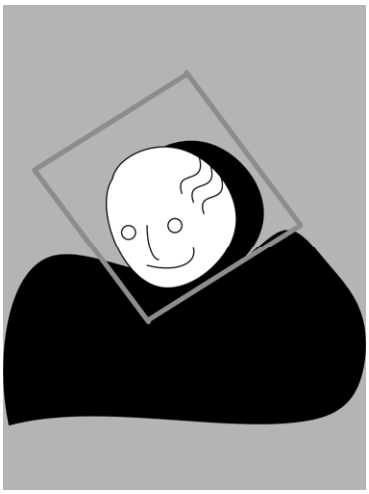


Fig. 2. 1 Camera filters

2.4 Summary

This chapter has presented the main components of an affordable image acquisition system used to track the head pose of the drivers in 3D. The system uses two Prosilica CV640 cameras as described in Appendix C. Through proper lighting control, the driver's face is illuminated with infrared projectors.

CHAPTER 3: Head Detection



To limit the search space that will be used for finding facial features, the head detection stage is needed. The detection will increase the video sequence processing speed and will eliminate potential noise similar to the facial features that are located outside of the area surrounding the head.

3.1 Introduction

Head detection is performed prior to the facial features detection and extraction. Once the head is detected, the localization of facial features is much easier because the head bounding box provides a space where the features are located.

To achieve head detection, a sub stage of background subtraction is needed. After removing the background, the remaining foreground will be the silhouette of the driver containing the head at the extreme top position of the silhouette.

This chapter is organized as follows: section 3.2 presents the literature review on background subtraction and head detection, section 3.3 presents the chosen concept and finally section 3.4 provides the summary.

3.2 Literature review

3.2.1 Background subtraction

Various techniques are used to implement background subtraction. These techniques can be applied to two different background categories: static and dynamic backgrounds [17]. For static backgrounds it is assumed that only the foreground changes, the background being stationary. For dynamic backgrounds, the background is more complex since the background and foreground can change at the same time. Examples of such dynamic backgrounds are the clouds in a scene or the movement of tree leaves because of the wind in an outdoors scene. In the context of the project we will be limited to the static backgrounds because the simulator is a stationary environment of the laboratory.

In his thesis, Lemieux [18] has classified background subtraction techniques in three categories depending on the camera sensor used: 2D Background subtraction in visible light spectrum, 2D Background subtraction by using infrared thermography and 3D background subtraction by using depth mapping.

3.2.1.1 Visible Light Spectrum

Various techniques have been used in visible light to subtract the background. Heikkila and Silven [19] used this model by marking the foreground pixel as in equation (3.1)

$$|I_t - B_t| > \tau \qquad (3.1)$$

where I_t is the current pixel, B_t the background pixel and τ the predefined threshold. The background is updated as in equation (3.2)

$$B_{t+1} = \alpha I_t + (1-\alpha) B_t \qquad (3.2)$$

where α is an importance coefficient over time.

Pixels can be defined in various colour spaces. In RGB (Red, Green and Blue), the colour space mostly used in display devices, each image pixel is composed of red, green and blue colour components in an additive model. However the use of other colour spaces such as

HSV (Hue, Saturation and Value) or HSL (Hue, Saturation and Luminance) can be more advantageous because the hue is invariant to changes of illumination.

Other techniques of background subtraction have been used but are not discussed in this thesis due to their complexity and computational load. Among these techniques are those which perform statistic modeling of every pixel. Others are inspired from the technique described above and are well described in [17].

3.2.1.2 Infrared Thermography

Infrared Thermography is used to locate object by the amount of heat emission in other words their temperature. The local infrared emission from the surface is recorded by an infrared detector according to the Stefan-Boltzmann law [20]. The recorded image is presented in monochromatic space where this space can be mapped to various different colour maps to distinguish the objects as shown in Fig. 3.1. The background is subtracted by applying a threshold in the temperature image. This technique has many advantages: The image content is simplified which speeds up processing, it is invariant to the change of illumination and can be used in the dark. The cost of infrared cameras and difficulties for implementing them in an off-the-shelf system, however, precluded their use for this project.

Fig. 3. 1 Infrared Thermography.

3.2.1.3 Disparity Maps

This technique is used in stereoscopic systems where the background is detected by considering that the foreground has a different depth from the background. However the real time implementation requires special hardware such as multiple high resolution cameras, C40 DSP array and the real-time processor board [21].

3.2.2 Head detection

Head detection techniques have been widely used in computer vision; these techniques include motion based methods [23], skin colour methods [27] and head-shoulder contour [22] and other advanced methods such as cascade of boosted classifiers for general objects detection [24, 25]. As the system has to run in real time, some methods such as motion based techniques were excluded because the driver does not move very much and there is no significant movement to be detected. Skin colour methods have been also excluded because the system uses monochromatic camera. The remaining two were carefully studied in order to choose the method that can run in real time while providing good head detection performance.

3.2.2.1 Cascade of boosted classifiers

It consists of classifiers trained to detect object rapidly with high detection rates. These methods use an image representation called "Integral Image" to allow features called Haar-like features (Fig. 3.2), to be computed very quickly. The Integral image ***I*** also known as summed area tables is an intermediate representation for the image and contains the sum of grey scale pixel values of image ***N*** with height **y** and width **x** as presented in equation (3.3).

$$I(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} N(x',y') \qquad (3.3)$$

This representation helps to calculate sum, mean and standard deviation over arbitrary up-right or rotated rectangular region of the image in constant time which makes possible to achieve a fast blurring or fast block correlation with variable window sizes [26]. This method uses simple classifiers in stages to discard bad region quickly and focus on

promising areas. The method provides an advantage of locating the head without requiring a background subtraction.

a)

b)

c)

Fig. 3. 2. Haar-like features a) Edge features b) Line features c) Center-surround features.

3.2.2.2 Head-shoulders contour

This method relies on the fact that the car driver moves very little which helps to maintain the shape of the head-shoulder contours as presented in Fig. 3.3. A proper threshold is applied to isolate the head and shoulders and, finally, the head is located to the extreme top part of the head-shoulder silhouette [22] [28].

3.3 Chosen approach

Computational performance between the Cascade of boosted classifiers and the Head-shoulders contour methods was the key factor in selecting the head detection algorithm for the simulator system. The head-shoulder contour method was chosen due to its good performance based on the processing time (Table 3.1).

As the head-shoulder contour method requires background subtraction, the decision to choose the best background subtraction method was simplified by the fact that the simulator is installed in a light-controlled environment; there was no need to use costly infrared cameras or complex disparity maps. Monochromatic cameras sensitive to near infrared light were used and a proper threshold was applied to grey images to isolate the foreground.

Table 3. 1. Performance Comparison.

Algorithm	**Left image**	**Right image**
	Processing time (ms)	
Our Head detection Algorithm	2.24	3.05
Cascade of boosted classifiers	79.36	73.30

The Head detection algorithm is described in a three-step process (appendix E):

- Image thresholding and binarization (for both the left and right image).
- Detection of the largest blob in the resulting binary image.
- Assignment of a rectangle (defined as the “head bounding box”) to the top part of the largest blob.

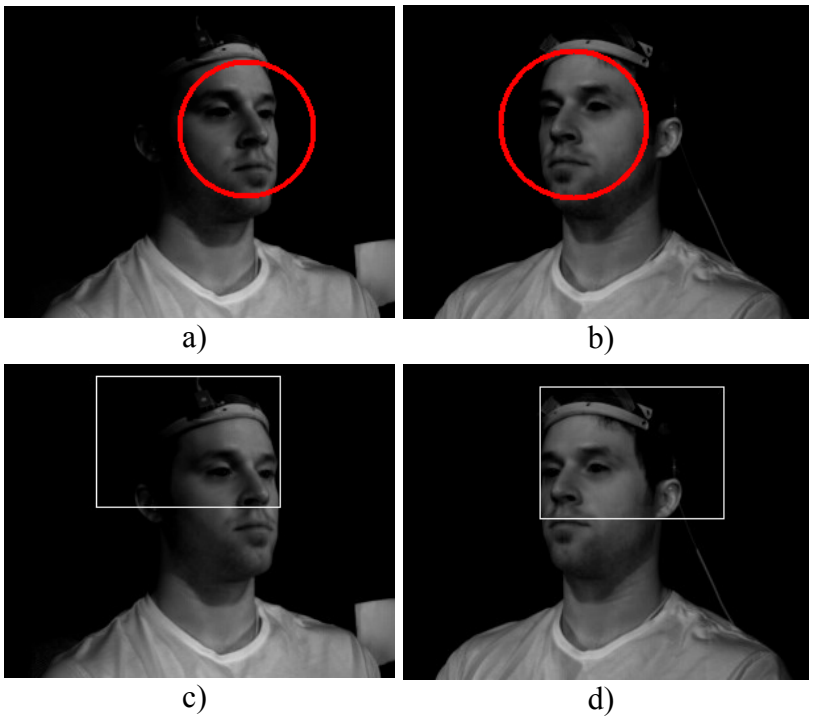
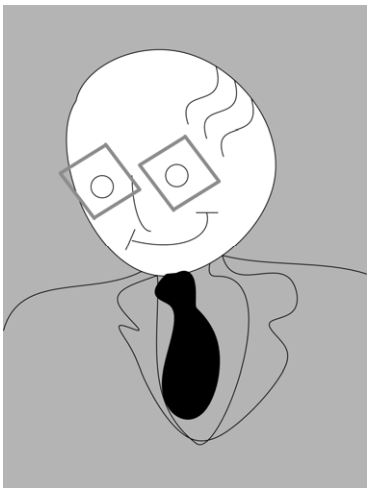


Fig. 3. 3. Head detection using: a, b) Cascade of boosted classifiers and c,d) Head-shoulder contour.

3.4 Summary

This chapter has presented a very fast head detection algorithm adapted to the light controlled environment of the driving simulation. Since the head detection was not the focus of the project, additional work on background subtraction and head detection algorithms would be necessary in order to have the system evolve from the simulator environment to a real driving environment such as a car. The head detection algorithm has increased the video sequence processing speed and eliminated potential noise which consists of areas which may look like facial features but are located outside of the head.

CHAPTER 4: Eyes Detection



Eye detection constitutes a processing stage that is performed prior to the facial features extraction. In this stage, regions of interest are obtained for further processing at the facial features extraction stage.

4.1 Introduction

To obtain the pose of the driver's head in 3D space, reference landmarks on the head need to be tracked in image plane (2D) for being used later to recover 3D pose of the head. Facial features can be defined as very pronounced parts of the face. This includes the hair, forehead, eyebrow, eyes, iris, nose, cheek, mouth, lips, philtrum, teeth, skin and chin [30] as shown in Fig. 4.1. These natural facial features can be added to artificial features such as eye glasses. The objective will be to detect and extract the most invariant features in different poses of the driver's head.

This chapter is organized as follows: section 4.2 presents the literature review on eye detection techniques, section 4.3 presents the chosen approach and, finally, section 4.4 provides the summary.

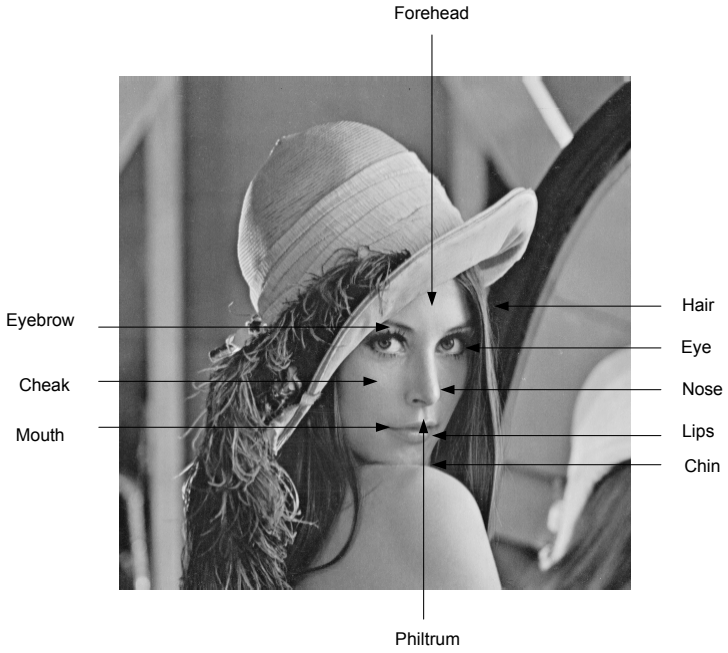


Fig. 4. 1. Facial features.

4.2 Literature review

4.2.1 Facial features detection

The head presents many facial features. However all of them are not good candidates for tracking. A good facial feature has to be visible and keep its shape during head motion or rotation. Gorodnichy [30] has examined the various facial features during head motion. He concluded that the nose is the only feature remaining clearly visible during head movement.

It is well known that a plane is spanned by two linearly independent vectors [31]. At least three points are needed to find these two vectors as shown in Fig. 4.2. This applies to our project, as the objective is to track the driver's head in 3D space. We thus need at least three facial features to determine the motion of the head in 3D space. As the nose tracking is mandatory, two other facial features need to be tracked to extract the two linearly independent vectors. Some facial features are too wide to provide a stable centroid. These features include the hair, the forehead, cheeks, skin and chin. Features like the mouth, lips and philtrum are often visible in head position and rotation poses, but do not keep their

shape and may change significantly when the driver is speaking or yawning. Among the remaining facial features of interest, the eyes are the best candidates because the iris is small and covered when eyes are blinking, teeth are also rarely visible and eyebrows may be absent especially for women wearing makeup.

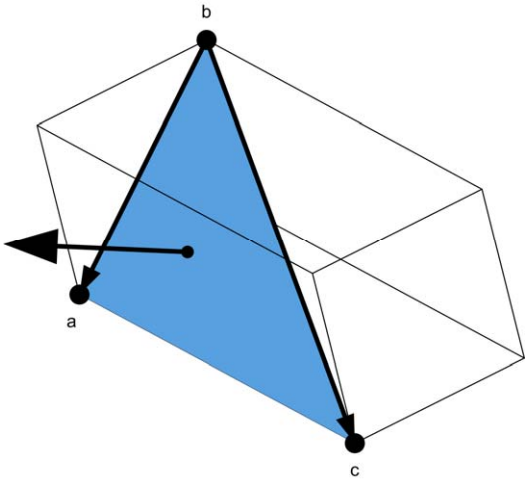


Fig. 4. 2. Plane passing through 3 points.

4.2.2 Eyes detection

Eyes detection has attracted a growing interest in research. This interest is explained by the use of eyes detection techniques in various applications such as human computer interfaces, in teleconferencing, virtual reality and other visual communication applications [32]. Eyes detection is also found in security applications such as surveillance face recognition and will be used more and more as security issues are increasing.

Various methods to detect the human eyes in a digital picture have been used. Among these, we have chosen the most representative to show the state of art in this field. These methods include colour-based, neural networks and Filter-based approaches.

Colour-based approaches: There are several ways of using colour as a cue to help eyes detection. Among them, we can find the work of Rein-Lien [33] who has proposed the most popular technique found in this category which consists in selecting the proper colour space for achieving a good detection. He used colour-based approaches by compensating the lighting during the detection phase to eliminate problems due to the presence of complex background and variations in lighting conditions. This compensation was performed by normalizing the colour appearance and choosing the YCbCr color space. YCbCr is one of colour spaces used in video systems, where Y is the luma (brightness) component and Cb and Cr are the blue and red chrominance components [34]. The choice of the YCbCr colour space helped to process a skin independently of the luminance. As in YCbCr colour space high Cb and low Cr values are found around the eyes, eyes are easily detected. However YCbCr is not the only colour space used for eye detection. Other colour spaces such as RGB (Red Green Blue) and HSV (Hue Saturation Value) have been used [36].

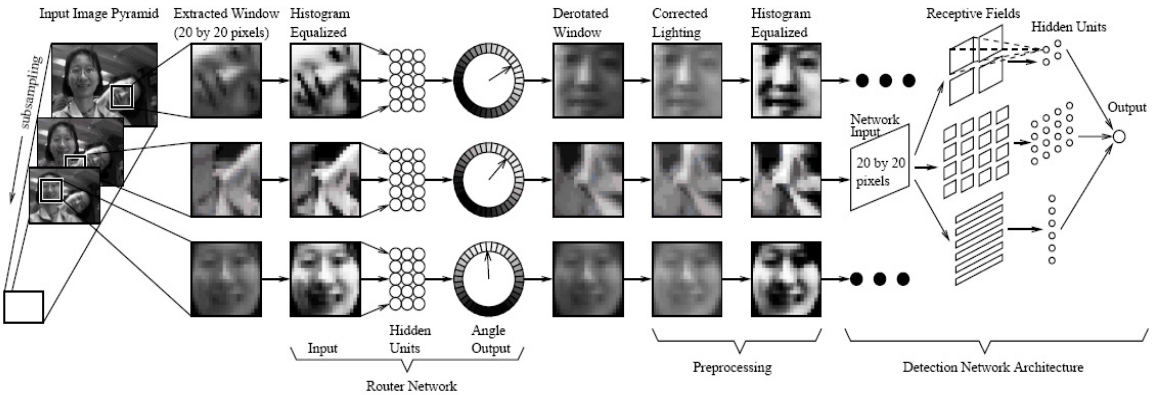
Feris [32] also used an approach which consists in generating a skin-colour distribution which is very common in colour-based approaches [36]. He chose to use a Gaussian distribution of skin-colour regions through supervised training. After thresholding and morphological closing followed by median filtering, facial features such as eyes were detected by template matching and other geometric constrains. Theses methods are mostly invariant to head rotation and can be used for real-time tracking. In this project, however, monochromatic images are used.

Neural networks approaches: Neural networks or artificial neural networks are parametric techniques modeled after biological brain neurons, to approximate a vector function of some inputs with a series of layers [35]. Each layer has a weight matrix, a bias vector and an output vector.

To use neural networks, images are divided in windows or grids with different sizes or scales depending on the application. Because illumination changes, the look of a face depends on the side that is illuminated. This illumination has to be removed before applying the neural networks. Rowley, Baluja and Kanade [37] used the neural network in two stages to locate face and eyes in front views with different rotations. The first stage was to estimate face orientation by using one neural network and the output had 36 units each

representing 10 degrees for a total of 360 degrees. In the second stage the rotation was corrected and a neural network with one output unit was used to determine whether or not there was a face as shown in Fig. 4.3.

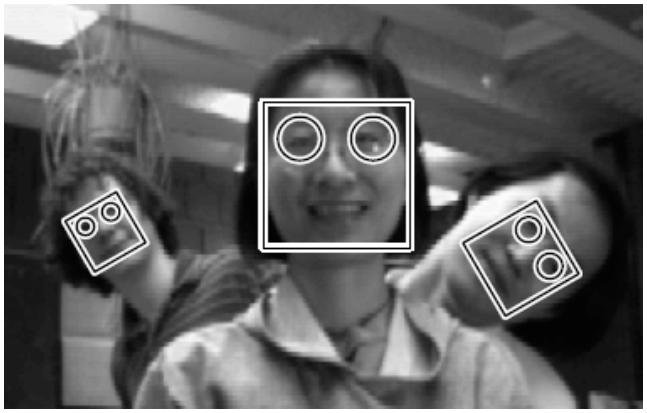
Fig. 4. 3. Face and eyes detection neural network architecture.



Source: Baluja and Kanade [37].

Finally a mask icon showing two eyes as two circles and the face as a rectangle was superimposed on each window as shown in Fig. 4.4.

Fig. 4. 4. Face and eyes detection neural network output.



Source: Baluja and Kanade [37].

