Universidad Politécnica de Cartagena
Departamento de Expresión Gráfica

Proyecto Fin de Carrera

3D Detection of People.

Autor: Antonio Patricio Bernal Rodríguez.
Director: Lucas Roca Nieto.
Titulación: Ingeniería en Automática y Electrónica Industrial.

21. Abril 2009

CONTENTS

1. Introduction.

1.1. Motivation.

The Nexus-project researches methods and concepts to support space and context for system-applications. An important requirement for this kind of systems is multisensor-integration and resolution of inconsistencies in the environment model.

This thesis is based on the movement detection of people in images from multiple cameras, and their location in the three-dimensional space, from the positions where the movement occurs in the different views. The accuracy of the three-dimensional location will be investigated in this thesis and how this accuracy is influenced when the cameras are connected in a network with a low frame rate.

1.2. Abstract.

The aim of this work is to obtain the three-dimensional location of people in a scene using the information provided by several cameras; these cameras must have a range of view in common. The place chosen to record the videos for this work is the *Nexus Lab* in the *Universität Stuttgärt*.

In this work, it is possible to distinguish two main tasks:

- The first task is to obtain a model of the camera which let us turn the two-dimensional information from the images taken from the cameras, in three-dimensional information in a known reference-frame; this topic is studied in chapter 2.

- The second task is to obtain relevant information from the images of each camera, that is, the location of the people, which appear in an image scene, must be known to compute their three-dimensional location. This information will be obtained by reviewing different types of methods. This topic is described in chapter 3.

Finally, the three-dimensional location of the people in the room has been accomplished by integrating the two-dimensional information obtained from the images of each camera with the model of the cameras and knowing their locations in the room. In order to quantify the measurement error of the system, several video tests have been recorded. The error analysis is accomplished in chapter 4.

1.3. Hardware.

The sensors used in this work, to obtain the information from the real world, are video cameras. The system has been thought out to support an undetermined number of cameras, but it has only been tested with three cameras. Two of them are IP-cameras; these cameras use Internet Protocol to transmit image data and control signals over a Fast Ethernet link. The IP-cameras used to record the images are from the German company Mobotix AG. The model of both IP-Cameras is MOBOTIX M1-IT, and their features are shown in the Table 1.3.1.

Figure 1.3.1. MOBOTIX M1-IT.

Product Type	MOBOTIX M1
Model	IT
Features	Mono, Audio, Microphone
Camera Code	006AC1
HARDWARE	
Board Revision	1.4c
Processor Type	Intel sa1110 rev 9
Processor Speed	206.00 MHz
ROM Size	8 MB
RAM Size	64 MB

Table 1.3.1. MOBOTIX M1-IT features.

The Company Mobotix AG offered a lot of software and development tools to control the camera's different resources, to record video sequences, to set alarms and to control the audio and microphones. The maximum frame rate, in the mode used in this work, is 25 fps for each camera, which is a higher rate than the one used to record videos test (2 to 10 fps).

The location of the MOBOTIX-M1-IT cameras in the Nexus Lab is shown in the Figure 1.3.2.

Figure 1.3.2. Location of the MOBOTIX-M1-IT in the Nexus Lab.

2. Multiple View Geometry.

This chapter describes the geometry, which lets us link groups of correspondent points in N-images, in a scene with points from the three-dimensional world. In order to achieve this aim, the chapter is divided in five parts. The first part describes the single view geometry, and the perspective or pinhole camera model. This is the foundation for the following parts. The second part describes the geometry for a stereo system; this geometry is called epipolar geometry. The third part describes the trifocal tensor, which plays the same role in three views, as the one played by the fundamental matrix in two views. The fourth part is about the triangulation problem in a stereo system. The fifth and last part describes how the cameras were modeled in our system.

2.1. Single View Geometry.

The aim of this chapter is to link the position of scene points with their corresponding image points. To do this, we need to model the geometric projection performed by the sensor. A camera is a mapping between the 3D world and a 2D image. The most common geometric model of a camera is the perspective or pinhole model. This model is very simple but very useful too.

Pinhole camera model.

Let consider the central projection of points in space onto a plan. Let the centre of projection be the origin of a coordinate system (*camera centre or optical centre*), and consider the plane z=f, (*f is focal length*) which is called the *image plane* or *focal plane*. A point in the space X=(X,Y,Z)' is mapped on the image plane where a line joining the point X to centre of projection meets the image plane.

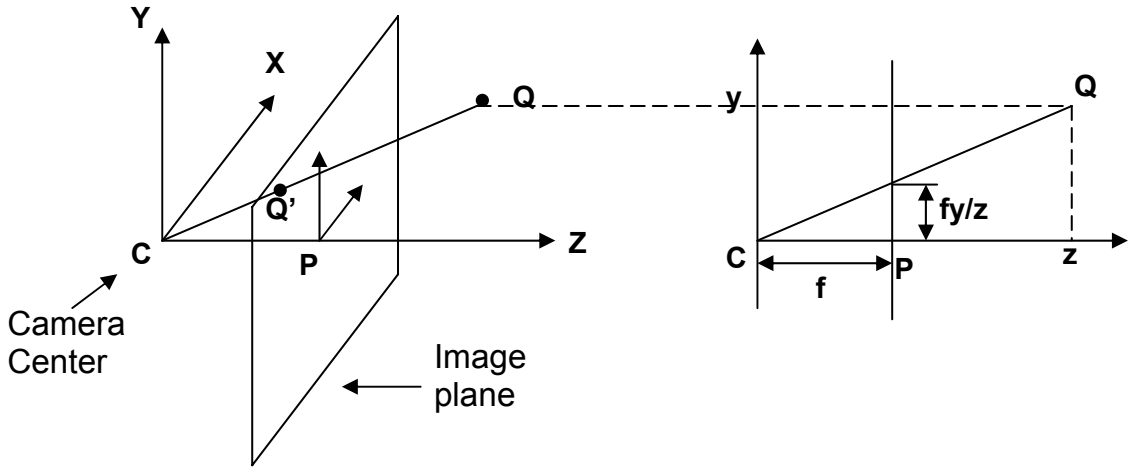


Figure. 2.1. Pinhole camera geometry.

By similar triangles, it is easy to compute that the point Q(X,Y,Z) is mapped to the point Q' (f X/Z, f Y/Z, f)' on the image plane. Using homogeneous

coordinates for the world and image points, then the projection can be expressed as a linear mapping between their homogeneous coordinates.

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

(2.1.1)

Now we have a projection, which mapped 3D points onto a plane, but the camera reference frame must be located in respect to another known reference frame. Then the coordinates of the image points in the camera reference frame can be obtained from pixel coordinates. For this we must know the camera's characteristics, which are called extrinsic and intrinsic parameters.

The **extrinsic parameters** are the parameters that define the location and orientation of the camera reference frame that corresponds to a known reference frame.

The **intrinsic parameters** are the parameters necessary to link the pixel coordinates of an image point, with the corresponding coordinates in the camera reference frame.

Extrinsic parameters.

The camera reference frame, which has been introduced for the purpose of writing the fundamental equations of perspective projection, is often unknown. The extrinsic parameters let us identify uniquely the transformation between the unknown camera reference frame with a known reference frame. To do this, we can use a translation vector **T**, which links the position of the two reference frames, and a rotation matrix **R,** that brings corresponding axes of the two reference frames onto each other. Given a certain point P expressed in the world and camera reference-frames, the relation between $\mathbf{P_w}$ and **Pc** can be written as:

$$P_c = R(P_w - T)$$

(2.1.2)

The camera extrinsic parameters, which specify the transformation between the camera and the world reference frame, are the translation vector **T**, and the three parameters which define the rotation matrix **R**,

Intrinsic parameters.

For a pinhole camera model there are three sets of intrinsic parameters needed, which specify respectively:

- The perspective projection, for which the only parameter is the focal length (f).
- The transformation between camera frames coordinates and pixel coordinates.
- The geometric distortion introduced by the optics.

To find the second set of intrinsic parameters, we must link the coordinates of an image point in pixel units (x_{im}, y_{im}) with the coordinates of the same point in the camera reference frame(x, y). Then, the relation between camera reference frame and image reference frame (in pixels) can be written as follows:

$$x = -(x_{im} - o_x)s_x$$
$$y = -(y_{im} - o_y)s_y$$

(2.1.3)

Where (o_x, o_y) are the pixel coordinates of the image center, and (s_x, s_y) are the effective size of the pixel.

The distortion introduced by the optics can be modeled as simple radial distortions, according to these relations:

$$x = x_d(1 + k_1 r^2 + k_2 r^4)$$
$$y = y_d(1 + k_1 r^2 + k_2 r^4)$$

(2.1.4)

Where $r^2 = x_d^2 + y_d^2$, and x_d, y_d are coordinates of the distorted points.

The intrinsic parameters are defined as the focal length ***f***, the location of the image center in pixel coordinates **(o_x, o_y)**, the effective pixel size in the horizontal and vertical direction**(s_x, s_y)**, and if required, the radial distortion coefficients, **k_1** and **k_2**.

To determine the value of the extrinsic and intrinsic parameters of the camera, we must calibrate the camera.

2.2. Two-View Geometry.

Stereo vision refers to the ability to infer information on the 3D structure and on the distance of a scene from two images taken from different viewpoints. A stereo system must solve two problems:

- The correspondence problem, which consists in determining which item in the first view, corresponds to which item in the second view.
- The reconstruction problem, given a number of corresponding parts of the left and right image, and information on the geometry of the stereo system. What can we say about the 3D location and structure of the observed objects?.

Epipolar Geometry describes the geometry of a stereo system. This geometry enables us to clarify which information is needed in order to perform the search for corresponding elements along image lines. The Epipolar Geometry is shown in Figure 2.2. The Figure 2.2. shows two pinhole cameras, their projection centers, O_l and O_r, and image planes, π_l and π_r. The focal lengths are denoted by f_l and f_r. The vector $P_l=(X_l, Y_l, Z_l)'$ and $P_r=(X_r, Y_r, Z_r)'$ refer to the same 3D point, P. The vectors $p_l=(x_l, y_l, z_l)'$ and $p_r=(x_r, y_r, z_r)'$ refer to the projections of P onto the left and right image plane respectively.