Another reference in neural network for facial features detection was developed in the early 90s by Debevec [38] who used the log-polar mapping function to generate the training set. Instead of using traditional neighbourhood sampling of features with a tiny rectilinear grid of points, he extracted a circle with a certain radius around the feature and represented it in log-polar mapping as shown in Fig. 4.5.

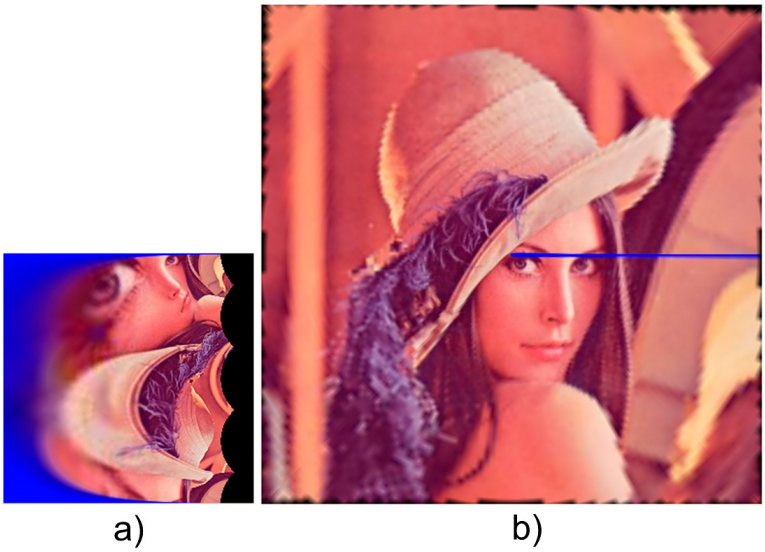


Fig. 4. 5. a) Forward log-polar transformation b) Inverse log-polar transformation.

The log-polar transformation emulates the human fovea vision and can be used for fast scale and rotation-invariant template matching. The output was formed with four neural network units representing the left eye, the right eye, the nose and the mouth.

Though the techniques described above had a very high detection rate and were suitable for greyscale images, they were mostly used for frontal faces. They are time consuming due to the training step of the neural networks that requires a very large amount of faces. As the Cephalo-Ocular Behaviour System involves in multi-face views, more robust techniques are needed.

Filter-based approaches: Some authors such as Kawato [39] have used a novel method for eyes detection by first detecting a region between the eyes as shown in Fig 4.6. Because there is a periodicity around the circle between brighter regions and darker regions, running the filter on circled pixels around each image pixel, the regions corresponding to the area between the eyes are detected. The filter that is used is similar to the Discrete Fourier Transform (DFT) as shown in equation (4.1).

$$f(x,y) = \left(\sum_{k=0}^{N-1} f_k \cos\left(4\pi k / N\right)\right)^2 + \left(\sum_{k=0}^{N-1} f_k \sin\left(4\pi k / N\right)\right)^2 \qquad (4.1)$$

where $f_i(i = 0,\ldots,N-1)$ are pixel values around a circle centred at (x, y).

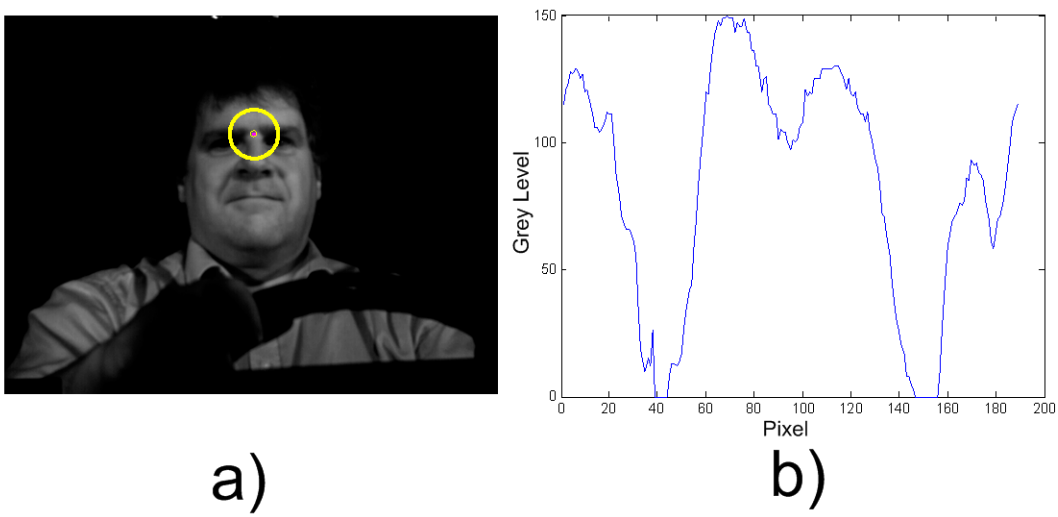
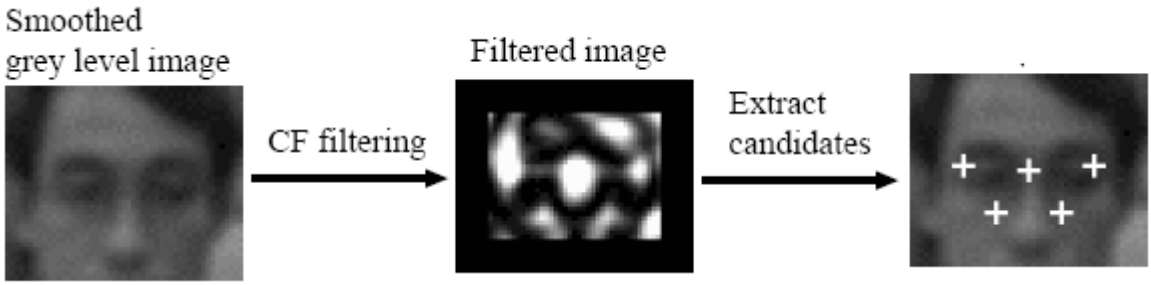


Fig. 4. 6. a) Circle with between eyes region as center. b) Grey level along the circle with between eyes region as center.

After obtaining candidates for the area between the eyes, eyes templates are matched with regions around the candidates and eyes are located where the matching score is high. The processing speed during the Circle-Frequency filtering as shown in Fig. 4.7 depends on the size of the filtering window. The smaller is the window, the faster is the filter.

Fig. 4. 7. Circle Frequency filter eyes detection overview.



Source: Kawato [40].

Other techniques such as Linear Spatial Filters commonly known as Template matching are widely used in computer vision [40]. Templates are images that contain common features for several objects of a given category, but sometimes different due to different illumination or unique object look. These approaches consist of locating the eye region roughly and matching candidate regions with templates. Candidates with a high matching score are kept.

Appendix F presents the most frequently used correlation functions. These functions measure the similarity between images and normalization helps the matching process in being independent to brightness or contrast changes [41].

Rough eye detection: This method is used as a first step and is followed by accurate methods. In colour and greyscale images, eyes are darker compared to the surrounding skin color, which makes image segmentation easy and fast to find. Some authors such as Peng et al. [46] used this method to detect the eyes roughly and later detect accurately the centre of each eye with template matching methods.

4.3 Chosen approach

The comparison presented in Table 4.1 between the different approaches reviewed above for eyes detection helps to eliminate the colour-based approaches because of the use of monochromatic cameras in the project. The neural network approach does not fit for real-time tracking. Finally, the two remaining approaches, filter based approaches and Rough eye detection were tested to choose the one with optimal performance for our application.

Table 4. 1. Eye Detection Methods.

Technique	Advantage	Drawback
Color	-Color offers more information to explore -Fast in processing time compared to other techniques	-Not applicable in greyscale images. -Difficulties in robust detection of skin colors with illumination changes
Neural Network	-Automatic learning -Simple coding -Fast if implemented carefully	-Not invariant in most head position. -Exhaustive training set -Sometimes eyes are not well detected
Filter	-Invariant to most head position -Easy to use -Fast if small search window is used	- Eyes are not detected to one pixel accuracy -Exhaustive in time if the search region is not chosen properly. .
Rough detection	-Simple implementation -Very Fast in processing time compared to other techniques	-Eyes are not detected to one pixel accuracy. -Needs extra methods to finish the detection job.

4.3.1 Vertical Frequency Filter

The test of the Circle-Frequency Filter developed by Kawato [39] on the driving video sequences did not provide good result. As the head was moving, for various subject the between eye region was not consistent. We suspect that the filter is very sensitive to the environment light setup; however, this work has been an inspiration in developing another means of filtering by taking into consideration the periodicity between dark eyebrows and eyes. There is a bright region between these two regions which allows the application of the filter (Equation 4.1) not on a circular path but rather on a vertical path as shown in Fig 4.8.

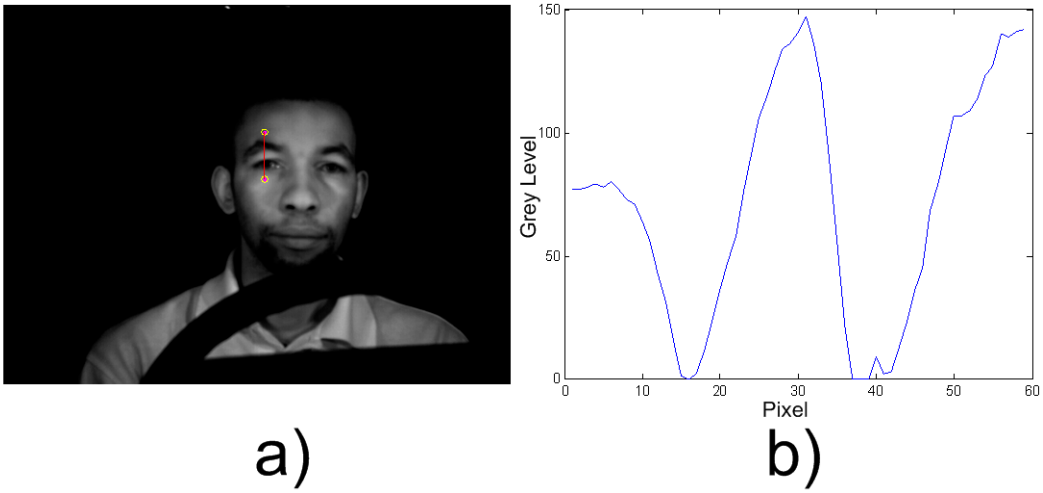


Fig. 4. 8. a) Vertical line through eye region. b) Grey level along the vertical through eye region.

The algorithm was implemented in 4 steps:

- Edge detection with the Sobel edge detector using only horizontal gradient.
- Image thresholding to get horizontal edges.
- Application of Vertical Frequency Filter.
- Image thresholding and morphological dilation to get eyes candidates.

The results such as those shown in Fig. 4.9 are very promising especially because the filter is invariant to the subject wearing glasses; however the vertical frequency filter is still under development, since all individuals do not always present significant features.

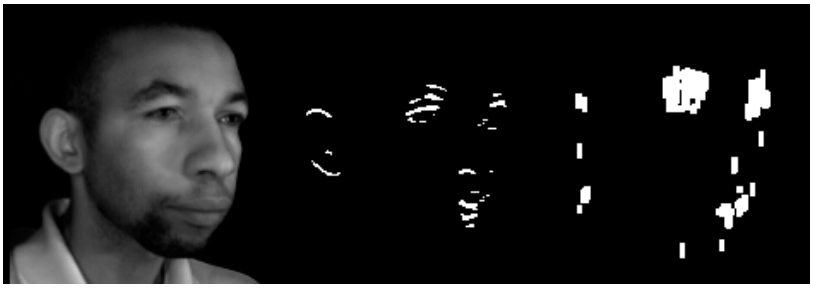


Fig. 4. 9. Vertical Frequency Filter result.

4.3.2 Rough eye detection

The rough eye detection approach was adopted for eyes detection. It has been found simple and very fast in terms of computational load. However this approach is not precise and may be corrupted by noise. An extra stage which will be discussed in the following chapter has been planned to improve results obtained at the rough eye detection stage.

The potential eye candidates are detected as follows:

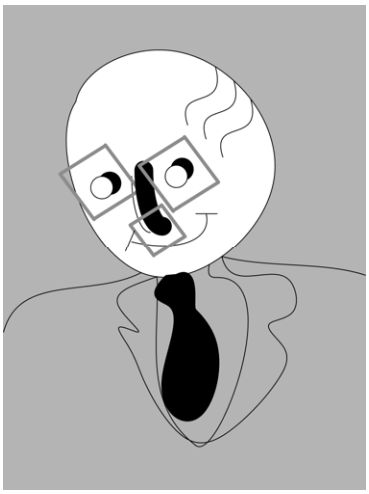
- Image histogram equalization is performed to enhance image contrast.
- Image thresholding and binarization of the equalized image.
- Image segmentation in connected regions.
- Regions discrimination according to their area size.
- Average eye pattern matching for obtained regions.

The average eye pattern was obtained by averaging several eye images with the same lighting conditions. In our case, illumination does not have a significant impact since all matching methods that were used are normalized. Finally the resulting candidates are fed to the feature extraction stage in order to complete the eyes detection.

4.4 Summary

In this chapter several approaches for eyes detection were analyzed and compared. A new method of eye detection using Vertical Frequency Filter was introduced but is reserved for future use when the tracking of subjects with additional facial features such as glasses will be needed. A rough eyes detection approach was proposed as a first stage that is performed before the algorithm for extracting facial features. This algorithm is described in the next chapter. The advantage of the rough detection approach is that it is suitable for real-time tracking.

CHAPTER 5: Facial Features Extraction



The facial features extraction stage follows the eyes detection stage described in the previous chapter. This stage aims at eliminating bad eye candidates for the eyes that may result from the rough detection stage. Regions of interest obtained at the eyes detection stage are processed to obtain real facial features that include both eyes and the tip of nose.

5.1 Introduction

In the previous chapter, an approach for rough detection of the eyes was presented. Though this approach is not very precise, a more accurate approach leading to robust eyes detection is described in this chapter. Once the eyes are detected, the tip of the nose is localized afterwards in the region under the two eyes.

This chapter is organized as follows: section 5.2 presents the literature review on the camera calibration and epipolar geometry for robust facial features extraction, section 5.3 presents the application of the camera calibration and epipolar geometry in features extraction and finally section 5.4 provides the summary.

5.2 Literature review

In the literature, the word detection is often used as a synonym for extraction [44]. Some authors, however, make a clear distinction between the two stages and suggest that feature extraction is needed when many feature candidates have been detected. There are many ways of discriminating bad candidates: Kawato [39] used a Circle-Frequency Filter to roughly detect the eye candidates and used a template matching approach as described above to extract the eyes. Other authors such as Murai [45] and Goronodnichy [30] have proposed stereo-matching as an operation to extract features. Most systems based on stereo-matching use at least two calibrated cameras (advanced techniques may use only one camera and work with non-calibrated cameras) and use epipolar geometry as a tool for matching. As our system uses several cameras, a stereo approach was used to extract the facial features robustly.

5.2.1 Camera Calibration

Camera calibration allows establishing a relationship between a 3D world coordinate and camera coordinate called standard coordinate system of the camera. A 3D point $\vec{X} = (X, Y, Z)$ is projected on the camera image plane at the point $\vec{x} = (x, y)$ as shown in Fig. 5.1.

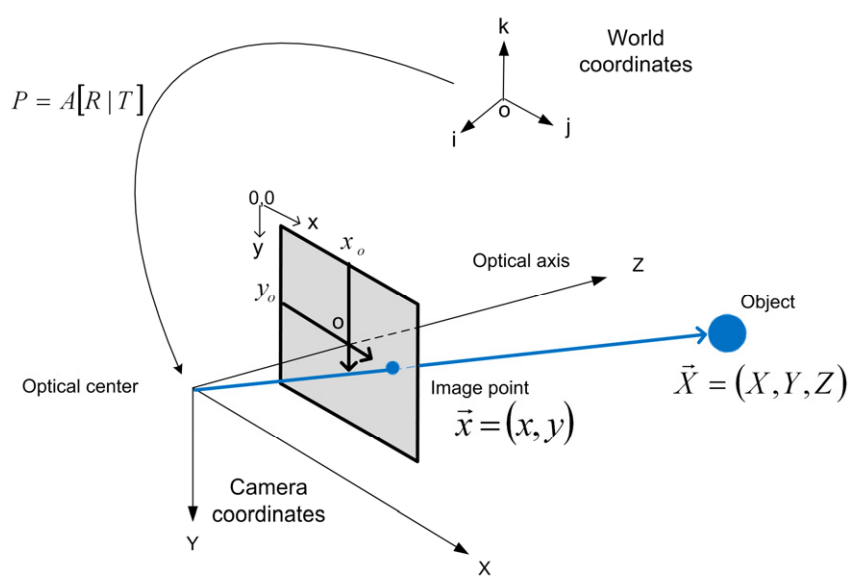


Fig. 5. 1. World and camera coordinate systems.

The relationship between the object point (X, Y, Z) and its corresponding image point (x, y) (both in orthonormal camera coordinates) is given by equations (5.1 and 5.2).

$$x = \alpha \frac{X}{Z} + x_o \tag{5.1}$$

$$y = \beta \frac{Y}{Z} + y_o \tag{5.2}$$

where $\alpha = kf$ and $\beta = lf$ are the focal length expressed in units of horizontal and vertical pixels, f is the focal length in millimetres, respectively k and l are the effective number of pixels per millimetre along the x and y axes, (x_o, y_o) are the coordinates of the principal point, given by the intersection of the optical axis with the image plane [50].

Let $\vec{X} = [X, Y, Z, 1]^T$ and $\vec{x} = [x, y, 1]^T$ be the homogenous coordinates of object point and its corresponding image, the relationship between the two is written as in (5.3).

$$s\vec{x} = A\vec{X} \Rightarrow \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} \alpha & 0 & x_o & 0 \\ 0 & \beta & y_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.3}$$

where the scaling factor s has value Z

Parameters α, β, x_o and y_o in matrix A do not depend on the position and orientation of the camera in space (e.g. the world coordinate system), and are thus called the *intrinsic* parameters.

In the above equations, it is assumed that object coordinates are expressed in the coordinate system attached to the camera but, normally, object coordinates are expressed in a world coordinate system. The new relationship is presented in equation (5.4).

$$s\vec{x} = A[R \mid T]\vec{X} \Rightarrow \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} \alpha & 0 & x_o & 0 \\ 0 & \beta & y_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.4}$$

The 12 parameters in matrices R and t define the position and orientation of the camera in space, and are thus called the *extrinsic* parameters and the matrix $P = A[R \mid T]$ is the projection matrix.

Finally the objective of geometric camera calibration is to obtain a projector equation in the world coordinates system for any object point in image plane and to predict the image coordinates of a point projected on the image plane. To obtain intrinsic and extrinsic parameters, most calibration techniques establish correspondences between 3D points on a calibration target and their 2D projections on the image plane of the camera.

5.2.2 Epipolar geometry

Epipolar geometry defines a relationship between the camera image planes of a stereo rig composed of two pinhole cameras (Fig. 5.2).