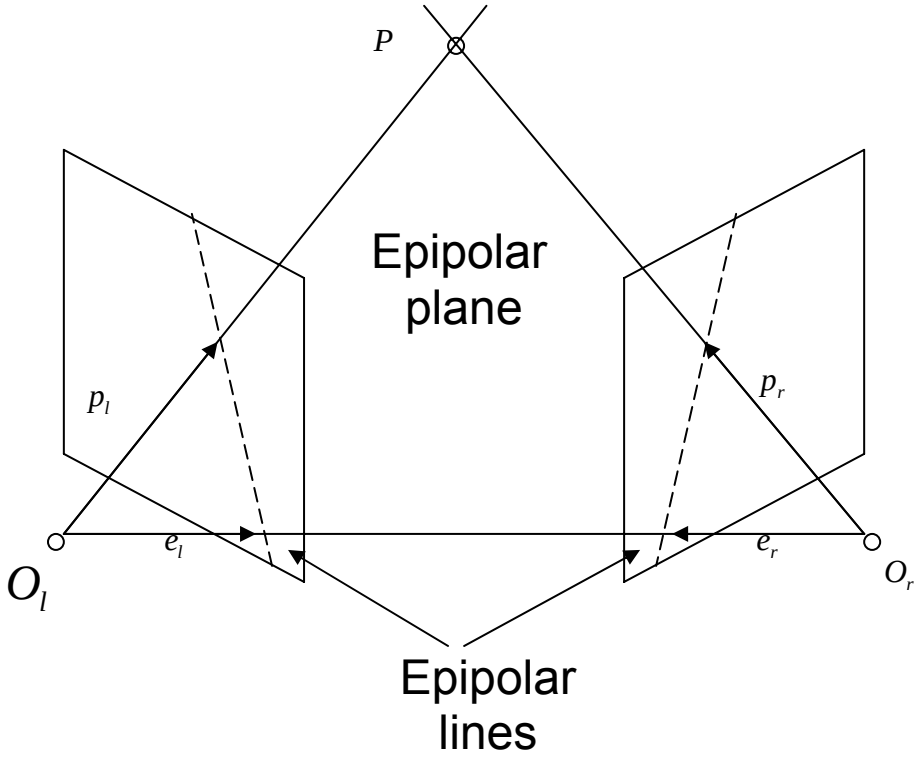


Figure 2.2. epipolar geometry.

The reference frames of the left and right cameras are related through the extrinsic parameters. These define a rigid transformation in 3D space, defined

by a translation vector **T= (O_l – O_r),** and a rotation vector **R**. Given a point **P** in the space, the relation between P_l and P_r is therefore:

$$P_r = R(P_l - T)$$

(2.2.1)

The name Epipolar Geometry is used because the points where the line through the centers of projection intersects the image planes are called epipoles.

The relation between a point in 3D space and its projections is described in vector form,

$$p_l = (f_l / Z_l)P_l$$
$$p_r = (f_r / Z_r)P_r$$

(2.2.2)

The practical importance of epipolar geometry comes from the fact that the plane identified by **P**, O_l, and O_r, is called epipolar plane. It intersects each image in a line, called epepolar line. This establishes a map between points in one image and lines in the other image.

Essential Matrix.

The equation of the epipolar plane through **P** can be written as:

$$(P_l - T)^T T \times P_l = 0$$

(2.2.3)

or using (2.2.1):

$$(R^T P_r)^T T \times P_l = 0$$

(2.2.4)

Writing the vector product as a matrix multiplication, we can write:

$$T \times P_l = S \cdot P_l$$

(2.2.5)

Where,

$$S = \begin{pmatrix} 0 & -Tz & Ty \\ Tz & 0 & -Tx \\ -Ty & Tx & 0 \end{pmatrix}.$$

So,

$$P_r^{\,T} \cdot E \cdot P_l = 0$$

(2.2.6)

With,

$$E = R \cdot S$$

(2.2.7)

Using (2.2.2), equation (2.2.6) can be rewritten as:

$$p_r^{\,T} \cdot E \cdot p_l = 0$$

(2.2.8)

The matrix E is called the essential matrix and establishes a natural link between the epipolar constraint and extrinsic parameters of the stereo system.

Fundamental Matrix.

Let $\mathbf{M_l}$ and $\mathbf{M_r}$ be the matrices of the intrinsic parameters of the left and the right camera. If $\overline{p}_l$ and $\overline{p}_r$ are the points in pixel coordinates corresponding to p_l and p_r in camera coordinates, we can write:

$$p_l = M_l^{\,-1} \overline{p}_l$$

$$p_r = M_r^{\,-1} \overline{p}_r$$

(2.2.9)

By substituting (2.2.9) into (2.2.8),

$$\overline{p}_l^{\,T} \cdot F \cdot \overline{p}_r = 0$$

(2.2.10)

Where,

$$F = (M_r^{\,-1})^T \cdot E \cdot M_l^{\,-1}$$

(2.2.11)

F is called fundamental matrix. The most important difference between essential and fundamental matrices is that the fundamental matrix is defined in terms of pixel coordinates, and the essential matrix in terms of camera coordinates.

2.3. Three-view geometry.

The trifocal tensor plays the same role in three views to the one played by the fundamental matrix in two, it encapsulates all the geometric relations between three view that are independent of scene structure.

There are several ways in which the trifocal tensor may be approached, but our starting point is the incidence relationship of three corresponding lines.

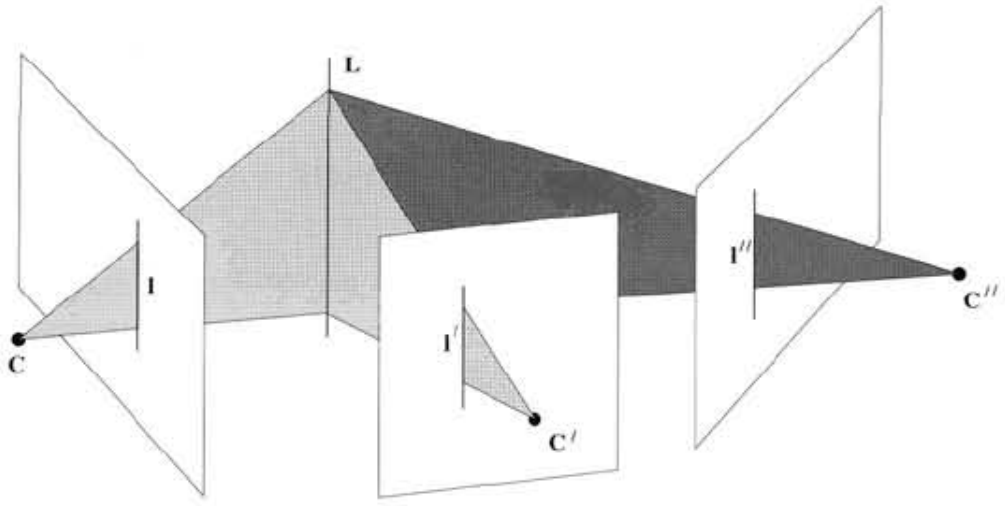


Figure. 2.3.1 Projection of a line in three views. Image taken from [Harley00].

Suppose a line **L** in 3D space is imaged in three views (**l**, **l'**, **l''**), the planes (**π**, **π'**, **π''**) back-projected from the lines in each view must all meet in a single line (**L**) in space.

Let the camera matrices for the three views be **P=[I|0]**, **P'=[A|a$_4$]**, **P''=[B|b$_4$]**, where **A** and **B** are 3 x 3 matrices, and the vectors **a$_i$** and **b$_i$** are the i-th columns of the respective camera matrices i=1,2,3,4.

Each image back-projects to a plane, these planes are:

$$\Pi = P^T \cdot l = \begin{pmatrix} l \\ 0 \end{pmatrix},$$

$$\Pi' = P'^T \cdot l' = \begin{pmatrix} A^T l' \\ a_4^T l' \end{pmatrix},$$

$$\Pi'' = P''^T \cdot l'' = \begin{pmatrix} B^T l'' \\ b_4^T l'' \end{pmatrix}. \quad \textbf{(2.3.1)}$$

Since the three image lines are derived from a single line in space, these three planes must meet in this common line in 3D space. This intersection constraint can be expressed by the requirement that **M= [π, π' , π'']**, has rank 2. This may be seen as follows. Points on the line of intersection may be represented as $\mathbf{X=\alpha X_1+\beta X_2}$, with $\mathbf{X_1}$ and $\mathbf{X_2}$ linearly independent. Such points lie on all three planes and so:

$$\Pi^T \cdot X = \Pi'^T \cdot X = \Pi''^T \cdot X = 0$$

(2.3.2)

It follows that **MX=0**. Consequently **M** has a 2D null-space since $\mathbf{MX_1=0}$, $\mathbf{MX_2=0}$. This intersection constraint induces a relation among the image lines **l**, **l'**, **l''**. Since the rank of **M** is 2, there is a linear dependence between its columns $\mathbf{m_i}$. Denoting:

$$M = [m_1, m_2, m_3]\begin{pmatrix} l & A^T l' B^T l'' \\ 0 & a_4^T l' b_4^T l'' \end{pmatrix}$$

(2.3.3)

The linear may be written $\mathbf{m_1= \alpha m_2+\beta m_3}$ **(1.3.4)**, then:

$$\alpha = k(b_4{}^T l'')\,,\ \beta = -k(a_4{}^T l') \text{ for some scalar } \mathbf{k}.$$

(2.3.5)

Applying this to (2.3.4):

$$l = (b_4{}^T l'')A^T l' - (a_4{}^T l')B^T l'' = (l''^T\ b_4)A^T l' - (l'^T\ a_4)B^T l''$$

(2.3.6)

And the i-th coordinate of **l** may therefore be written as:

$$l_i = l''^T\ (b_4 a_i{}^T)l' - l'^T\ (a_4 b_i{}^T)l'' = l'^T\ (a_i b_4{}^T)l'' - l'^T\ (a_4 b_i{}^T)l'',$$

(2.3.7)

and introducing the notation:

$$T_i = a_i b_4{}^T - a_4 b_i{}^T$$

(2.3.8)

the incidence relation can be written

$$l_i = l'^T\ T_i l''$$

(2.3.9)

The set of three matrices **{T_1, T_2, T_3}** constitute the trifocal tensor in matrix notation. Denoting the ensemble of the three matrices T_i by **[T_1, T_2, T_3]**, relation (2.3.9) may be written as,

$$l^T = l'^T [T_1, T_2, T_3] l''$$

(2.3.10)

where **l'^{T} [T_1, T_2, T_3] l''** is understood to represent the vector **(l'^{T} T_1 l'', l'^{T} T_2 l'', l'^{T} T_3 l'').**

Various linear relationships between lines and points can be deduced in three images involving the trifocal tensor:

- Line-line-line correspondence.

$$l^T = l'^T [T_1, T_2, T_3] l'' \text{ or } l^T = (l'^T [T_1, T_2, T_3] l'')[l]_x = 0^T$$

- Point-line-line correspondence.

$$l'^T (\sum_i x^i T_i) l'' = 0$$

- Point-line point correspondence.

$$l'^T (\sum_i x^i T_i)[x'']_x = 0$$

- Point-point-line correspondence.

$$[x']_x (\sum_i x^i T_i) l'' = 0$$

- Point-point-point correspondence.

$$[x']_x (\sum_i x^i T_i)[x'']_x = 0_{3x3}$$

(* the notation $[x]_x$ expresses the cross product).

2.4. Triangulation.

The reconstruction by triangulation can be performed, if the intrinsic and the extrinsic parameters of a stereo system are known. We can assume to know intrinsic and extrinsic parameters if the geometry of the system does not change with time. Then, the intrinsic and extrinsic parameters for each camera can be estimated through several procedures.