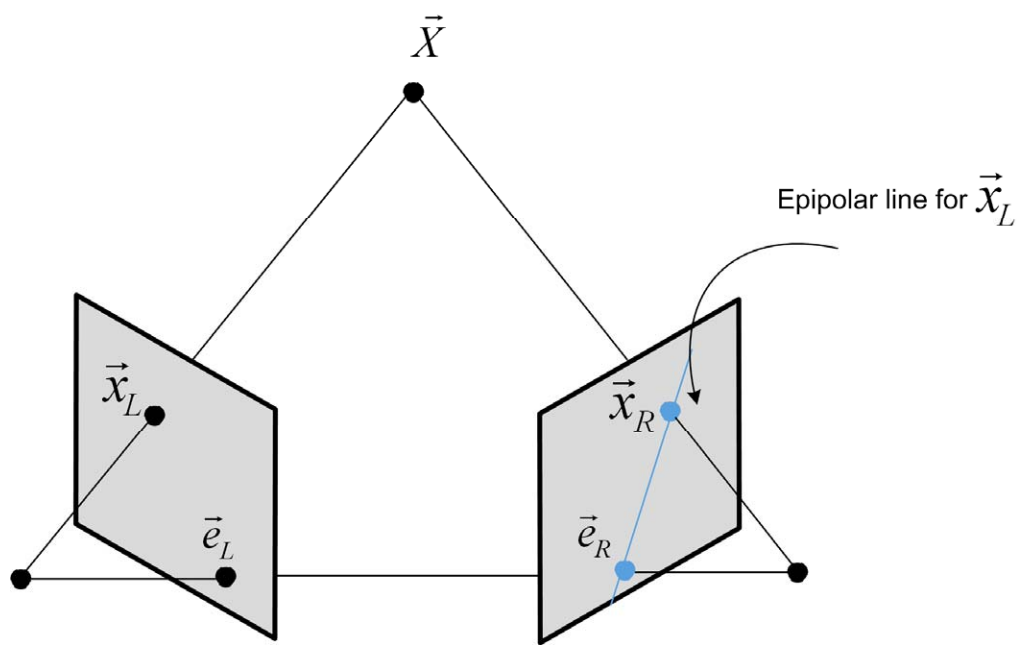


Fig. 5. 2. Epipolar line.

A 3D point $\vec{X}$ in world coordinates is projected on both images image planes, to points $\vec{x}_L$ and $\vec{x}_R$. Every point $\vec{x}_L$ in the left image plane has a conjugate point $\vec{x}_R$ in the right image plane located on a line called *epipolar line* and all epipolar lines lying in one image plane pass through a common point $\vec{e}_L$ or $\vec{e}_R$ depending on the image plane of reference. This common point is called *epipole* and is a projection of the optical centre of one camera on the image plane of the other camera [47].

To calculate epipolar lines some mathematical tools are needed. 3D points $\vec{x}_L$ and $\vec{x}_R$ in the left and right camera reference frames are related to each other respectively through the rigid transformation equation (Equation 5.5) and shown in Fig. 5.3 where *R* is the rotation matrix and *t* the translation matrix.

$$\vec{x}_L = R\vec{x}_R + T \tag{5.5}$$

The three vectors $\overrightarrow{o_L x_L}$, $\overrightarrow{o_R x_R}$ and $\overrightarrow{o_L o_R}$ lie in the same plane. It is well known that any vector resulting from the vector product of two set of vectors is geometrically perpendicular to the planed spanned by these two vectors. Furthermore the scalar product between any perpendicular vectors is zero. Therefore the vector $\left[\overrightarrow{o_L o_R} \otimes \overrightarrow{o_R x_R}\right]$ is perpendicular to the vector $\overrightarrow{o_L x_L}$ as presented in equation (5.6).

$$\overrightarrow{o_L x_L} \bullet \left[\overrightarrow{o_L o_R} \otimes \overrightarrow{o_R x_R}\right] = 0 \tag{5.6}$$

We can write this equation in the coordinate system associated to the first camera as

$$\vec{x}_L \bullet \left(T \otimes R\vec{x}_R\right) = 0 \tag{5.7}$$

The development of equation (5.7) in matrix form is presented as.

$$\begin{cases} \vec{x}_L \bullet (T \otimes R\vec{x}_R) = 0 \\ \Leftrightarrow \vec{x}_L{}^T [T]_x R\vec{x}_R = 0 \\ \Leftrightarrow \vec{x}_L{}^T E\vec{x}_R \quad = 0 \end{cases} \tag{5.8}$$

where $[T]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$, $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ and

$E = \begin{bmatrix} -t_z r_{21} + t_y r_{31} & -t_z r_{22} + t_y r_{32} & -t_z r_{23} + t_y r_{33} \\ t_z r_{11} - t_x r_{31} & t_z r_{12} - t_x r_{32} & t_z r_{13} - t_x r_{33} \\ -t_y r_{11} + t_x r_{21} & -t_y r_{12} + t_x r_{22} & -t_y r_{13} + t_x r_{23} \end{bmatrix}$ is called the *essential matrix.*

The essential matrix maps a 3D point defined in the first camera world reference of the stereo rig in the second camera world reference through an epipolar line. $\vec{x}_L$ belongs to the epipolar line $E\vec{x}_R$ and $\vec{x}_R$ belongs to the epipolar line $E^T\vec{x}_L$.

Another matrix called *fundamental matrix* maps a 2D point defined in the first camera reference of a stereo rig into the second camera reference through an epipolar line as described in equation (5.9).

$$\begin{cases} \vec{x}_L{}^T A_L{}^{-T} E A_R{}^{-1} \vec{x}_R = 0 \\ \Leftrightarrow \vec{x}_L{}^T F\vec{x}_R = 0 \end{cases} \tag{5.9}$$

where A_L and A_R are respectively the intrinsic parameters matrices for the left and right cameras and F is the fundamental matrix.

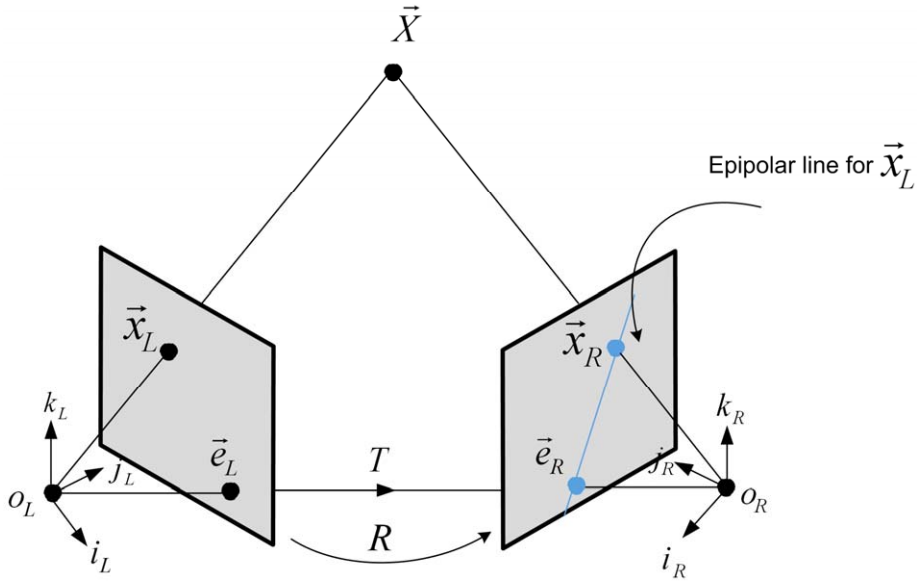


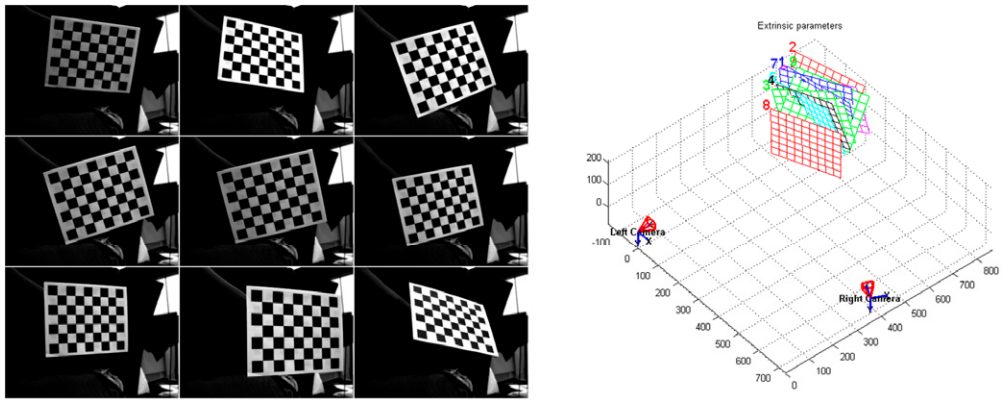
Fig. 5. 3. Rigid motion transformation between two cameras.

5.3 Facial features extraction discussion

5.3.1 Eyes extraction

When two stereo images of a driver are observed, only one eye can be easily detected by applying the image processing techniques described in chapter 4. Most of the time, the other eye is in contact with the background which makes segmentation difficult. To overcome the segmentation problem, a pair of camera was calibrated geometrically using the camera calibration toolbox developed by Bouguet [48] in MATLAB (Fig.5.4). After calibration, the epipolar geometry of the stereo pair described above can be exploited. Before using epipolar geometry, the rough eye detection approach was used to isolate eye regions that do not touch the background as shown is Fig. 5.5. For eye regions touching the background, an eye template built by averaging several human eyes regions was matched with image regions along the epipolar line from the bottom to the top until an image region that looks like a peninsula[1] is detected. The matching score is high in this region because it is the location of the eye region that touches the background.

1. Peninsula: A piece of land almost completely surrounded by water.



a) b)

Fig. 5. 4. Calibration results.

The extraction procedure described above can be summarized in five steps:

A corresponding epipolar line for every eye candidate in both images is obtained.

- A candidate that matches a candidate in another image on the corresponding epipolar line is detected by using pattern matching, the template being an eye average image.
- A test is performed to check if the found match is in the candidates set of the second image.
- A test is performed to make sure that the candidate is not outside of the head bounding boxes in both images.
- A validation is done on the results with a neural network using the distance information between the two eyes in every image (see next section).

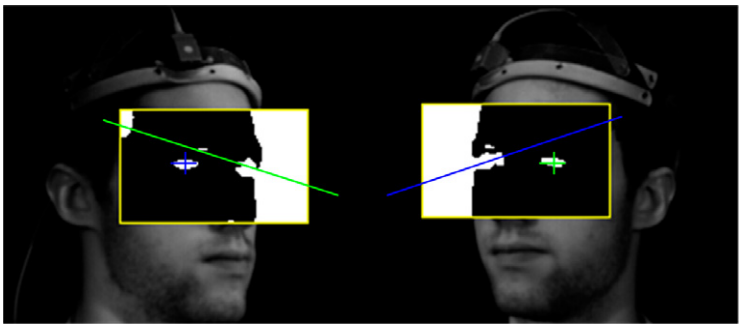


Fig. 5. 5. Epipolar geometry in use.

5.3.1 Eyes extraction validation with a neural network

The chosen algorithms so far, combined with the epipolar geometry, help to track the eyes in 2D plane with a detection rate up to 90% whenever both eyes are visible in left and right video frame sequences of this project. Though this detection rate is very high the remaining 10% of bad detection is enough to distort the 3D reconstruction discussed in the next chapter. To solve this problem, a multilayer perceptron described in Appendix G was used to improve detection achieved at previous stages and to validate the good detections.

For all adult drivers sitting in the same car seat, head motion shares the same 3D space bounded by a virtual cube as shown in Fig. 5.6. As all adults have relatively the same distance between their eyes, it is easy to build vectors in the left image plane that have correspondences in the right image plane.

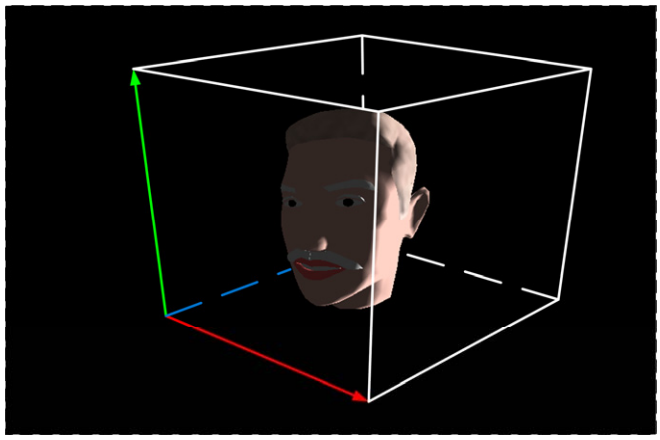


Fig. 5. 6. Driver's head bounding cube.

Training data were constructed by using four vector modulus in each camera frame as shown in Fig. 5.7, and the output 1 or 0 was assigned as a neural output to validate the detection result.

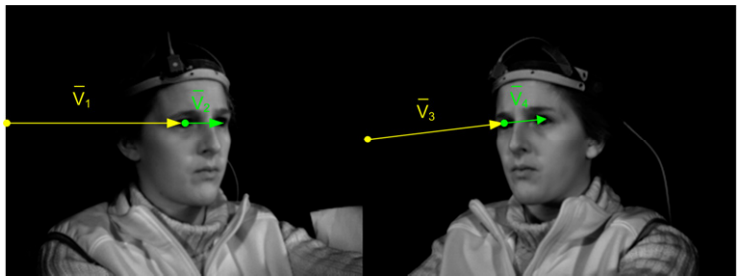


Fig. 5. 7. Epipolar geometry in use.

5.3.1 Nose tip extraction

The nose has been presented by Goronodnichy [30] as the best facial feature to be tracked. He concluded that the human nose was the only facial feature clearly visible during head motions hence the interest of implementing a robust nose detection approach. Researchers use different techniques to solve the problem, the most representative include template matching approaches and approaches based on nose photometric properties.

Template matching approach: Goronodnichy showed that the intensity profile around the tip of the nose stays the same when the head rotates. He has proposed to track the tip of

nose by using a technique used in pattern recognition and machine learning called feature vector. A feature vector is an n-dimensional vector of numerical features that represent an object [44]. With the feature vector, a template is not only made of pixels but also contains other properties such as geometric properties of the feature. The algorithm proposed by Goronodnichy also relies on tracking results obtained in previous frames, before processing new frames for tracking.

Nose photometric properties approach: This approach relies on the work of Gurbuz and Kawato [42]. It assumes that eyes have been detected correctly and that the tip of the nose is located in the highest intensity region within a search region estimated from the two eye positions.

After successfully extracting two eyes from each image plane, the approach of nose extraction by using photometric properties can be implemented in simple steps:

This technique has been chosen for the current use and has been implemented in the following steps:

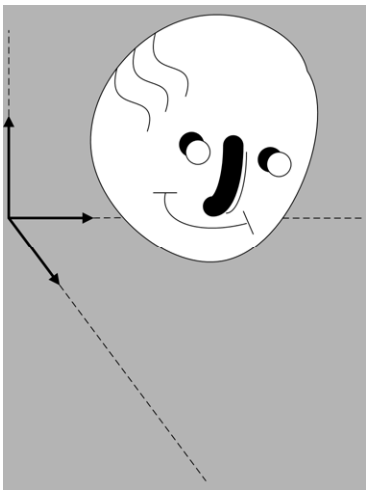
- Nose region histogram equalization.
- Edge detection with Sobel edge detector using only horizontal gradient.
- Image thresholding and morphological dilation to get connected horizontal edges.
- Brightest point detection in the connected horizontal edge regions.

After the detection of all facial features, the 3D reconstruction stage discussed in the next chapter can be implemented.

5.4 Summary

In this chapter we applied mathematical tools such as geometrical camera calibration and epipolar geometry to extract eyes and the tip of the nose from the image sequences. A neural network that led to a robust facial features detection and extraction system validated this extraction.

CHAPTER 6 : 3D Reconstruction



The 3D reconstruction is the stage at which the position of a real 3D point observed by the stereo pair is recovered.

6.1 Introduction

The reconstruction of the head pose in 3D space is the main objective of this thesis. This reconstruction is possible when the geometric relationship between cameras in a stereo rig is known and when the projection of the same point on the head can be found in the left and right images. The geometric relationship between cameras was described in the previous chapter when the topic of camera calibration was discussed. As all facial features (eyes, tip of the nose) have been detected in both left and right images, the 3D reconstruction of a plane associated with the triangle formed by the features can be found by implementing a good 3D reconstruction technique.

This chapter is organized as follows: section 6.2 presents the Literature review on different techniques used for 3D reconstruction, section 6.3 describes the head pose computation and finally, section 6.4 provides the summary.

6.2 Literature review

Modeling a 3D scene from several images is known as 3D reconstruction. This field has received much interest in the computer vision community. As the cost of computers is dropping, many applications using 3D reconstruction are emerging. The number of publications in 3D computer vision, 3D medical imaging, satellite imaging, etc proves this.

The problem of 3D reconstruction or triangulation problem is formulated as follows; A 3D point $\vec{X}$ is projected at point $\vec{x}_L$ and $\vec{x}_R$ in corresponding left and right image planes of a stereo pair. Suppose P_L and P_R are corresponding projections matrices used in the projection, two rays can be reconstructed from the corresponding projected points $\vec{x}_L$ and $\vec{x}_R$ to the initial 3D point $\vec{X}$. In theory, the two lines should intersect perfectly in space but due to noise and other errors it is not the case. The following sections describe different approaches that have been proposed for 3D reconstruction.

6.2.1 Mid-point triangulation

This is the most intuitive method for 3D reconstruction but it is not optimal. Two rays corresponding to the projected points in both left and right image planes do not intersect at the original 3D point from which the images originated. There is always an error due to various factors such as digitizing error, sensor noise, matching errors, etc. To minimize the error, the 3D point is chosen to be located in halfway on a vector that is perpendicular to both rays as shown in Fig. 6.1.

In the previous chapter it has been shown that the point $\vec{X}$ in world coordinate system can be decomposed in two points $\vec{X}_L = [X_L \quad Y_L \quad Z_L]$ and $\vec{X}_R = [X_R \quad Y_R \quad Z_R]$ in corresponding left and right camera system coordinates, and these two points are related by the rigid transformation equation (6.1)

$$\vec{X}_L = R\vec{X}_R + T \qquad (6.1)$$

where R and T are respectively the rotation matrix and translation matrix of the frame transformation between the scene reference frame and the camera reference frame.

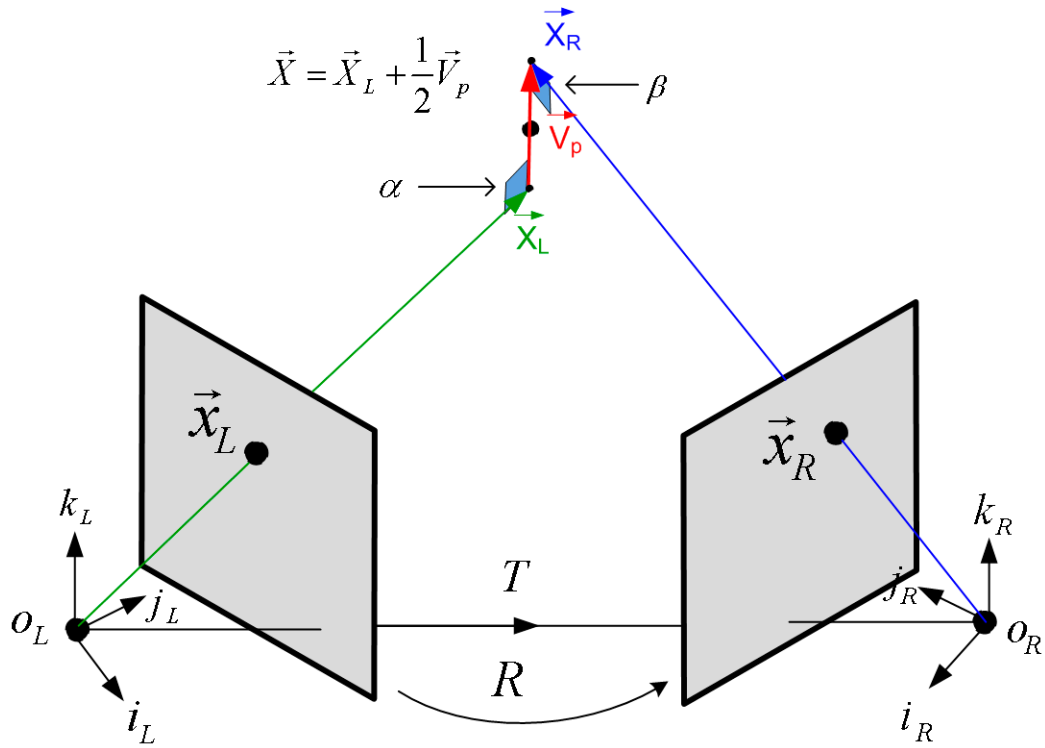


Fig. 6. 1. Mid-point triangulation.

Let us consider the left camera coordinate system as the reference. $\vec{X}_L$ and $\vec{X}_R$ are respectively mapped on image planes at points $\vec{x}_L = [x_L \quad y_L \quad 1]$ and $\vec{x}_R = [x_R \quad y_R \quad 1]$ in homogeneous form through equations (6.2).

$$\begin{cases} \vec{X}_L = a\vec{x}_L \\ \vec{X}_R = R^{-1}(b\vec{x}_R - T) \end{cases} \tag{6.2}$$

where a and b are real numbers.

A perpendicular vector $\vec{V}_p$ between $\vec{X}_L$ and $\vec{X}_R$ is given by the vector product of the two ray vectors as shown in equation (6.3).

$$\vec{V}_p = c\left(\vec{x}_L \otimes \left(R^{-1}\vec{x}_R - T\right) - T\right) \tag{6.3}$$

where c is a real number.

To find the 3 real numbers a, b and c, the equation to be solved is given in (6.4).

$$\begin{aligned}
&\vec{X}_L + \vec{V}_p = \vec{X}_R \\
&\Leftrightarrow (a\vec{x}_L) + c\left(\vec{x}_L \otimes \left(R^{-1}\vec{x}_R - T\right) - T\right) = R^{-1}\left(b\vec{x}_R - T\right) \\
&\Leftrightarrow (a\vec{x}_L) - \left(bR^{-1}\vec{x}_R\right) + c\left(\vec{x}_L \otimes \left(R^{-1}\vec{x}_R - T\right) - T\right) = -R^{-1}T
\end{aligned} \tag{6.4}$$

Finally the mid-point is computed with

$$\vec{X} = \vec{X}_L + \frac{1}{2}\vec{V}_p \tag{6.5}$$

As mentioned earlier, the use of the mid-point technique is not optimal according to Hartley [53]. The mid-point of the perpendicular vector does not provide any guarantee on the equality between two angles α and β as shown in Fig. 6.1. This angle equality is desirable for optimal triangulation.

6.2.2 Linear triangulation

This is the most common method for 3D reconstruction; the algorithm presented below has been implemented by Bouguet in his *Camera Calibration Toolbox* for MATLAB.

In the mid-point triangulation presented in the previous section, it has been shown that point $\vec{X}$ in the world coordinate system can be expressed in two reference frames $\vec{X}_L = \begin{bmatrix} X_L & Y_L & Z_L \end{bmatrix}$ and $\vec{X}_R = \begin{bmatrix} X_R & Y_R & Z_R \end{bmatrix}$ (in corresponding left and right camera system coordinates), these two points are being related by the rigid transformation equation (6.1).

Let $\vec{x}_{L,R} = \vec{X}_{L,R} / Z_{L,R} = \begin{bmatrix} x_{L,R} & y_{L,R} & 1 \end{bmatrix}$ be the coordinate vectors resulting from the perspective projections using the left and right projective matrices P_L and P_R mentioned in section 6.2.

Equation (6.1) can now be written in closed-form as in (6.6) and in matrix form as in (6.8).

$$\vec{x}_L Z_L = R\left(\vec{x}_R Z_R\right) + T \tag{6.6}$$

$$\Leftrightarrow -R\vec{x}_R Z_R + \vec{x}_L Z_L = T \qquad (6.7)$$

$$\Leftrightarrow \begin{bmatrix} -R\vec{x}_R & \vec{x}_L \end{bmatrix} \begin{bmatrix} Z_R \\ Z_L \end{bmatrix} = T \qquad (6.8)$$

In this case the triangulation consists of obtaining $\vec{X}_L$ and $\vec{X}_R$ from $\vec{x}_L$ and $\vec{x}_R$ as shown in Fig. 6.2.

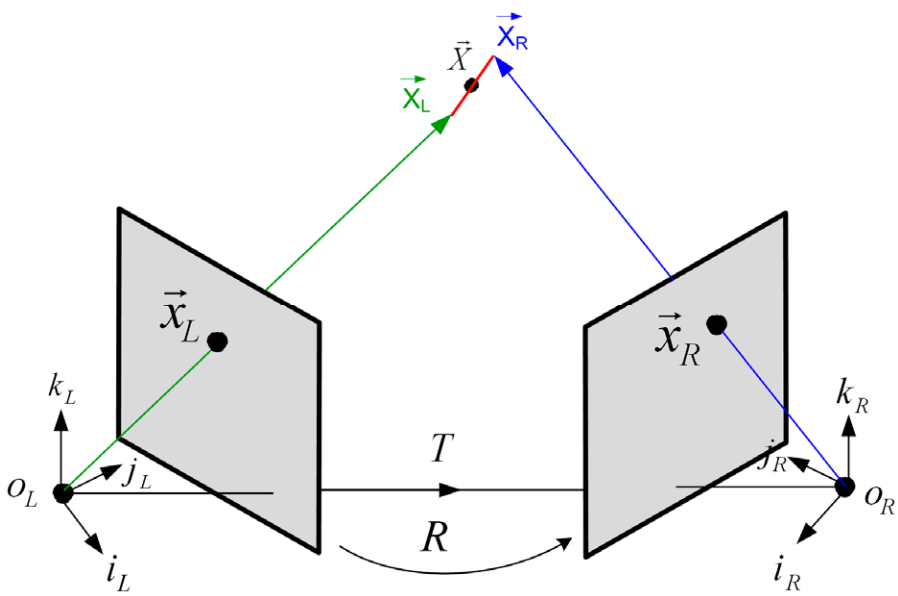


Fig. 6. 2. Linear triangulation.

On way of solving the three equations above (6.8) in two unknowns (Z_L and Z_R) is through a least-squares solution. Let $A = \begin{bmatrix} -R\vec{x}_R & \vec{x}_L \end{bmatrix}$ be a 3 x 2 matrix, the solution is presented in equation (6.9) by using a pseudo-inverse $\left(A^T A\right)^{-1} A^T$.

$$\begin{bmatrix} Z_R \\ Z_L \end{bmatrix} = \left(A^T A\right)^{-1} A^T T \tag{6.9}$$

$$\Leftrightarrow \begin{bmatrix} Z_R \\ Z_L \end{bmatrix} = \left(\begin{bmatrix} -R\vec{x}_R \\ \vec{x}_L \end{bmatrix} \begin{bmatrix} -R\vec{x}_R & \vec{x}_L \end{bmatrix} \right)^{-1} \begin{bmatrix} -R\vec{x}_R \\ \vec{x}_L \end{bmatrix} T \tag{6.10}$$

$$\Leftrightarrow \begin{bmatrix} Z_R \\ Z_L \end{bmatrix} = \left(\begin{bmatrix} \left\| -R\vec{x}_R \right\|^2 & \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle \\ \left\langle \vec{x}_L, -R\vec{x}_R \right\rangle & \left\| \vec{x}_L \right\|^2 \end{bmatrix} \right)^{-1} \begin{bmatrix} \left\langle -R\vec{x}_R, T \right\rangle \\ \left\langle T, \vec{x}_L \right\rangle \end{bmatrix} \tag{6.11}$$

$$\Leftrightarrow \begin{bmatrix} Z_R \\ Z_L \end{bmatrix} = \frac{1}{\left\| -R\vec{x}_R \right\|^2 \left\| \vec{x}_L \right\|^2 - \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle^2} \begin{bmatrix} \left\| \vec{x}_L \right\|^2 & -\left\langle -R\vec{x}_R, \vec{x}_L \right\rangle \\ -\left\langle \vec{x}_L, -R\vec{x}_R \right\rangle & \left\| -R\vec{x}_R \right\|^2 \end{bmatrix} \begin{bmatrix} \left\langle -R\vec{x}_R, T \right\rangle \\ \left\langle T, \vec{x}_L \right\rangle \end{bmatrix} \tag{6.12}$$

$$\Leftrightarrow \begin{bmatrix} Z_R \\ Z_L \end{bmatrix} = \begin{bmatrix} \dfrac{\left\| \vec{x}_L \right\|^2 \left\langle -R\vec{x}_R, T \right\rangle - \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle \left\langle T, \vec{x}_L \right\rangle}{\left\| -R\vec{x}_R \right\|^2 \left\| \vec{x}_L \right\|^2 - \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle^2} \\ \dfrac{-\left\langle \vec{x}_L, -R\vec{x}_R \right\rangle \left\langle -R\vec{x}_R, T \right\rangle + \left\| -R\vec{x}_R \right\|^2 \left\langle T, \vec{x}_L \right\rangle}{\left\| -R\vec{x}_R \right\|^2 \left\| \vec{x}_L \right\|^2 - \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle^2} \end{bmatrix} \tag{6.13}$$

The expressions of Z_L and Z_R are presented in equations (6.14 and 6.15)

$$Z_L = \frac{\left\| -R\vec{x}_R \right\|^2 \left\langle T, \vec{x}_L \right\rangle - \left\langle \vec{x}_L, -R\vec{x}_R \right\rangle \left\langle -R\vec{x}_R, T \right\rangle}{\left\| -R\vec{x}_R \right\|^2 \left\| \vec{x}_L \right\|^2 - \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle^2} \tag{6.14}$$

$$Z_R = \frac{\left\| \vec{x}_L \right\|^2 \left\langle -R\vec{x}_R, T \right\rangle - \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle \left\langle T, \vec{x}_L \right\rangle}{\left\| -R\vec{x}_R \right\|^2 \left\| \vec{x}_L \right\|^2 - \left\langle -R\vec{x}_R, \vec{x}_L \right\rangle^2} \tag{6.15}$$

where $\langle ..., ... \rangle$ is a the dot product operator[1]. Finally, the point $\vec{X}$ in the world coordinate system is obtained by considering one of the two camera coordinate systems as a reference and using the results above for retrieving $\vec{X}_L$ and $\vec{X}_R$.

---

1. $\langle (x_1, ..., x_n), (y_1, ..., y_n) \rangle := \sum_{i=1}^{n} x_i y_i = x_1 y_1 + ... + x_n y_n$

6.2.3 Bundle adjustment

Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structures and viewing parameters (camera pose and/or calibration) estimates [51]. This method searches for the optimal 3D point and the optimal projection matrices simultaneously, by minimizing the distance between observed points in the image planes and projected 3D scene points from the projection matrices as shown in equation (6.16).

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} D\left(\vec{x}_{ij}, P_i \vec{X}_j\right) \qquad 6.16$$

where $\vec{x}$ is the observed image point, P the unknown projection matrix, $\vec{X}$ the 3D reconstructed point and D the Euclidian distance. To resolve this problem, many alternative optimization algorithms such as Levenberg-Marquardt's Algorithm or Dog Leg Algorithm can be used [53].

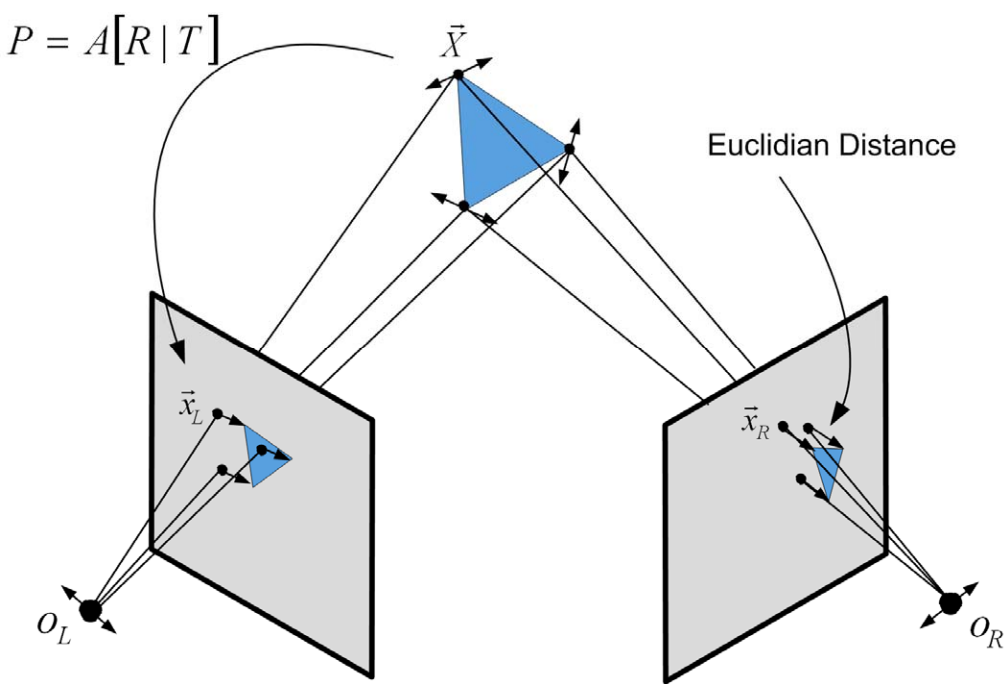


Fig. 6. 3. Bundle adjustment technique.

6.3 Head pose generation

Given the triangulation techniques used to reconstruct a 3D point from a pair of image points obtained from the respective left and right image planes of our stereo rig, it is possible to choose one technique that meets the need of a real-time and robust system.

The comparison of the three techniques as presented in sections 6.2.1 to 6.2.3 and which is summarized in Table 6.1, reveals that bundle adjustment may not be fast enough due to the use of iterative algorithms. The mid-point and linear triangulation approaches are not optimal but are sufficient for our application. Due to the critics done by Hartley [53] on the mid-point approach, linear triangulation was chosen as the best technique for our system.

Table 6. 1. 3D reconstruction methods.

Technique	Advantage	Drawback
Mid-point triangulation	-Intuitive. -Simple coding.	-Too many approximations so that it does not give optimal result. -Needs good projection matrices in order to provide better results.
Linear triangulation	-The most common triangulation method. -Simple coding.	-Is not the most optimal either. -Needs good projection matrices in order to provide better results.
Bundle adjustment	-Corrects the projection matrices error. -Does not need very good projection matrices.	-Uses iterative techniques, with a high computational load. -Needs more than two views to work optimally.

After obtaining 3D points for the tip of nose, the left and right eyes using linear triangulation, a 3D OPENGL model of the head shown in Fig. 6.4, was used to show the orientation of the plane formed by two eyes and nose in space.

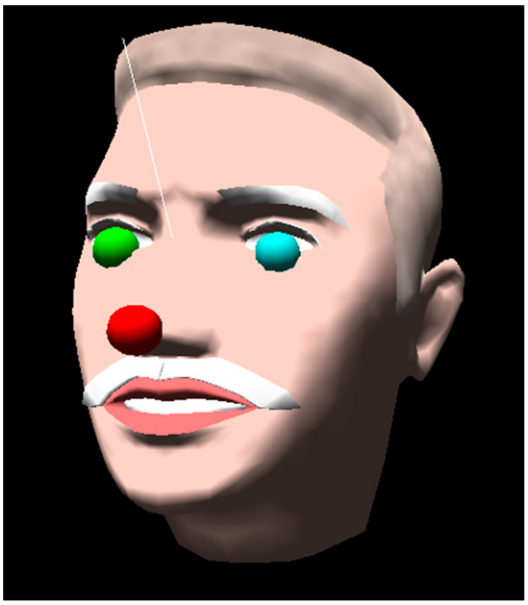


Fig. 6. 4. 3D model fitting.

There are two desirable features for the generated 3D points along successive frames: the first is that the location of facial features obtained in one frame should help to locate the same facial features in successive frames. The second is that the transition from one point to another should be smooth.

The first desirable feature was addressed by optimizing the system in order to avoid the search for facial features in each frame. The system runs in two modes. In the first mode, called “detection mode”, the head and all three facial features are detected as discussed in previous chapters. Then, once the first pose estimates become available, the second mode is initiated. In this mode the head detection stage is skipped by only using previous location of the eyes in both left and right frames to locate the new facial features position. This strategy is shown in Fig. 6.5.

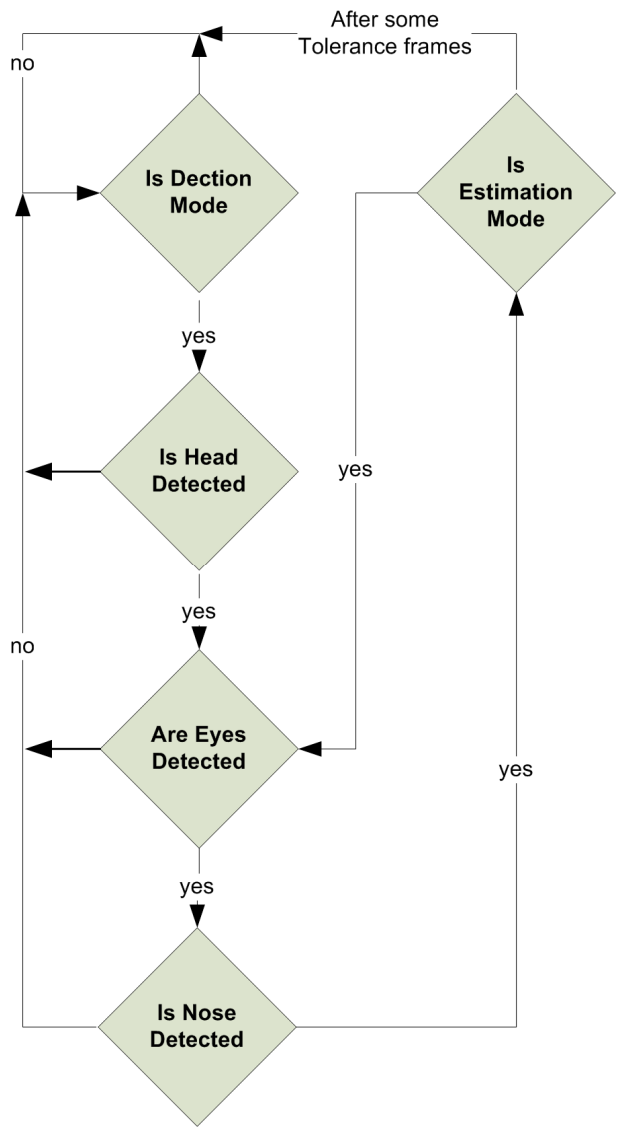


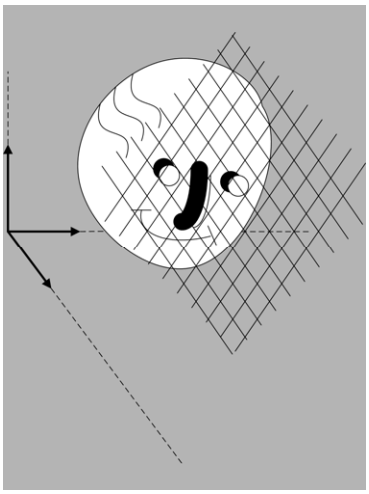
Fig. 6. 5. System mode cycle.

The second desirable feature of smooth and fluent transition between successive frames is discussed in Appendix H where a Kalman filter capable of estimating a good 3D point while minimizing the effect of the noise is discussed.