According to Figure 2.1, the point **P**, projected into the pair of corresponding points $\mathbf{p_l}$ and $\mathbf{p_r}$, lies at intersection of the two rays from $\mathbf{O_l}$ through $\mathbf{p_l}$ and from $\mathbf{O_r}$ through $\mathbf{p_r}$ respectively, but the problem is, since parameters and image locations are known only approximately, their intersection can only be estimated as the point of minimum distance between both rays (see Figure 2.4).

Let $a\ \mathbf{p_l}$ be the ray, ***l***, through $\mathbf{O_l}$ and $\mathbf{p_l}$. Let $\mathbf{T} + b\ \mathbf{R}^{\mathbf{T}}\ \mathbf{p_r}$ be the ray, ***r***, through $\mathbf{O_r}$ and $\mathbf{p_r}$. Let ***w*** be an orthogonal vector to both ***l*** and ***r***. So we must determine the midpoint, **P'**, of the segment parallel to ***w*** that joins ***l*** and ***r***, as shown in Figure 2.4.1.

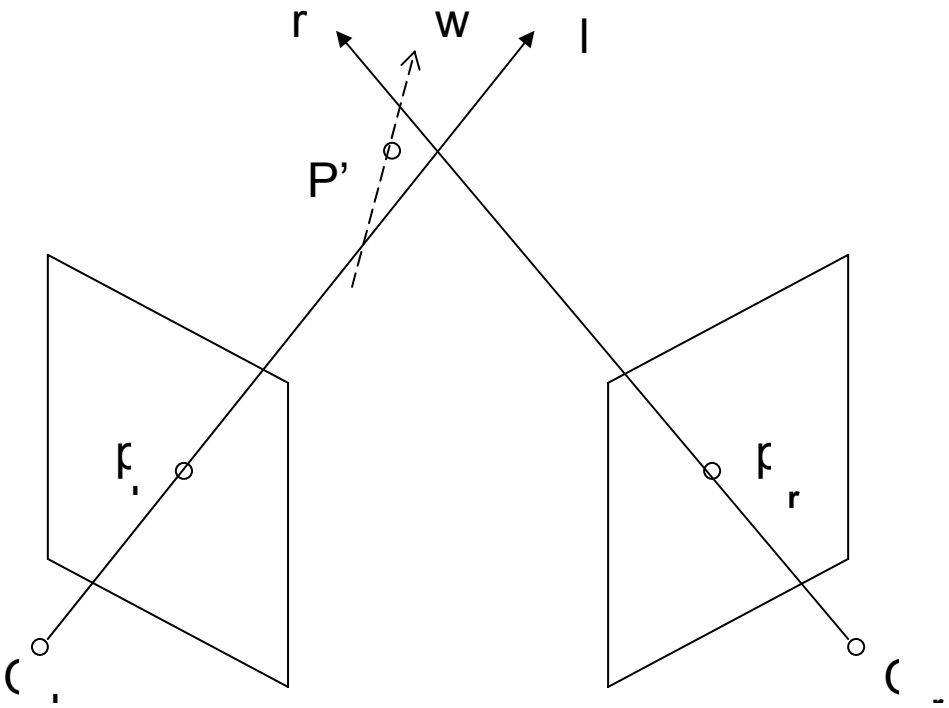


Figure 2.4.1. Triangulation with nonintersecting rays.

The endpoint of the segment $a_0\ \mathbf{p_l}$ and $\mathbf{T}+b_0\ \mathbf{R}^{\mathbf{T}}\ \mathbf{p_r}$, can be computed solving the linear system of equations:

$$a \cdot p_l - T - b \cdot R^T p_r + c(p_l \times R^T p_r) = 0$$

(2.4.1)

Then the midpoint P' can be computed:

$$P' = (a_0 p_l + T + b_0 R^T p) / 2$$

(2.4.2)

In the case of several cameras, a way to compute the three-dimensional point is, compute for each pair of cameras the point **P_i'** and then compute the mean point. Assuming the number of cameras in the system is **n**, there will be **Ncomb** midpoints,

$$Ncomb = \binom{n}{2} = \frac{n!}{2!(n-2)!}$$

(2.4.3)

Then the midpoint can be calculated as,

$$P' = \frac{1}{Ncomb} \sum_{i=1}^{Ncomb} P'_i$$

(2.4.4)

2.5. Camera calibration process.

In section 2.1, the pinhole camera model was described, as well as how the model of the camera was defined by the extrinsic and intrinsic parameters. In this section the computation of these parameters method is described.

The algorithm OpenCV pretends that there is no distortion in the camera, to compute the focal lengths and offsets using Zhang's method [Zhang00]. Once these parameters have been computed, camera distortion parameters are computed using Brown's method [Brown71]. This OpenCV algorithm works well with planar patterns, the most common is a chessboard. The main inputs for this algorithm are:

- The three-dimensional locations of the points in the frame defined by the pattern. If we use a chessboard all the z-coordinates can be set to zero, and the x- and y- coordinates can be easily computed knowing the length of the side of the square.

- The coordinates in of the points (in pixels) in the image. OpenCV implement a function to get the position of the corners of a chessboard in an image (see Figure 2.5.1).