6.4 Summary

In this chapter after comparing three approaches for 3D reconstruction by triangulation, a linear triangulation based approach was used to get the position of the facial features in 3D space. The Kalman filter was also used to smooth the transition of 3D points from successive video frames. All the choices made in this chapter are suitable for real-time tracking.

CHAPTER 7 : Experimental Results



System performance consists of obtaining experimental results using the implemented system. At this stage, experimental results are evaluated in order to see how successful the system is, and how it can be compared to similar systems.

7.1 Introduction

The evaluation of the vision system that is being discussed in this thesis was carried out by creating a video database that meets the project needs. In the database that is still being populated, we chose 20 video sequences captured by our stereo rig. The database is very representative as shown in Fig. 7.1. It includes women, men, young, old, white and one black person.

The system performance was evaluated stage by stage: In both head detection and facial features detection stages the evaluation is based on the percentage of correct detection rate (CDR) and false detection rate (FDR) when system parameters are changing. The CDR is the percentage of frames when the detection is performed as planned, and the FDR is the percentage of frames when the detection occurs in the wrong place. Both CDR and FDR are exclusive in our system.

Fig. 7. 1. Video Sequences Database, the number of frames in each video sequence is respectively: a) 366 b) 653 c) 380 d) 492 e) 645 f) 172 g) 735 h) 441) i) 491 j) 2005.

At the 3D reconstruction stage, measurements along the six axes of motion known as degrees of freedom of the head (longitudinal, vertical and lateral, roll, pitch and yaw) as presented in Appendix I and Fig. 7.2, are obtained and correlated with 3D measurements provided by a motion tracking device installed on the driver's head.

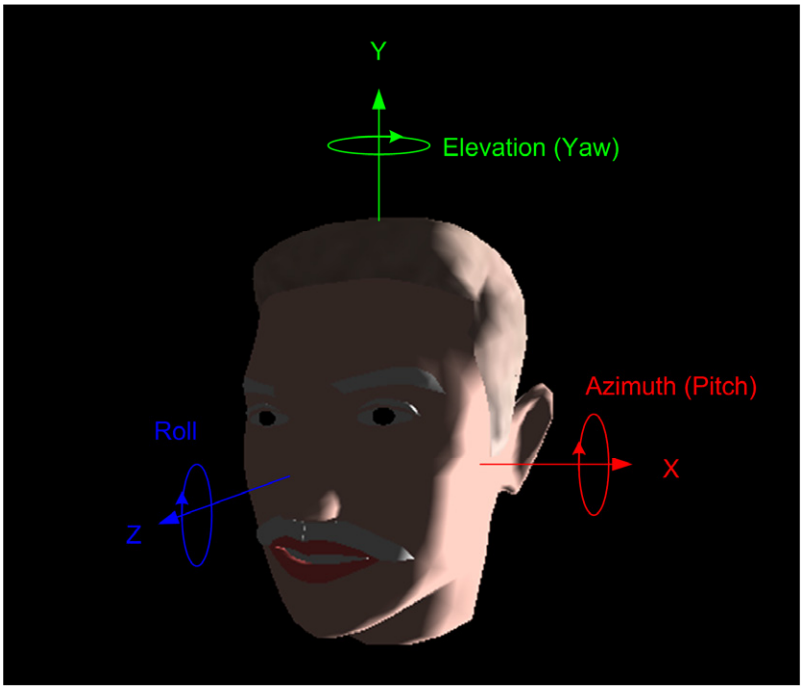


Fig. 7. 2. Degrees of freedom for the head.

This chapter is organized as follows: section 7.2 presents the Head detection performance, section 7.3 describes the Facial features detection performance, section 7.4 presents the 3D reconstruction performance and finally, section 7.5 provides a summary of the results.

7.2 Head detection performance

The head detection was the starting point of our algorithm for tracking the head in order to obtain the 3D pose. As our system stages are sequential, every stage is important and has to achieve a good performance otherwise it will affect the performance of following stages. A stage has a good performance if it has a high percentage of correct detection rate and a low percentage of false detection rate. Our head detection approach has a good performance as shown in Fig.7.3. It has an average correct detection rate of 99.91% and an average false detection rate of 0.09%.

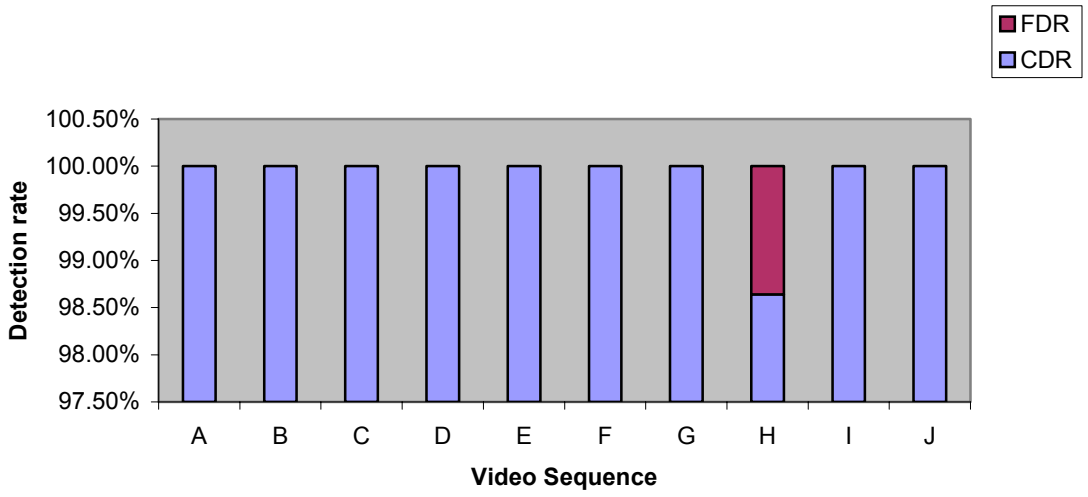


Fig. 7. 3. Head detection performance.

7.3 Facial features detection performance

The algorithm used to detect facial features must maximize the percentage of correct detection rate and minimize the percentage of false detection. As this algorithm works in two stages as described in previous chapters and needs proper setting of various parameters to work, system performance is evaluated by changing only one parameter at a time. These parameters include parameter "*tolerance*" needed when results from previous frames are used in the remaining frames to choose a search region that is smaller than the head region.

The parameter "*high in*" is used for eyes region histogram equalization, the threshold parameter is used after eyes region histogram equalization is performed, parameter "*high in*" is used in nose region histogram equalization, finally two threshold parameters are used after normalized cross correlation and normalized coefficient correlation are performed as implemented in OPENCV.

7.3.1 Effect of "Tolerance" parameter

This parameter is used in the algorithm described in chapter 6. The position of detected facial features in previous video frames is used to build a more restrained region where to detect facial features in following frames, instead of detecting the whole driver's head. This parameter consists of the number of frames during which estimation mode operates before changing to detection mode.

By observing Fig. 7.4 and 7.5, we can conclude that, as the tolerance parameter increases, the detection rate drops slightly while the frame rate increases, which is the desired effect.

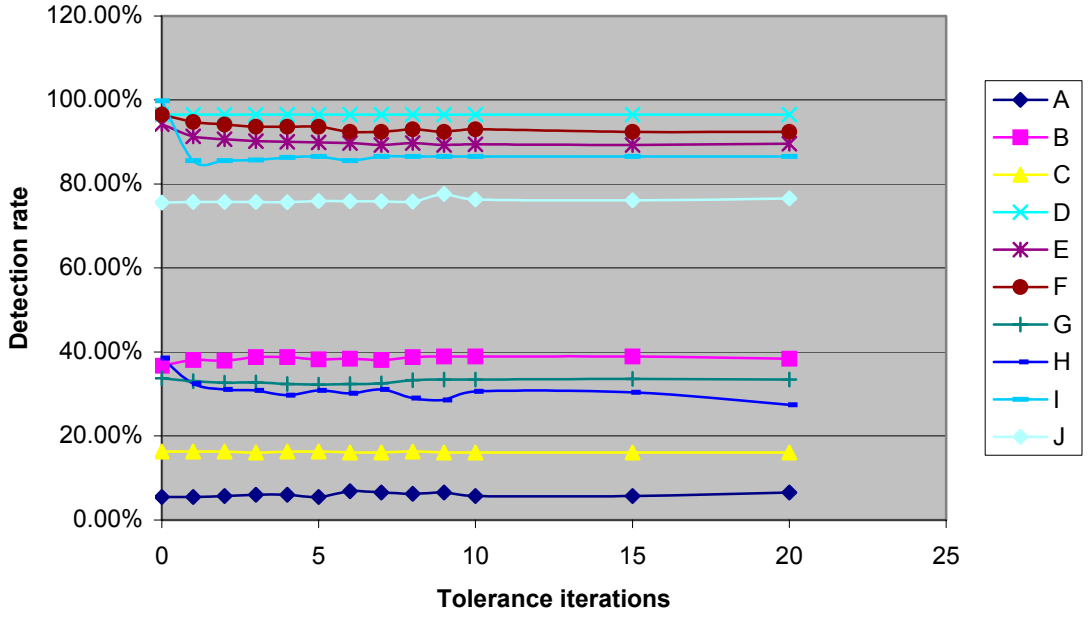


Fig. 7. 4. Effect of tolerance on detection rate.

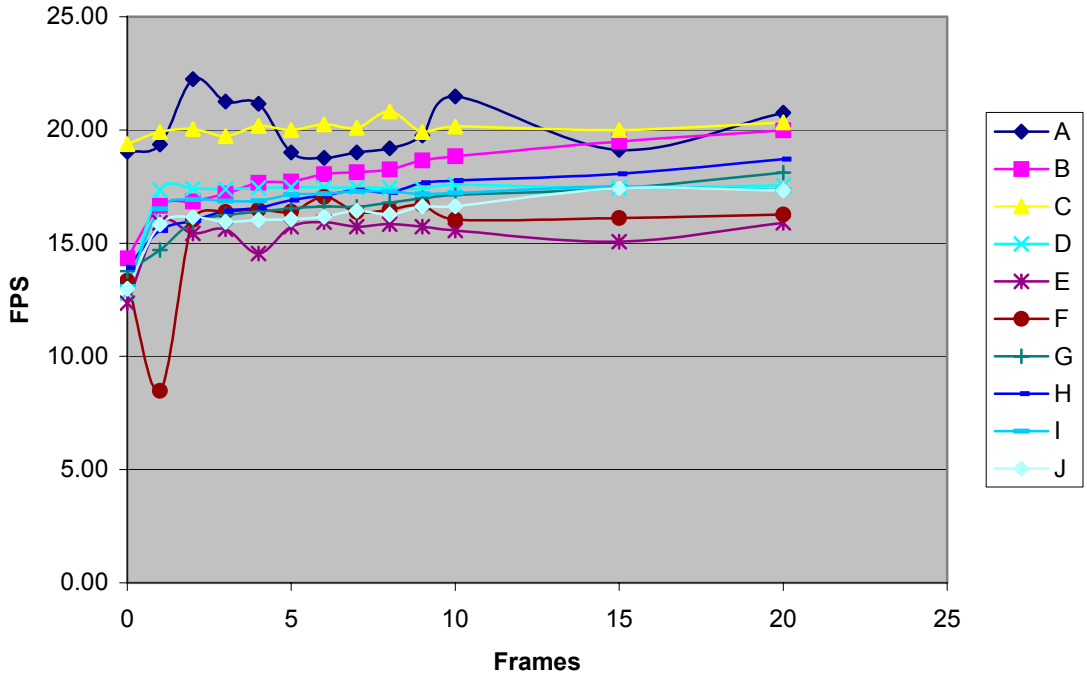


Fig. 7. 5. Effect of tolerance on the frame rate.

In Fig. 7.4 above, we notice a poor performance in detection rate of video sequences A and C and a very good frame rate performance for the same sequences. This is explained by the fact that subjects were wearing glasses. The detection of eye features for subject wearing glasses is not yet implemented. The frame rate is high because when facial features are not detected, the computation load is low.

7.3.2 Effect of "high in" parameter in eyes region histogram equalization

The algorithm used for histogram equalization is a clone of the function "*imadjust*" used in a numerical computing environment and programming language called MATLAB as described in equation (7.1).

$$J = imadjust(I, [low\_in \;\; high\_in], [low\_out \;\; high\_out], gamma) \qquad (7.1)$$

This function maps the values in image I to new values in image J such that values between parameters low_in and $high\_in$ map to values between parameters

low_out and $high\_out$. Parameter $gamma$ specifies the shape of the curve describing the relationship between the values in I and J. By default low_out and $high\_out$ are set to 0 and 1 respectively, to cover the whole greyscale range, low_in is set to 0 and $gamma$ to 1 to allow linear mapping, the only variable is $high\_in$.

By observing carefully the detection rate according to the change of the parameter $high\_in$ as plotted in Fig. 7.6, we cannot infer any tendency; still it has been observed empirically the best range of the parameter $high\_in$ is between 0.2 and 0.8 for most video sequences.

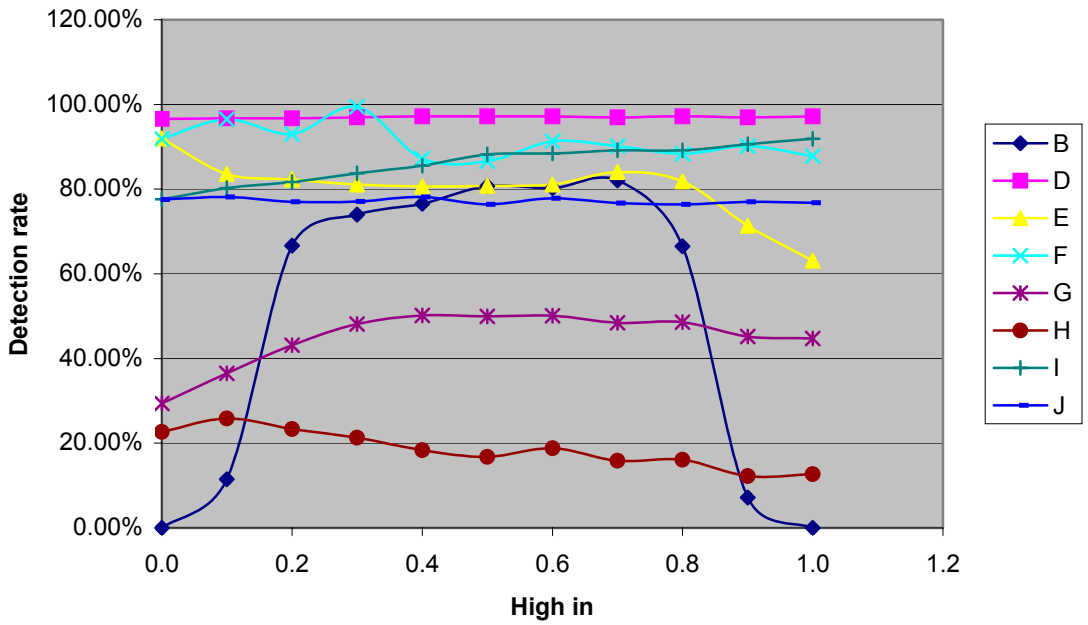


Fig. 7. 6. Effect of "High in" parameter on the detection rate.

7.3.3 Effect of threshold parameter after eye region histogram equalization

The threshold parameter is used to roughly detect eyes features as described in chapter 5. During the thresholding operation, a binary image is obtained from a greyscale image by using the function *cvThreshold* from OPENCV. This binary image is a pre-processing phase to obtain features blobs. The plots in Fig 7.7 show that values lower than 200 are the best choice for thresholding.

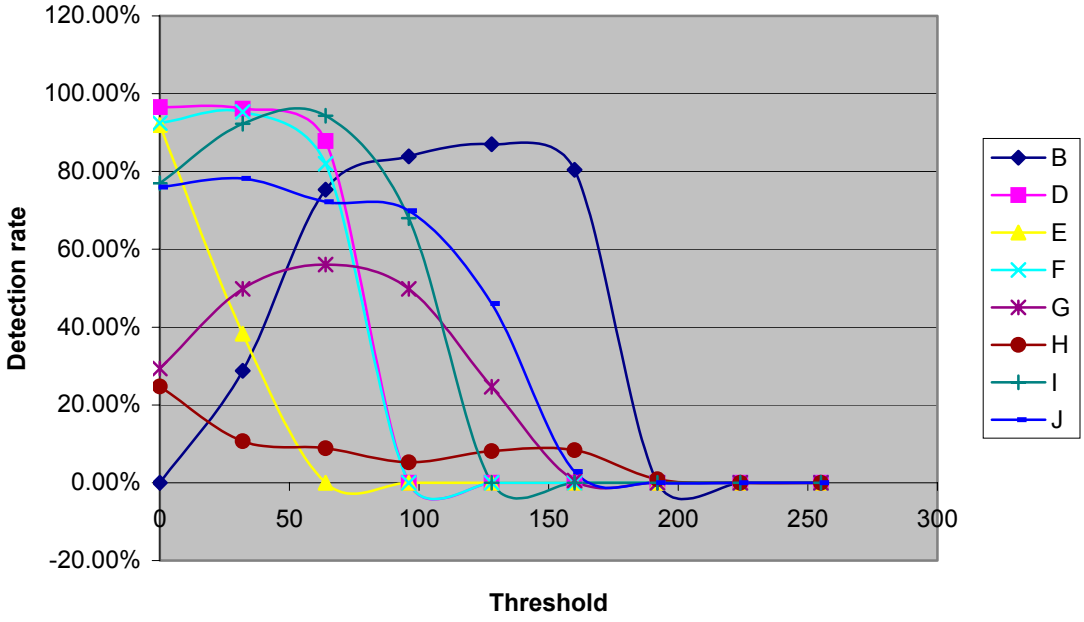


Fig. 7. 7. Effect of threshold parameter on the detection rate.

7.3.3 Effect of "high in" parameter in nose region histogram equalization

The parameter "high in" in nose region histogram equalization is similar to the parameter used in eyes region histogram equalization. Firstly, histogram equalization was performed for the whole subject image, after noticing that the function *imadjust* generates its best results when applied on small regions, we chose to perform the image histogram equalization in two separated small regions: the eye regions and the nose region.

By observing the detection rate according to the change of the parameter $high\_in$ as plotted in Fig. 7.8, we can infer that the best results are obtained above the value 0.5.

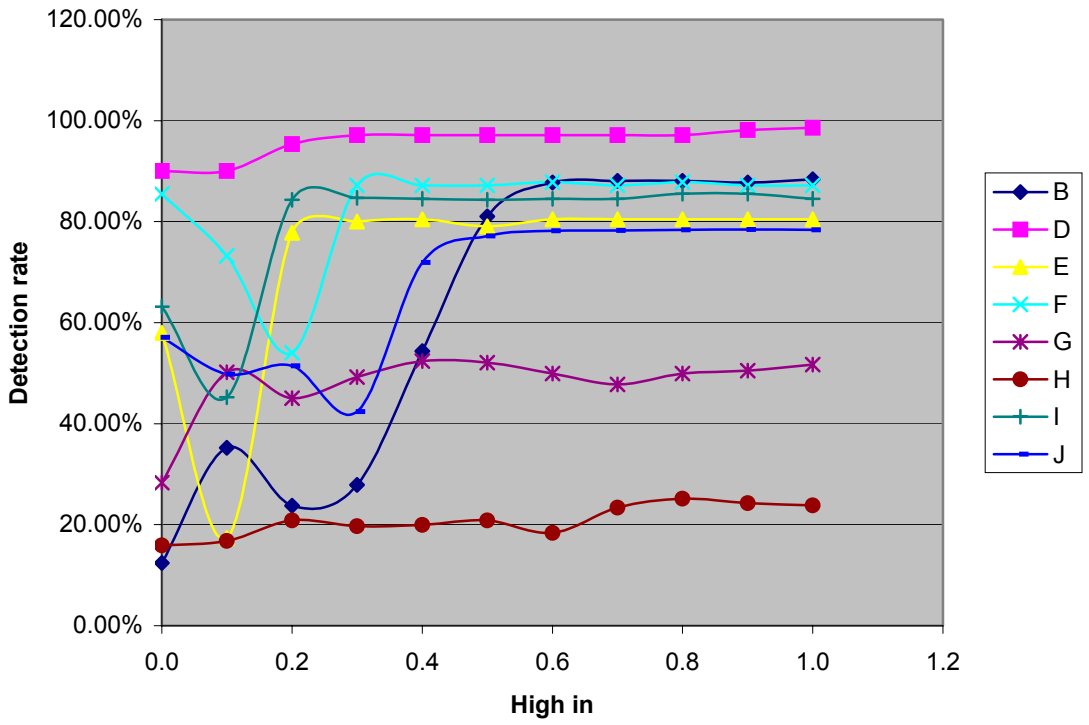


Fig. 7. 8. Effect of "High in" parameter on the detection rate.