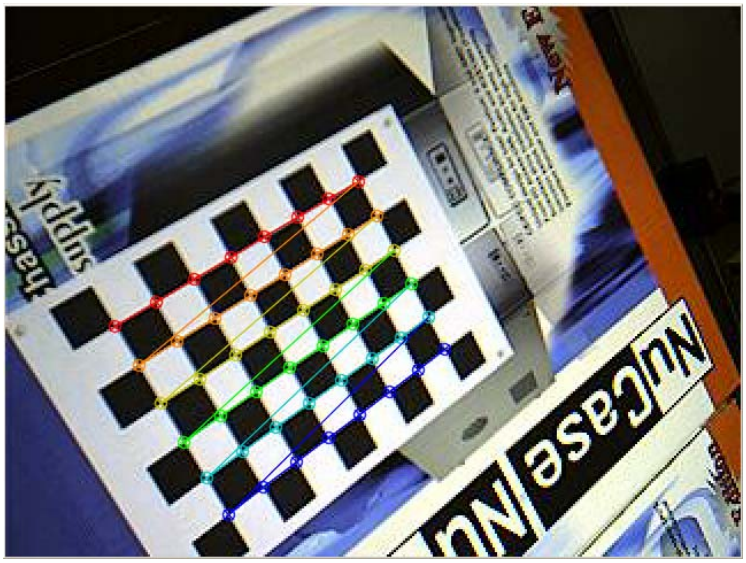


Figure 2.5.1 Chessboard corner detection.

With this information this algorithm computes, the intrinsic matrix of the camera, the distortion coefficients (for our application will be supposed that there is no distortion in the camera), and the rotation and translation vectors for each image of the pattern.

The way the openCV algorithm provides the extrinsic parameters is,

$$P_c = R\ P_w + T$$

(2.5.1)

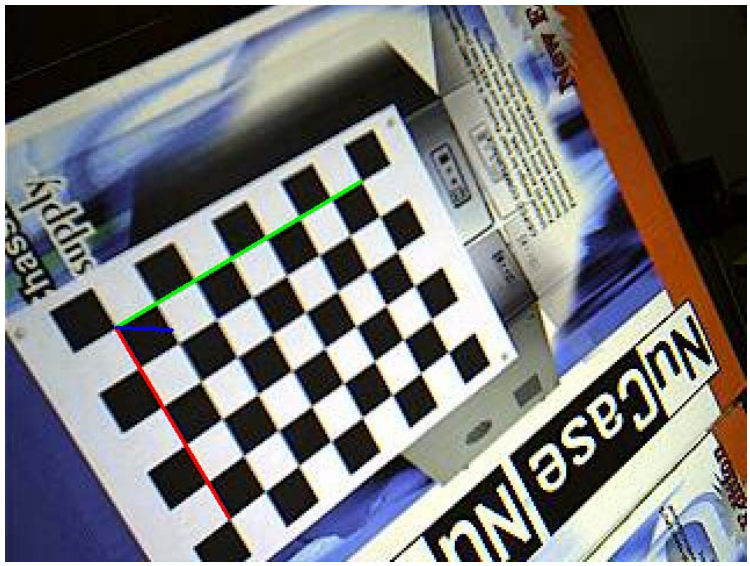


Figure 2.5.2. chessboard-reference-frame, (red→x-axis, green→y-axis, blue→z-axis)

The problem now is how to relate the different frames-references of the cameras. When the cameras are close enough, and it is possible to take images of a pattern from the cameras, without changing the location of the pattern, it is possible to apply different techniques to get the fundamental matrices of each pair of cameras, but this is not our case.

In order to compute the fundamental matrices of each pair of cameras, let define a world-reference-frame ('w'), two camera-reference-frames (left 'l' and right 'r') and two chessboard-reference-frames (left-chessboard 'lc' and right-chessboard 'rc'). Then we can write the expression (2.5.1) as,

$$P_r = R_{r\_cr} P_{cr} + T_{r\_cr}$$
$$P_l = R_{l\_cl} P_{cl} + T_{l\_cl} \quad \textbf{(2.5.2)}$$

The rotation and translation matrices, which relate the chessboard-reference-frames and the world-reference-frame, can be measured. The measurements related to the translation matrices were accomplished with a meter and the rotation matrix can be easily computed if the chessboard-reference-frames are chosen carefully. Then the expressions which relate points of the chessboard-reference-frames and world-reference-frame can be written as,

$$P_w = R_{w\_cr} P_{cr} + T_{w\_cr}$$
$$P_w = R_{w\_cl} P_{cl} + T_{w\_cl} \quad \textbf{(2.5.3)}$$

Isolating $\mathbf{P_{cr}}$ and $\mathbf{P_{cl}}$ from (2.5.3) and substituting them in (2.5.2), we obtain,

$$P_r = R_{r\_cr} R_{w\_cr}^T (P_w - T_{w\_cr}) + T_{r\_cr}$$
$$P_l = R_{l\_cl} R_{w\_cl}^T (P_w - T_{w\_cl}) + T_{l\_cl}$$

(2.5.4)

Isolating $\mathbf{P_w}$ from one of the equation above and substituting it in the other,

$$P_r = R_{r\_cr} R_{w\_cr}^T (((R_{l\_cl} R_{w\_cl}^T)^T (P_l - T_{l\_cl})) + T_{w\_cl} - T_{w\_cr}) + T_{r\_cr}$$

(2.5.5)

Comparing this expression with the expression (2.2.1),

$$P_r = R_{rl}(P_l - T_{lr}) \Rightarrow P_l = R_{rl}^T P_r + T_{lr}$$
$$R = R_{r\_cr} R_{w\_cr}^T (R_{l\_cl} R_{w\_cl}^T)^T = R_{rl}$$
$$T = R_{r\_cr} R_{w\_cr}^T (-(R_{l\_cl} R_{w\_cl}^T)^T T_{l\_cl} + T_{w\_cl} - T_{w\_cr}) + T_{r\_cr} = T_{lr}$$

(2.5.5)

This way, it is possible to compute the essential and fundamental matrices, for each pair of cameras of our system, as was explained in the section 2.2.