7.3.4 Effect of threshold parameter after normalized cross correlation

The threshold parameter after normalized cross correlation is used after eyes candidates are compared to an eye average template discussed in chapter 5, by using the function *cvMatchTemplate* from OPENCV, and a threshold is used to decide whether the comparison was a match or not.

The plots in Fig 7.9 show that values before the maximum of the greyscale range 255 are the best.

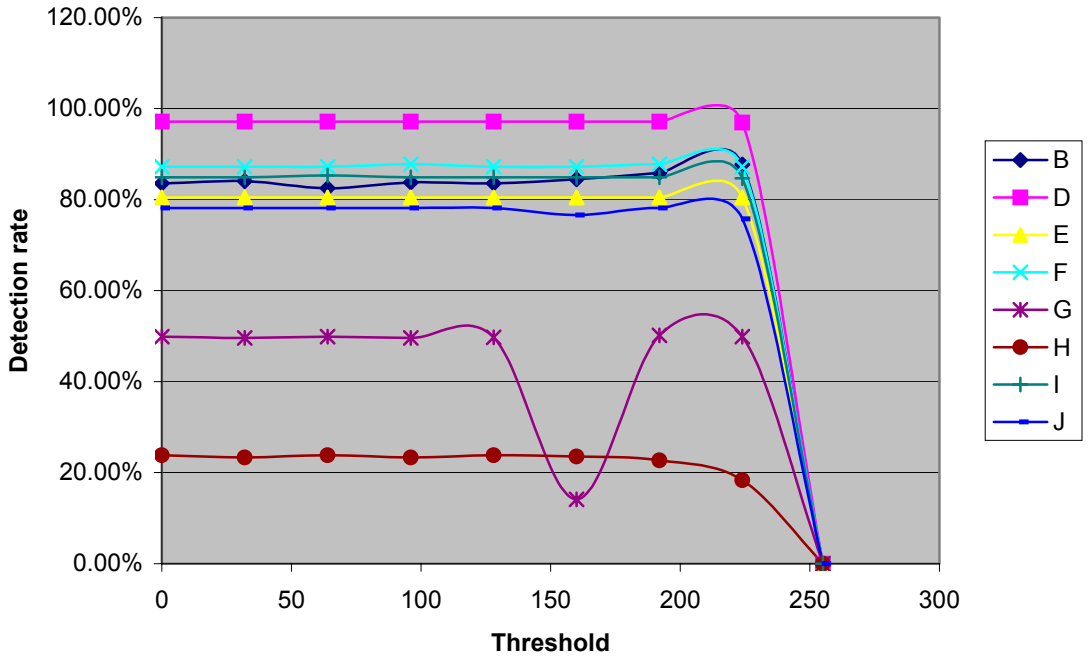


Fig. 7. 9. Effect of threshold parameter after normalized cross correlation on the detection rate.

7.3.5 Effect of threshold parameter after normalized coefficient correlation

The threshold parameter after normalized coefficient correlation is similar to the previous parameter except that a normalized coefficient correlation is used in function *cvMatchTemplate* from OPENCV. The best threshold value is selected just before the drop in the plot shown in Fig. 7.10.

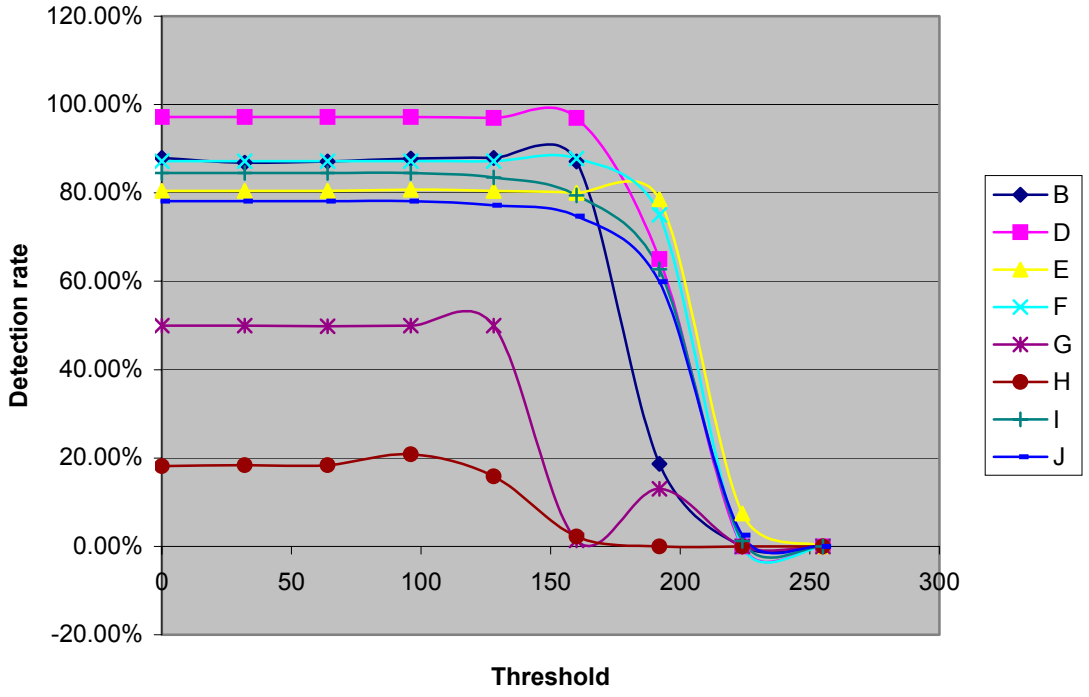


Fig. 7. 10. Effect of threshold parameter after normalized coefficient correlation on the detection rate.

7.3.6 Discussion on parameters

After analysing each system parameter plot presented above, we tried to combine system parameter values providing the highest detection rate in each plot, in order to build an optimal system with highest correct detection rate possible. However, the combination did not provide the highest detection rate expected as it is shown in the first row of Table 7.1.

Table 7. 1. Overall performance.

Video Sequence	B	D	E	F	G	H	I	J
CDR from the combination of all parameter values that provide highest CDR in each plot	81.32%	97.97%	95.66%	93.02%	52.65%	47.39%	0.00%	78.65%
CDR from parameter values that provide highest CDR in each plot	88.36%	98.58%	95.66%	99.42%	56.05%	47.39%	99.08%	78.65%

The results presented in the second row of Table 7.1 were obtained from parameter values that provide highest correct detection rate in each plot. By observing carefully the table above, we notice that video sequences G and H had the least performance among others. A presumed explanation is the imprecision in the method called 8-points algorithm defended by Hartley [56], used to estimate the fundamental matrix. This algorithm needs enhanced and well scattered edges across the stereo images which is not the case in video sequences G and H, where subjects did not wear a motion tracking device like in other video sequences, to allow enough enhanced and well scatted edges used in fundamental matrix estimation. The other video sequences were taken after the calibration of the stereo rig was done, which helped to calculate the fundamental matrix directly from geometric calibration results. By excluding the two video sequences G and H with poor performance results, we achieve an average detection rate of 93%.

7.4 3D reconstruction performance

7.4.1 Levenberg-Marquardt optimization

The estimated pose obtained from the 3D reconstruction stage and the 3D measurements obtained from the head motion tracker mounted on driver's head are not in the same coordinates system. Moreover position of this device changes from one driver to another. To compare these two measurements estimated pose from the 3D reconstruction stage have to be transposed to the head motion tracker device system coordinate.

The transposition is performed by using optimal estimation of translation and rotation parameters obtained from Levenberg-Marquardt optimization algorithm described in Appendix J. The plot comparing both data is shown in Fig.7.11.

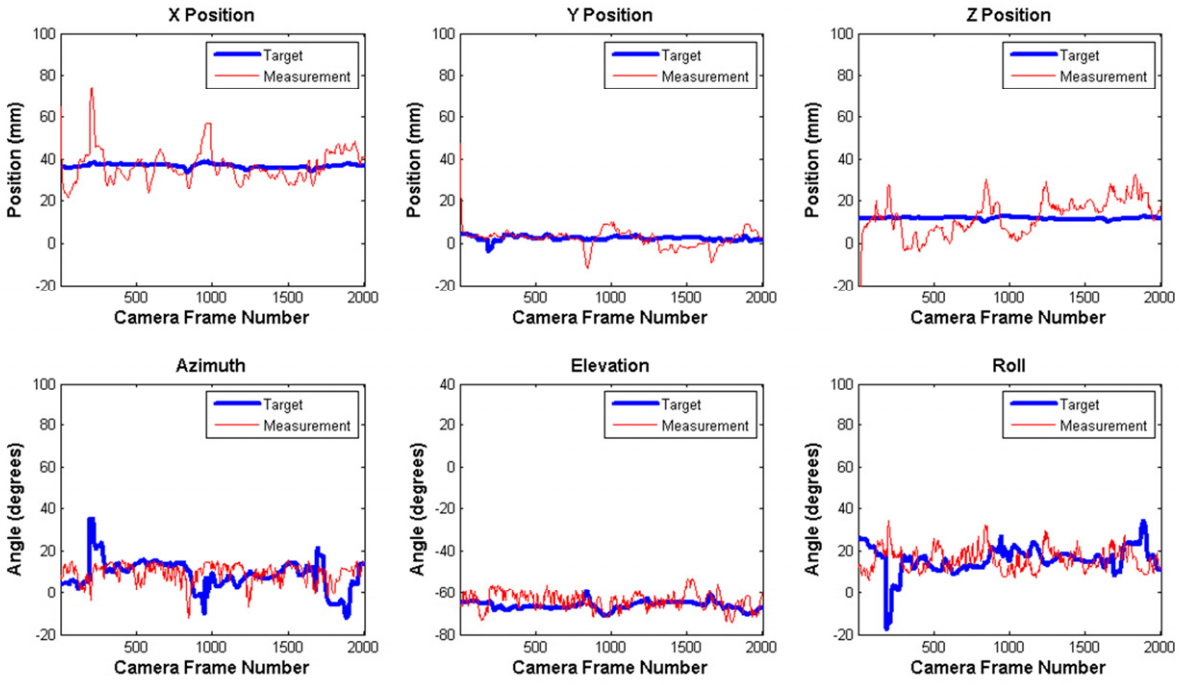


Fig. 7. 11. 3D measurements comparison for Video Sequence J

By observing the plot above, we realize that the estimated position swings along the measured position in satisfactory manner, the worst error which is in X position plot, is only 20 mm. Still the position measurement is to be well tested for subjects' head moving too much, because in the current project video database we did not have the opportunity to have such subjects. The estimated orientation is close to the measured orientation. The observed offset between estimated orientation and measured orientation is due to the delay in the response of the Kalman filter.

7.4.2 Kalman filter

Data obtained while tracking in computer vision is often noisy due to various reasons such as sensor noise or algorithms used for tracking. There is a need for filtering the raw results with, for instance, a Kalman filter.

The Kalman filter used to estimate 3D points helps reduce the quadratic error or mean square error (MSE) between the measured and estimated 3D points of the facial features as shown in equation (7.1) .

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\theta}_i - \theta_i\right)^2 \qquad (7.1)$$

where n is the data size and $\hat{\theta}$ the estimation of the data θ.

Results presented in Table 7.2 show that for all video sequences tested, the Kalman filter always yield a smaller mean square error.

Table 7. 2. Mean Square Error between estimated pose with and without Kalman filter.

MSE		Video Sequences					
		B	D	E	F	I	J
X(mm)	Without Kalman	8.264	2.928	28.934	4.427	10.937	53.737
	With Kalman	7.413	2.079	19.828	4.137	8.663	49.004
Y(mm)	Without Kalman	7.088	0.563	6.898	1.644	4.516	12.920
	With Kalman	6.642	0.334	5.499	1.074	3.004	11.985
Z(mm)	Without Kalman	15.247	0.811	19.989	5.026	13.563	75.700
	With Kalman	6.022	0.534	17.288	1.495	11.083	68.802
Azimuth (degrees)	Without Kalman	1.345	36.086	5.220	15.747	3.800	173.118
	With Kalman	1.279	15.628	3.844	5.998	1.797	105.015
Elevation (degrees)	Without Kalman	3.884	8.457	19.221	21.129	6.549	51.444
	With Kalman	1.849	4.384	7.298	6.795	2.211	36.462
Roll (degrees)	Without Kalman	23.309	13.821	39.124	8.036	9.101	115.121
	With Kalman	20.081	5.943	12.482	4.631	2.419	100.497

The Kalman filter not only minimizes the mean square error, it allows also a smooth transition between estimated 3D pose as shown in Fig. 7.2.

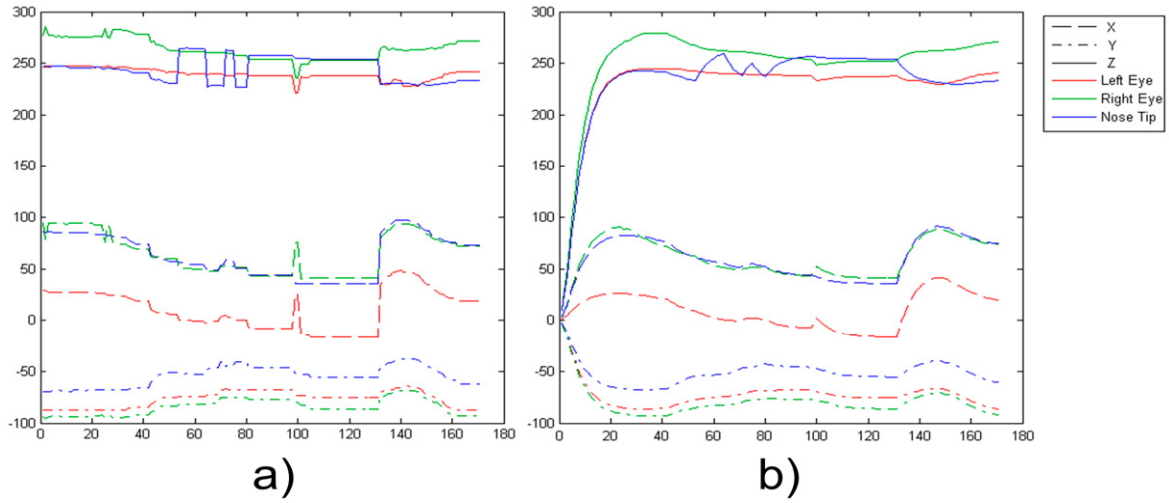


Fig. 7. 12. a) Cockpit Setup Diagram3D points without *Kalman filter* b) the same 3D points with *Kalman filter*.

7.5 Summary

The evaluation of the system performance was presented in this chapter. The correct head detection rate and correct facial features detection rate show that the system discussed in this thesis is able to track the head motion in 3D space and in real-time when system parameters are set properly. Finally, the results obtained from the estimated pose are comparable to the data from the head motion tracker.

Conclusion and Suggestions for Future Work

A computer vision system for tracking the head pose in 3D was designed and implemented. As the system is part of an integrated system to analyze the behaviour of the drivers and could be expandable for a later use in other similar systems, a modular design was adopted by dividing the system into different modules. Four main modules were implemented: image acquisition, head detection, facial features extraction and detection, and the 3D reconstruction.

An affordable image acquisition system using two Prosilica CV640 cameras was proposed, and infrared projectors were used to illuminate driver's face without obscuring their view.

The head detection module was implemented by taking advantage of the light-controlled environment. This helped to implement a simple yet very cost effective algorithm that contributed to increase video sequence processing speed and eliminated potential problems caused by areas that may look like facial features but are located outside of the head.

The facial features module consists of a detection algorithm that was implemented to detect roughly the eyes location and use of camera calibration and epipolar geometry to extract all the facial features from the image sequences. This extraction was validated by a neural network that led to a robust system.

Finally, a linear triangulation approach was implemented to achieve 3D reconstruction using all selected facial features. A Kalman filter was also used to smooth the transition of 3D points from successive video frames. The system is suitable for real-time tracking.

Future work will consist in extending the algorithms to drivers wearing glasses and reducing the maximum error for azimuth, pitch and roll in 3D estimates. We also intend to install the tracking system on board a real car. In this case, the Head Detection algorithm will need to be revisited in order to be able to cope with varying background conditions. Finally, the training of the neural network will be automated in order to facilitate 2D tracking whenever the cameras are moved.

Glossary of Abbreviations and Symbols

2D	Two dimensions.
3D	Three dimensions.
ASL	Applied Science Laboratories.
COBVIS-D	Cephalo-Ocular Behaviour and Visual Search Patterns of Drivers.
DFT	Discrete Fourier Transform.
FPS	Frames Per Second (Frame rate).
HSL	Hue, Saturation and Luminance colors channels.
HSV	Hue, Saturation and Value colour channels.
MATLAB	Matrix Laboratory (a numerical computing environment and programming language).
MLP	Multi-Layer Perceptron.
OPENCV	Open Computer Vision Library.
OPENGL	Open Graphics Library.
RGB	Red, Green and Blue colour channels.
VFF	Vertical Frequency Filter.

Bibliography

[1] C. C. Wang, D. B. Carr, *Older Driver Safety, A Report from the Older Drivers Project,* Journal of the American Geriatrics Society Volume 52, Issue 1, Page 143-149, January 2004.

[2] *Computer Vision*, Wikipedia the Free encyclopaedia http://en.wikipedia.org/wiki/Computer_vision accessed January 20, 2007.

[3] E-F. Lally, *Mosaic Guidance For Interplanetary Travel*, Space Flight Report to the Nation, American Rocket Society, New York, pages 2249–61, October 9–15, 1961.

[4] R. Huber, C. Nowak, B. Spatzek, *Image Acquisition Using Aperture Control Adapted To Spatio-temporal Properties*, Journal Machine Vision and Applications Volume15, Issue 4, Pages 204-215, October 2004.

[5] D. Gorodnichy, S. Malik, G. Roth, *Affordable 3D Face Tracking Using Projective Vision*, In Proc. Intern. Conf. on Vision Interface (VI'2002), Calgary, May 2002.

[6] K. Terada, A. Oba, A. Ito, *3D Human Head Tracking Using Hypothesized Polygon Model*, Systems, Man and Cybernetics, 2005 IEEE International Conference, Volume 2, Pages:1396 - 1401, October 2005.

[7] G. Loy, E. J. Holden, R. Owens, *A 3D Head Tracker For An Automatic Lip-reading System*, Proc of Australian Conference on Robotics and Automation (ACRA), 2000.

[8] M. L. Cascia, S. Sclaroff, V. Athitsos, *Fast Reliable Head Tracking Under Varying Illumination: An Approach Based On Registration Of Texture-Mapped 3D Models*, IEEE PAMI, vol. 21, no.6, June 1999.

[9] D. B. Russakoff, M. Herman, *Head Tracking Using Stereo,* Machine Vision and Applications, vol. 13, pages 164-173, 2002.

[10] H. Nanda, K. Fujimura, *A Robust Elliptical Head Tracker, Automatic Face And Gesture Recognition*, 2004. Proceedings. Sixth IEEE International Conference, Pages 469 - 474, 17-19 May 2004.

[11] I. Ravyse, V. Enescu, H. Sahli, *Kernel-based Head Tracker For Videophony*, Image Processing, ICIP 2005. IEEE International Conference, 11-14 September 2005.

[12] M. Kourogi, T. Kurata, *A Wearable Augmented Reality System With Personal Positioning based on Walking Locomotion Analysis*, Proceedings of the 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality, Page 342, 2003.

[13] J. D. Mulder, J. Jansen, A. van Rhijn, *An Affordable Optical Head Tracking System for Desktop VR/AR Systems*, Proceedings of the workshop on Virtual environments, Zurich, 2003.

[14] W.D. McCarty, S. Sheasby, P. Amburn, M.R. Stytz, C. Switzer, *A Virtual Cockpit For A Distributed Interactive Simulation*, Computer Graphics and Applications, IEEE Volume 14, Issue 1, Pages 49 - 54, January 1994.

[15] F. E. Ababsa, M. Mallem, *Inertial And Vision Head Tracker Sensor Fusion Using A Particle Filter For Augmented Reality Systems*, Circuits and Systems, ISCAS 04. Proceedings of the 2004 International Symposium, 23-26 May 2004.

[16] M. Seki, T. Wada, H. Fujiwara, K. Sumi, *Background Subtraction Based On Co Occurrence Of Image Variations*, Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on Volume 2, Pages II-65 - II-72 vol.2, Issue 18-20 June 2003.

[17] A. M. McIvor , *Background Subtraction Techniques*, In Proc. of Image and Vision Computing, Auckland, New Zealand, 2000.

[18] A. Lemieux, *Système D'identification De Personnes Par Vision Numérique*, M. Sc. Thesis, Université Laval, 2003.

[19] J. Heikkila and O. Silven, *A Real-time System For Monitoring Of Cyclists And Pedestrians*, Second IEEE Workshop on Visual Surveillance Fort Collins, Colorado pages 74-81, June 1999.

[20] X. Maldague, S. Marinetti, *Pulse Phase Infrared Thermography*, http://www.gel.ulaval.ca/~maldagx/r_1086.pdf accessed April 9, 2007.

[21] T. Kanade, A. Yoshida, K. Oda, H. Kano et M. Tanaka, *A Stereo Machine For Video-rate Dense Depth Mapping And Its New Applications*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 196–202, 1996.

[22] K. Yong-Guk, L. Jeong-Eom, K. Sang-Jun, C. Soo-Mi and P. Gwi-Tae, *Head Detection Of The Car Occupant Based On Contour Models And Support Vector*, Proceedings of the 18th international conference on Innovations in Applied Artificial Intelligence, Pages 59-61, 2005.

[23] Y. Owechkp, N. Srinivasa, S. Medasani, and R. Boscolo, *Vision-Based Fusion System for Smart Airbag Applicaions*, IEEE, Intelligent Vehicle Symposium, vol. 1, pages 245-250, 2002.

[24] P. Viola and M. J. Jones, *Rapid Object Detection Using A Boosted Cascade Of Simple Features*, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1, pages I-511-I-518 vol.1. IEEE CVPR, 2001.

[25] R. Lienhart and J. Maydt, *An Extended Set Of Haar-like Features For Rapid Object Detection*, IEEE ICIP 2002, Vol. 1, pages 900-903, September 2002.

[26] A. Nüchter, H. Surmann and J. Hertzberg, *Automatic Classification Of Objects in 3D Laser Range Scans*, http://www.ais.fraunhofer.de/ARC/3D/download/ias2004/ias2004.html accessed April 15, 2007.

[27] R. Patil, P. E. Rybski, T. Kanade, and M. M. Veloso, *People Detection And Tracking In High Resolution Panoramic Video Mosaic*, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2004), vol. 1, pages 1323-1328, 2004.

[28] C. Yisheng, *Human Head Detection And Tracking*, unpublished.

[29] *Face*, Wikipedia The Free encyclopaedia http://en.wikipedia.org/wiki/Face accessed April 18, 2007.

[30] D. Gorodnichy, *On Importance Of Nose For Face Tracking*, In Proc. IEEE Intern. Conf. on Automatic Face and Gesture Recognition (FG'2002), Washington, D.C., May 2002.

[31] E. W. Weisstein, *Plane*, MathWorld, A Wolfram Web Resource. http://mathworld.wolfram.com/Plane.html.

[32] R. Feris, T. Campos and R. Cesar, *Detection And Tracking Of Facial Features In Video Sequences*, Lecture Notes in Artificial Intelligence, vol. 1793, pages 127-135, Springer-Verlag, April 2000.

[33] H. Rein-Lien, M. Abdel-Mottaleb, A.K. Jain, *Face Detection In Color Images*, Proceedings. 2001 International Conference on Image Processing, Volume 1, Issue , Pages1046 - 1049 vol.1, 2001.

[34] *YcbCr*, Wikipedia The Free encyclopaedia http://en.wikipedia.org/wiki/YCbCr accessed April 30, 2007.

[35] A. Forsyth and J. Ponce, *Computer Vision A Modern Approach*, Prentice Hall, New Jersey , 2003.

[36] R.T. Kumar, S. K. Raja, A.G. Ramakrishnan, *Eye Detection Using Color Cues And Projection Functions*, Image Processing. 2002. Proceedings. 2002 International Conference on, pages III-337- III-340 vol.3, 2002.

[37] H. Rowley, S. Baluja, T. Kanade, *Rotation Invariant Neural Network-Based Face Detection*, Proceedings of Computer Vision and Pattern Recognition, 1998.

[38] P. Debevec, *A Neural Network For Facial Feature Location*, UC Berkeley CS283 Project Report, December 1992. http://www.debevec.org/FaceRecognition/

[39] S. Kawato, N. Tetsutani, *Circle-Frequency Filter And Its Application*, Proc. Int. Workshop on Advanced Image Technology, pages 217-222, February 2001.

[40] *Template Matching*, Wikipedia The Free encyclopaedia http://en.wikipedia.org/wiki/Template_matching accessed May 06, 2007.

[41] K. Briechle, U. D. Hanebeck, *Template Matching using Fast Normalized Cross Correlation*, Proceedings of SPIE, Band 4387, AeroSense Symposium, Orlando, Florida, 2001.

[42] S. Gurbuz, K. Kinoshita, and S. Kawato, *Real-time Human Nose Bridge Tracking In Presence Of Geometry And Illumination Changes*, in Second International Workshop on Man-Machine Symbiotic Systems, Kyoto, Japan, 2004.

[43] *Feature Vector*, Wikipedia the Free encyclopaedia http://en.wikipedia.org/wiki/Feature_vector accessed May 08, 2007.

[44] J. O. Kim, J. S. Kim, Y. R. Seo, B. R. Lee, C. H. Chung, K. S. Lee, W. Y. Yim, S. H Lee, *On Extraction Of Facial Features From Color Images*, Computational Science and Its Applications,Springer Berlin / Heidelberg, 2004.

[45] S. Murai, *GIS Work Book (Technical Course)*, Japan Association of Surveyors (JAS) Institute of Industrial Science, Tokyo, 1997.

[46] K. Peng, L. Chen, S. Ruan, G. Kukharev, *A Robust And Efficient Algorithm For Eye Detection On Gray Intensity Face*,ICAPR (2) pages 302-308, 2005.

[47] A. Fusiello, E. Trucco and A. Verri, *A Compact Algorithm For Rectification Of Stereo Pairs*, Machine Vision and Applications.12, pages 16-22, 2000.

[48] J-Y. Bouguet, *Camera Calibration Toolbox For Matlab*, http://www.vision.caltech.edu/bouguetj/calib_doc/ accessed May 21, 2007.

[49] M. Parizeau, *Réseaux De Neurones GIF-21140 et GIF–64326*, Université Laval, Québec, 2006.

[50] H. Demuth, M. Beale and M. Hagan, *Neural Network Toolbox User's Guide, Mathworks*, Massachusetts ,2006.

[51] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, *Bundle Adjustment - A Modern Synthesis,* Vision Algorithms: Theory and Practice, Springer-Verlag, pages 298-372, 2000.

[52] I. A. Lourakis and A. Argyros, *Is Levenberg-Marquardt The Most Efficient Optimization Algorithm For Implementing Bundle Adjustment*, Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2, Pages 1526 - 1531, 2005.

[53] R. I. Hartley, P. Sturm, *Triangulation*, Computer Vision and Image Understanding, Vol. 68, no.2, pages 146-157, 1997.

[54] E. Cheever, *Introduction To Kalman Filters*, http://www.swarthmore.edu/NatSci/echeeve1/Ref/Kalman/ScalarKalman.html, accessed July 1, 2007.

[55] G. Welch, G. Bishop, *An Introduction To The Kalman Filter*, http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, accessed July 1, 2007.

[56] R. I. Hartley, *In Defence of the 8-Point Algorithm*, International Conference on Computer Vision, 1995.

Appendix A: Motion Tracking Device (Flock of Bird)

Fig. A. 1. Motion Tracker.

Table A. 1. Motion Tracker Specifications.

Specifications	Values
Price	$ 2,495
Technical	
Tracking Range	±4’ (1.2m) ±10’ (3.05m) optional in any direction
Angular Range	±180° Azimuth & Roll, ± 90° Elevation
Static Accuracy	Position: 0.07" (1.8mm) RMS
Orientation	Position: 0.07" (1.8mm) RMS
Static Resolution	Position: 0.02’ (0.5mm) @ 12" (30.5cm)
Orientation	0.1° @ 12" (30.5cm)
Update Rate	Up to 144 measurements/second
Outputs	X,Y, Z positional coordinates and orientation angles, or rotation matrix
Interface	RS-232 with selectable baud rates to 115,200
Format	Binary
Modes	Point or Stream
Physical	
Transmitter	Mid-Range Transmitter: 9.6cm cube with 3.05M cable, or extended range
Transmitter option	30.5cm cube with 6.1m cable
Sensor	25.4mm x 25.4mm x 20.3mm cube (or optional 3-button mouse) with
	or 10.7M cable. Weight: 21 g (0.7 oz) without cable, 169 g (6.0 oz)
Enclosure	with 3.05M cable, 394 g (13.9 oz) with 10.7M cable.
Power	User provided or optional external plug-in: US/European version
Environment	Metal objects and stray magnetic fields in the operation volume will
	degrade performance.

Appendix B: Eye Tracking Device (ASL Model R6)

Fig. B. 1. Eye Tracker.

Table B. 1. Eye Tracker specifications.

Specifications	Values
Price	$ 32,540 - $ 42,455
Control Unit	
Dimensions (H/W/D)	3 in/9.75 in/10.25 in
Weight	4.25 lbs
Power	100-240 VAC, 25 watts
Display	9 inch b&w monitors for eye and scene cameras
Remote Optics	
Sampling and Output Rates	50 Hz or 60 Hz, 120 Hz, 240 Hz and 360 Hz (optional)
Measurement principle	pupil-corneal reflection
System accuracy	0.5 degree visual angle
Resolution	0.25 degree visual angle
Head movement	one square foot
Max. distance optics to eye	40 in
Visual range	50 degrees horizontally, 40 degrees vertically
Dimensions (H/W/D)	4 in/5.5 in/6 in
Weight	2.75 pounds

Appendix C: Camera (Prosilica CV640)

Fig. C. 1. Camera Prosilica.

Table C. 1. Camera Prosilica Specifications.

Specifications	Values
Price	$ 1,550
Sensor Size (H x V)	659 x 494 pixels
Sensor Format	½ inc
Dynamic Range	65 dB (112 dB in EDR mode)
Sensor Type	Progressive Scan
Exposure Type	Snapshot shutter
Pixel Size	9.9 x 9.9 um
Frame Rate (8-bit raw)	120 fps (NI CVS) 100fps (MS Windows) – more with ROI
Output Type	IEEE 1394A
DCAM Compliance	IIDC 1.30 & 1.31
Output Format	8 or 10 bits per pixel
Synchronization	External trigger and sync.
Exposure Control	Programmable - 10us to 5s
Power	1.8 W 12VDC @ 150 mA
Lens Mount	C-mount
Housing Size	79 x 51 x 38 (L x W x H in mm)
Weight	220 g

Appendix D: Infrared Illuminators (CSI-IR)

Fig. D. 2. Infrared Illuminators.

Table D. 2. CSI-IR Infrared Illuminators Specifications.

Specifications	Values
Price	$ 286
Distance	18m (60ft) – 200m(660ft)
Illumination Method	Special LED array
Power	6W-55W, DC or AC input
Lifespan	10,000 + hours
Construction	Extruded aluminium housing
Weight	150g (33lbs)-1.3kg(2.8lbs)
Wave length	850nm, 940nm

Appendix E: Head Detection Algorithm using Head-Shoulder Contour method

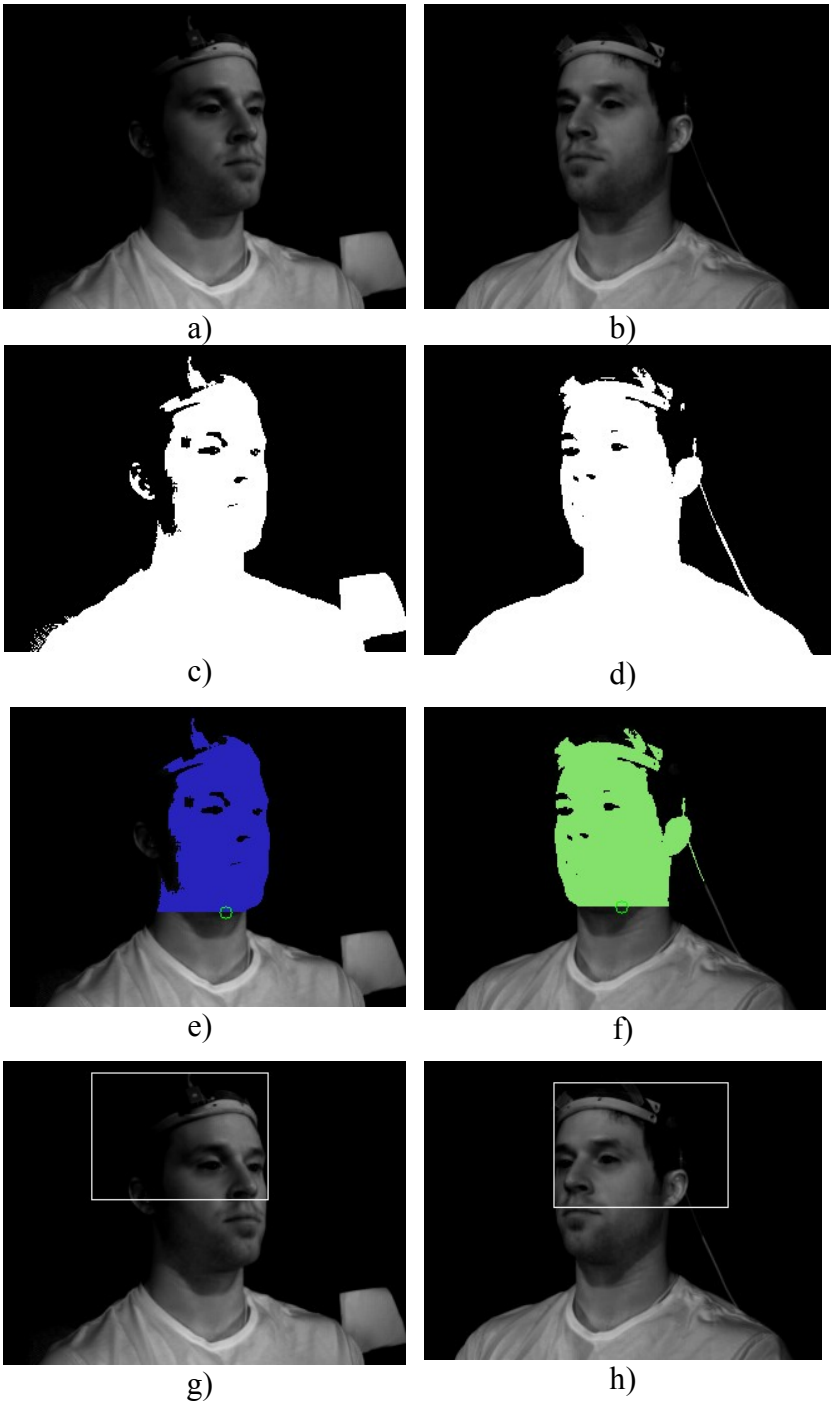


Fig. E. 1. a, b) Original frames c,d) Images thresholding and largest blob detection e,f) Head detection and g,h)Bounding box assignation to the head.

Appendix F: Template matching functions in Open Computer Vision Library (OPENCV)

"Given a source image with pixels and a template with pixels, the resulting image has pixels, and the pixel value in each location (x,y) characterizes the similarity between the template and the image rectangle with the top-left corner at (x,y) and the right-bottom corner at (x + w - 1, y + h - 1)".

$$S(x,y)=\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\left[T(x',y')-I(x+x',y+y')\right]^2$$

a)

$$S(x,y)=\frac{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\left[T(x',y')-I(x+x',y+y')\right]^2}{\sqrt{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}T(x',y')^2\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}I(x+x',y+y')^2}}$$

b)

$$C(x,y)=\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}T(x',y')I(x+x',y+y')$$

c)

$$C(x,y)=\frac{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}T(x',y')I(x+x',y+y')}{\sqrt{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}T(x',y')^2\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}I(x+x',y+y')^2}}$$

d)

$$R(x,y)=\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\tilde{T}(x',y')\tilde{I}(x+x',y+y')$$

e)

$$R(x,y)=\frac{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\tilde{T}(x',y')\tilde{I}(x+x',y+y')}{\sqrt{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\tilde{T}(x',y')^2\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\tilde{I}(x+x',y+y')^2}}$$

f)

a) Fig. F. 1. Matching Algorithms a) Square difference b) Normalized square difference c) Cross correlation d) Normalized cross correlation e) Correlation coefficient f) Normalized correlation coefficient.

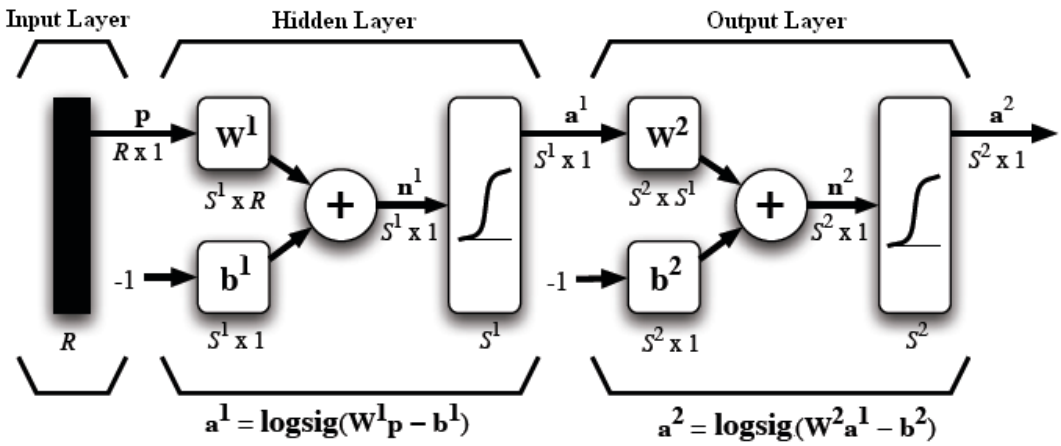
where $I(x,y)$ is a value of image pixel and $T(x,y)$ a value of template pixel at location (x, y).

where $\tilde{T}(x',y')=T(x',y')-\overline{T}$ and $\tilde{I}(x+x',y+y')=I(x+x',y+y')-\bar{I}(x,y)$, and where $\overline{T}$ stands for the average value of pixels in the template raster and $\bar{I}(x,y)$ stands for the average value of the pixels in the current window of the image.