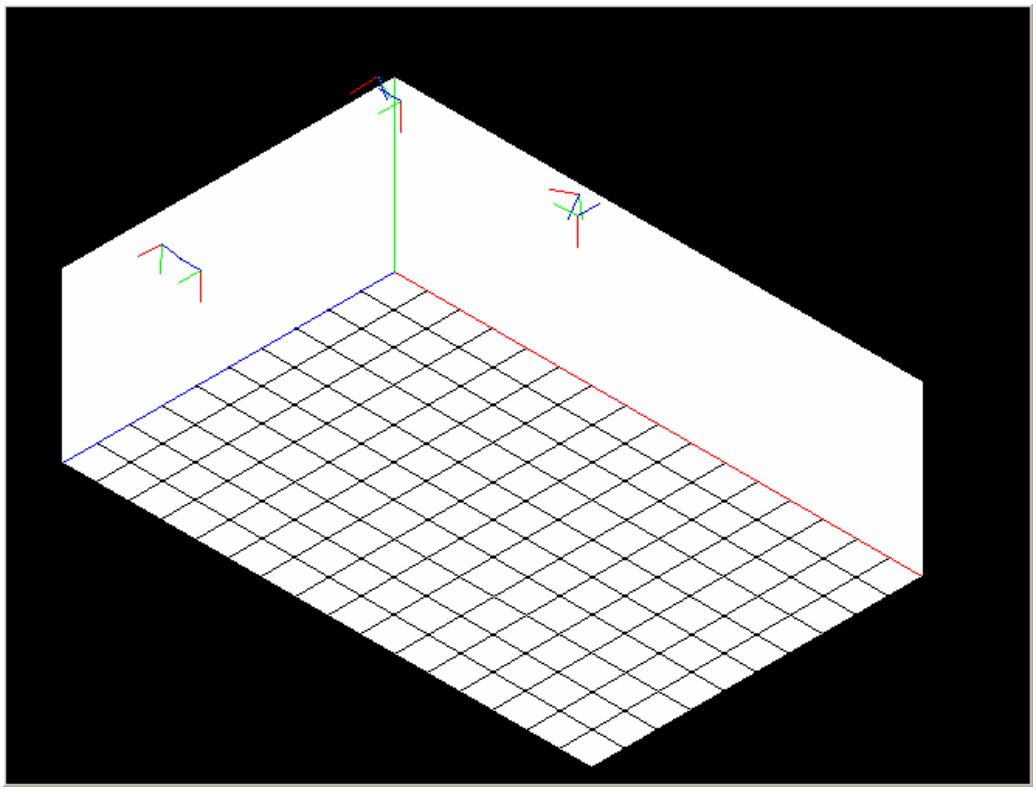


Figure 2.5.3. Isometric projection of the room with the world-reference-frame, cameras-reference-frames and chessboard-reference-frames.

3. Motion detection and tracking of people.

In this chapter several ways to get the position of persons are described, as well as objects in movement in a room and how to track them along a certain period of time.
The aim of the following algorithms is to provide the pixel coordinates of the objects of interest. Once this is done, in the different videos, recorded simultaneously from the cameras, and assuming the intrinsic and extrinsic parameters of the cameras are known, it is possible to compute the three-dimensional location of the people or objects in movement in the room.
OpenCV implements a lot of functions which are very useful to achieve this aim.

3.1. Motion templates.

Motion templates were invented in the MIT Media Lab by Bobick and Davis [Bobick96]. This last author made further development with Gary R. Bradsky [Bradski00]. Motion templates can be used to:

- Determine the current pose of the objects in movement.
- Segment and measure the motions induced by the objects in a video scene. These segmented regions are not “motion blobs”, instead they are motion regions naturally connected to the moving parts of the object of interest.

For the purpose of this work the first of the use is the most interesting.

Using motion templates requires a silhouette of an object. There are many different ways to get a silhouette which could work in other situations, but they are based in assumptions which are not true in our case, some of them are mentioned below.

- Using chroma keying, this is useful if there is a known background colour, then you can take as foreground anything different in the image.
- A similar way is to learn the background model, using for instance averaging background method and extract foreground as silhouettes.
- It is also possible to use active techniques like infrared technology or thermal images.

A simple and effective way to get these silhouettes when stationary cameras are used is to employ frame-to-frame differencing, this will provide the moving edges of objects, which is enough to make motion templates work. Differencing several images and thresholding them we get a mask where pixels, set to non-zero value, represent the points where the motion occurs. Let consider a single channel floating-point matrix with the same dimensions as the mask, which elements are set to the time stamp of the system where the mask pixels have non-zero value, and they are set to zero if there has been no motion detected

for a long time (*duration)*. Otherwise they keep their values. This matrix is known as Motion History Image (MHI). This can be written:

$$MHI(x,y) = \begin{cases} timestamp & if \quad silhouette(x,y) != 0 \\ 0 & if \quad silhouette(x,y) = 0 \ and \ MHI(x,y) < timestamp - duration \\ MHI(x,y) & otherwhise \end{cases}$$

(3.1.1)

The OpenCV function which makes all this is *cvUpdateMotionHistory()*.