Appendix G: Neural Network : Multilayer Perceptron

The MLP is a parametric technique to approximate a vector function of some input with a series of layers [35].
Each layer has a weight matrix **W**, a bias vector **b** and an output vector **a**. The network proposed for the validation of 2D results is shown in Fig. G.1. It has **R** inputs in the first layer also called the input layer, 10 hidden neurons in a hidden layer and 1 neuron in an output layer.

Fig. G. 1. Multilayer perceptron.



Source: Parizeau [49].

The output equation (F.1) of every layer is described by:

$$a^k = f^k\left(W^k a^{k-1} + b^k\right) \qquad \text{For } k = 1...M \quad \text{(G.1)}$$

where M is the number of layers. To approximate the function, the network needs to be trained in order to adjust its weights. The algorithm needed to do the adjustment is called *Back propagation*. At every training epoch an error vector $e(t)$ is obtained in equation (G.2).

$$e(t) = d(t) - a^M(t) \qquad \text{(G.2)}$$

where $d(t)$ is desired output, the target in the other words. The performance is expressed by minimizing the estimated mean square error $\hat{F}(x)$ in equation (G.3).

$$\hat{F}(x) = E\left(e(t)^T e(t)\right) \qquad \text{(G.3)}$$

where E is the mathematical expectation and x combines both network weights and bias. Notice that it is easy to do all mathematics in the last layer where the target is given, the problem arises at the intermediate layers and thus the chain rule of derivate was needed to keep propagating the error [49].

To optimize $\boldsymbol{x}$ we use the gradient descent algorithm described in the following equation (G.4).

$$x_{k+1} = x_k - \eta_k g_k \tag{G.4}$$

where η is the learning rate and g the current gradient.

The gradient descent algorithm is too slow, a faster technique was needed. The Quasi-Newton method was developed to solve the speed problem. Its basic step is presented in equation (G.5).

$$x_{k+1} = x_k - A_k^{-1} g_k \tag{G.5}$$

where A_k^{-1} is the Hessian Matrix (second derivates) of the performance index at the current values of the weights and biases. For our problem we chose the Levenberg-Marquardt algorithm that approximates the Hessian matrix H and the gradient g as presented in equation (G.6) by using the Jacobian matrix J that contains first derivates of the network errors e with respect to the weights and biases.

$$\left.\begin{aligned} H &= J^T J \\ g &= J^T e \end{aligned}\right\} \tag{G.6}$$

The algorithms step update as presented in equation (G.7) becomes

$$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T e \tag{G.7}$$

where μ is zero the algorithm becomes Newton's method approximating Hessian matrix. When μ is large it becomes gradient descent method with small step size [51].

Appendix H: Kalman filter

The *Kalman filter* is defined as "a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error" [54].

In any discrete time linear system subject to noise, a true state x_k of a process at time k that depends on the state x_{k-1} at time (k-1) and the measurement z_k of this true state x_k are related according to the linear stochastic difference equations in (H.1) and diagrammed in Fig. H.1.

$$\begin{cases} x_k = A_k x_{k-1} + B_k u_k + w_k \\ z_k = H_k x_k + v_k \end{cases} \qquad \text{(H.1)}$$

where

- A_k is the state transition n x n matrix applied to the previous state x_{k-1}
- B_k is the control-input n x 1 matrix applied to the control vector u_k
- H_k is the measurement matrix which maps the true state space into the measured space.
- w_k and v_k are normally-distributed process and measurement noise, respectively with Q as process noise covariance matrix and R measurement noise covariance matrix.

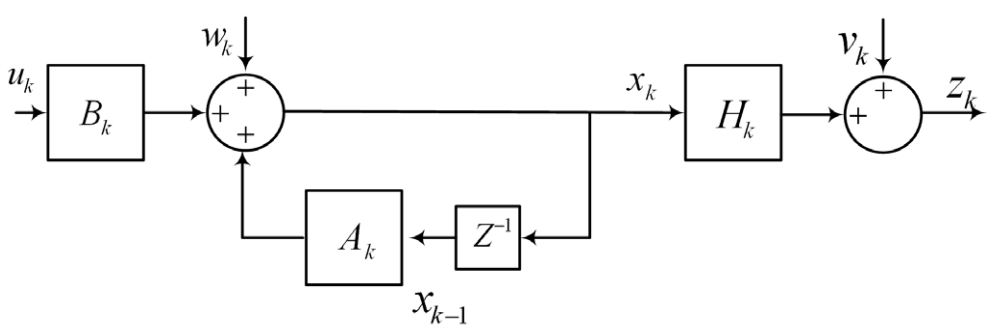


Fig. H. 1. Kalman filter state and measurement models.

Unfortunately, this ideal model cannot be obtained due to the unknown noise sources w_k and v_k [55], hence the use of the estimated model in determining the system state and measurement as diagrammed in Fig. H.2.

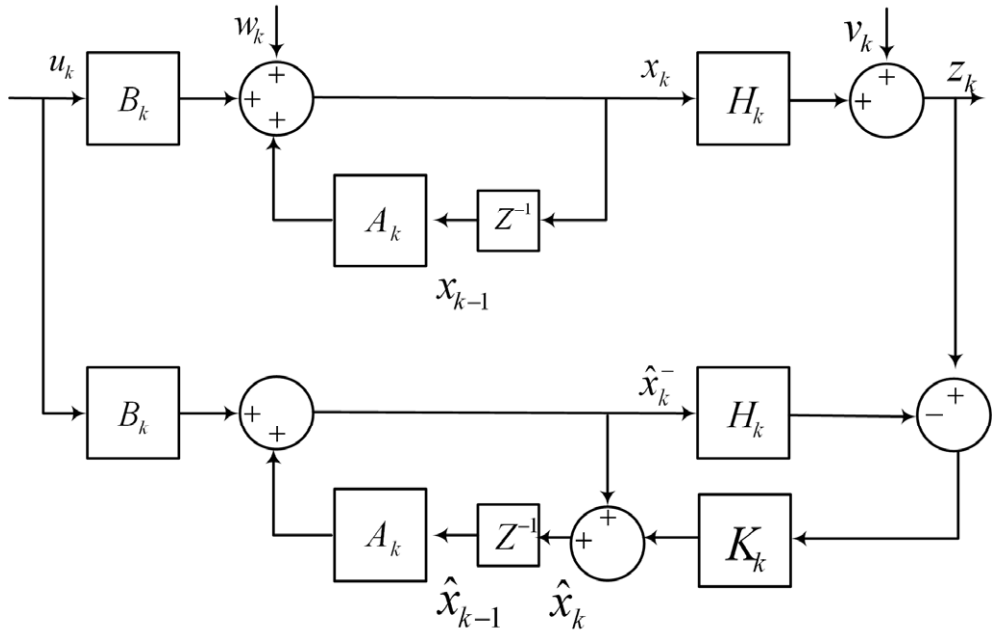


Fig. H. 2. Comparison between ideal model and estimated model.

In this estimates model, the absence of noise sources w_k and v_k is compensated by some equations series described in equation (H.2).

$$\begin{cases} \hat{x}_k^- = A_k \hat{x}_{k-1} + B_k u_k \\ R = z_k - \hat{z}_k = z_k - H_k \hat{x}_k^- \\ \hat{x}_k = \hat{x}_k^- + K_k(R) = \hat{x}_k^- + K_k\left(z_k - H_k \hat{x}_k^-\right) \end{cases} \qquad \text{(H.2)}$$

where

- the original estimate of x_k is now called $\hat{x}_k^-$ and is referred to as a *priori* estimate this a priori estimate is used to predict an estimate for the output $\hat{z}_k$
- the difference R is called measurement residual or innovation if it is zero the estimation is perfect.

The quantity K_k called *kalman gain* is used to refine the estimate.

Finally, the kalman filtering is achieved in two phases: predict phase and correct phase as shown in Fig. H.3 where the covariance error P_k is used to compute the *kalman gain*.

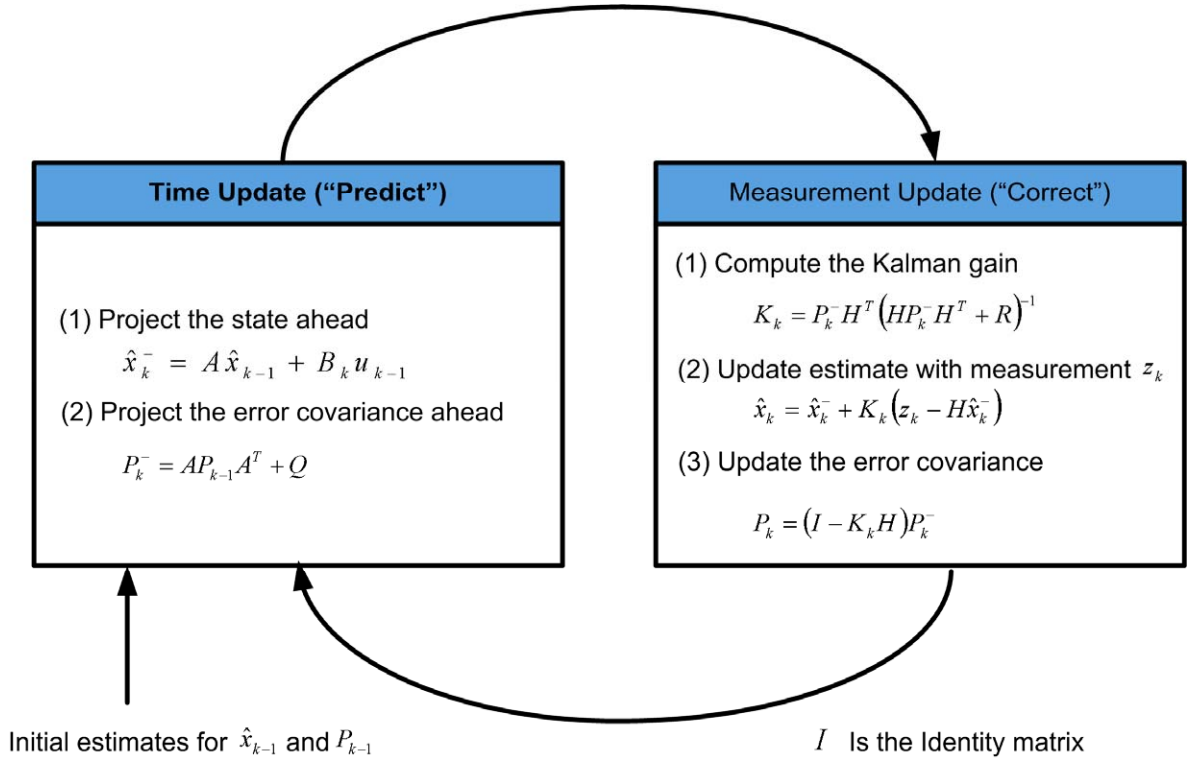


Fig. H. 3. Kalman filter phases.

A computer vision library called OPENCV was used below for the implementation of the *kalman filter*.

```
typedef struct CvKalman
{
    int MP;                       // number of measurement vector dimensions
    int DP;                       // number of state vector dimensions
    int CP;                       // number of control vector dimensions
    CvMat* state_pre;             // predicted state (x'(k)): x(k)=A*x(k-1)+B*u(k)
    CvMat* state_post;            // corrected state (x(k)) : x(k)=x'(k)+K(k)*(z(k)-H*x'(k))
    CvMat* transition_matrix;     // state transition matrix (A)
    CvMat* control_matrix;        // control matrix (B)
                                  // (it is not used if there is no control)
    CvMat* measurement_matrix;    // measurement matrix (H)
    CvMat* process_noise_cov;     // process noise covariance matrix (Q)
    CvMat* measurement_noise_cov; // measurement noise covariance matrix (R)
    CvMat* error_cov_pre;         // priori error estimate covariance matrix (P'(k)):
                                  // P'(k)=A*P(k-1)*At + Q
    CvMat* gain;                  // Kalman gain matrix (K(k)):
                                  // K(k)=P'(k)*Ht*inv(H*P'(k)*Ht+R)
    CvMat* error_cov_post;        // posteriori error estimate covariance matrix (P(k)):
                                  // P(k)=(I-K(k)*H)*P'(k)
}
CvKalman;
```

```
const int ELEMENTS_NBR =  3;      // The 3D points has three coordinates x, y and z

// Kalman filter main struct initialization
CvKalman* lpKalmanFilter = cvCreateKalman( ELEMENTS_NBR, ELEMENTS_NBR, 0 );

// Matrix for measurements for a 3D point
CvMat*   lpMeasurement  = cvCreateMat( ELEMENTS_NBR, 1, CV_32FC1 );

//Initialization of the kalman matrices
cvSetIdentity(lpKalmanFilter->transition_matrix,     cvRealScalar(1));        // Matrix A
cvSetIdentity(lpKalmanFilter->measurement_matrix,    cvRealScalar(1));        // Matrix H
cvSetIdentity(lpKalmanFilter->process_noise_cov,     cvRealScalar(0.125));    // Matrix Q
cvSetIdentity(lpKalmanFilter->measurement_noise_cov, cvRealScalar(102.4));    // Matrix R
cvSetIdentity(lpKalmanFilter->error_cov_post,        cvRealScalar(1));        // Matrix P

// initial state initialization
lpKalmanFilter->state_post->data.fl[0] = 0;                            // x
lpKalmanFilter->state_post->data.fl[1] = 0;                            // y
lpKalmanFilter->state_post->data.fl[2] = 0;                            // z

//Function to predict a filtered point for each video frame
CvPoint3D64f PointPrediction(CvPoint3D64f measuredPoint) {
       CvPoint3D64f lpredictedPoint;

       //measured point
       lpMeasurement->data.fl[0] = measuredPoint.x;                    // x
       lpMeasurement->data.fl[1] = measuredPoint.y;                    // y
       lpMeasurement->data.fl[2] = measuredPoint.z;                    // z

       // Necessary to first predict and then update estimates, Predict next value
        // Function updates kalman->state_pre which is next predicted
        cvKalmanPredict(lpKalmanFilter,0);

        // Function updates internal matrices
        // Function updates kalman->state_post which is corrected
        cvKalmanCorrect(lpKalmanFilter,lpMeasurement);

       lpredictedPoint.x = lpKalmanFilter->state_post->data.fl[0];     // x
       lpredictedPoint.y = lpKalmanFilter->state_post->data.fl[1];     // y
       lpredictedPoint.z = lpKalmanFilter->state_post->data.fl[2];     // z

       return lpredictedPoint;
}
```

Appendix I: Motion Tracker 3D Measurements Example

Table I. 1. Motion Tracker 3D Measurements Example.

Frames	X	Y	Z	Azimuth	Elevation	Roll
1	37.138	6.251	16.520	-5.603	-69.212	43.856
3	37.149	6.240	16.543	-5.603	-69.322	43.834
4	37.160	6.240	16.543	-5.515	-69.300	43.768
5	37.171	6.229	16.554	-5.559	-69.388	43.790
6	37.182	6.240	16.554	-5.515	-69.432	43.790
7	37.182	6.240	16.587	-5.603	-69.542	43.834
8	37.193	6.251	16.576	-5.493	-69.586	43.725
9	37.216	6.240	16.587	-5.471	-69.630	43.703
10	37.227	6.240	16.599	-5.493	-69.740	43.725
11	37.238	6.240	16.610	-5.581	-69.805	43.768
12	37.260	6.251	16.621	-5.537	-69.871	43.746
13	37.272	6.240	16.632	-5.581	-69.871	43.790
14	37.283	6.251	16.621	-5.515	-69.937	43.725
15	37.294	6.262	16.632	-5.449	-69.981	43.703
16	37.327	6.251	16.632	-5.493	-70.069	43.703
17	37.338	6.240	16.643	-5.581	-70.113	43.768
18	37.338	6.251	16.677	-5.537	-70.223	43.746
19	37.361	6.251	16.677	-5.493	-70.245	43.681
20	37.361	6.251	16.699	-5.471	-70.333	43.681
21	37.383	6.240	16.688	-5.471	-70.311	43.615
22	37.383	6.262	16.699	-5.383	-70.377	43.571
23	37.405	6.262	16.699	-5.383	-70.377	43.571
24	37.405	6.251	16.699	-5.405	-70.399	43.549
25	37.394	6.262	16.710	-5.383	-70.399	43.549
26	37.394	6.251	16.710	-5.427	-70.399	43.549
27	37.394	6.251	16.710	-5.383	-70.399	43.527
28	37.394	6.240	16.699	-5.405	-70.377	43.527
29	37.394	6.240	16.699	-5.405	-70.377	43.527
30	37.394	6.240	16.699	-5.405	-70.377	43.549
31	37.394	6.251	16.699	-5.383	-70.355	43.527
32	37.383	6.240	16.699	-5.339	-70.333	43.505
33	37.372	6.240	16.666	-5.317	-70.201	43.439
34	37.338	6.206	16.654	-5.339	-70.091	43.439
35	37.327	6.206	16.643	-5.317	-70.003	43.395
36	37.283	6.184	16.632	-5.317	-69.871	43.395
37	37.249	6.184	16.610	-5.207	-69.761	43.285
38	37.249	6.184	16.599	-5.229	-69.696	43.307
39	37.238	6.184	16.576	-5.163	-69.630	43.219
40	37.227	6.184	16.565	-5.032	-69.564	43.109

Appendix J: Levenberg-Marquardt Algorithm

Levenberg-Marquardt algorithm is an iterative algorithm used for curve fitting. The algorithm minimizes the sum of the squares of the deviations as presented in equation (J.1)

$$S(p) = \sum_{i=1}^{n} \left[y_i - f(x_i \mid p) \right]^2 \qquad \text{(J.1)}$$

where

p is the vector of parameters to be determined

f is the fitting function

n is the data size

x_i is the data to be fitted to y_i so that $f(x_i \mid p) = y_i$

Let J be the *Jacobian* of $f(x \mid p)$, for every iteration step the parameter vector p is updated as in equation (J.2)

$$p = p + q \qquad \text{(J.2)}$$

and q is computed by linearization approximation of equation (J.3)

$$f(p+q) \approx f(p) + Jq \qquad \text{(J.3)}$$

When the equation (J.1) is at the minimum, the gradient $\frac{\partial \left[f(x \mid p+q)^2 \right]}{\partial q} = 0$,then, by using the approximation presented in (J.3), q can be obtained from the equations (J.4)

$$\begin{cases} \dfrac{d\left[f(x \mid p) + Jq \right]^2}{dq} = 0 \\ \Leftrightarrow \dfrac{d\left[f(x \mid p)^2 + 2 f(x \mid p) Jq + (Jq)^2 \right]}{dq} = 0 \\ \Leftrightarrow f(x \mid p) J^T + \left(J^T Jq \right) = 0 \\ \Rightarrow f(x \mid p) J^T + \left(J^T J + \lambda I \right) q = 0 \end{cases} \qquad \text{(J.4)}$$

where I is the identity matrix and the positive λ is the dumping factor, the more is λ the *Levenberg-Marquardt algorithm* will behave like the *steepest descent process*. On the other hand, the smaller λ is, the more the *Levenberg-Marquardt algorithm* will behave like the *Gauss-Newton process*.