Once Motion History Image is computed, it is possible to estimate the direction and the magnitude of the movement taking the gradient of the MHI image. Taking the gradient of the MHI, we would get direction vectors pointing in the direction of the movement. Gradients of the MHI can be calculated efficiently with Sobel filters. Some gradients will be too large due to the edges of the silhouettes, but since the time stamp duration is known, it is possible to reject these invalid gradients. The OpenCV function which computes the MHI gradient is *cvCalcMotionGradient()*.

The overall direction of motion is computed as the vector sum of the valid gradient directions. This is done by the function *cvCalcGlobalOrientation()*.

By isolating regions of the MHI, it is possible to determine the local motion within these regions. The MHI image is scanned for current silhouette regions. When a region marked with the most current time stamp is found, the region's perimeter is searched for sufficiently recent motion (recent silhouettes) just outside its perimeter. When such motion is found, a downward stepping flood fill is performed to isolate the local region of motion, so we get the current location of the object of interest. Once found, we can calculate local motion gradient direction in the region, then remove that region, and repeat the process until all regions are found. The OpenCV function that isolates and computes local motion is *CvSegmentMotion()*.

Using all these functions we can get a bounding box of the different regions where movements were found in the images of the multiple cameras of the system. This way we can select a point or several points within this box and search for possible correspondent points in the other images. There is also available information about direction and magnitude of these movements but they have not been taken into account.

Figure 3.1.1 Image of a person in movement in the nexus lab being tracked by Motion Templates Algorithm.

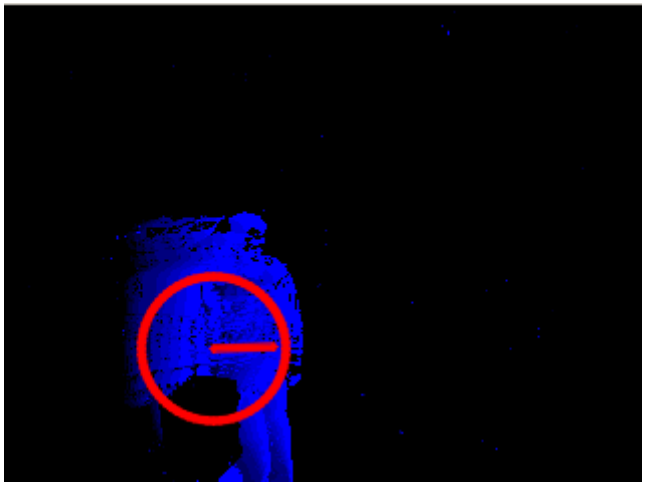


Figure 3.1.2 Motion History Image of the Figure 3.1.1., the circle with the line represent the direction and the location of the movement.

3.2. Optical flow.

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and the scene. Optical flow techniques such as motion detection, object segmentation, time-to-collision and focus of expansion calculations, motion compensated encoding, and stereo disparity measurement use the motion of the objects surfaces, and edges. In this work, different optical flow techniques have been studied in order to detect and track objects (people) in movement in a room.

Optical flow techniques can be classified in:

- ***Dense optical flow techniques.*** OpenCV functions implement two of this kind of techniques *Horn Schunck Method* and *Bock Matching*. Dense methods present a high computational cost, because they try to find the displacement or velocity of each pixel between the previous and the current frame.

- ***Sparse optical flow techniques.*** OpenCV functions implement optical flow Lucas-Kanade method and its modification, which works with image pyramids. This kind of algorithms tracks a given subset of pixels between frames.

Horn Schunck Method.

The Horn Schunck method [Horn81] assumes that:

- The surface being imaged is flat, to avoid variations in brightness.

- Incident illumination is uniform across the surface, so brightness at a point in the image is then proportional to the reflectance of the surface.

- Reflectance varies smoothly and has no spatial discontinuities. This condition assures us that the image brightness is differentiable.

Defining the brightness of a pixel in the image at time *t* as *(E(x,y,t))*, according to the previous assumptions, this value does not change with time. This can be written:

$$\frac{dE(x,y,t)}{dt} = 0 \Rightarrow \frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$$

(3.2.1.)

This is a single linear equation in the two unknowns (*dx/dt, dy/dt*), and it is solved introducing the local flow velocity constraint. However it is impossible to determine the component of the movement in the direction of the iso-brightness contours, at right angles to the brightness gradient, so a new constraint it is needed, the smoothness constraint.

One way to express the smoothness constraint of the optical flow velocity is to minimize the sum of the squares of the Laplacians of the x and y components of the flow. The solution of these equations devised by Horn and Schunck is:

$$\frac{\partial^2 u}{\partial x^2} - \frac{1}{\alpha} E_x (E_x u + E_y v + E_t) = 0$$
$$\frac{\partial^2 v}{\partial y^2} - \frac{1}{\alpha} E_y (E_x u + E_y v + E_t) = 0$$

(3.2.2.)

In this expression,

$$u = \frac{dx}{dt}, v = \frac{dy}{dt}, E_x = \frac{\partial E}{\partial x}, E_y = \frac{\partial E}{\partial y}, E_t = \frac{\partial E}{\partial t}$$

And α is a constant weighting coefficient known as the regularization constant.

Block Matching

Block Matching algorithms, try to match pixels in a previous frame with pixels in the current frame. These algorithms divide the image into blocks, which usually overlap, and compute the motion of these blocks. The implementation in OpenCV uses a spiral search that works out from the location of the original block in the previous frame, and compares the candidate new blocks with the original. This comparison is a sum of absolute differences of the pixels. If a good enough match is found, the search is terminated.

Both of these Dense Optical Flow Techniques were considered not useful for our purpose, because the information they provide us was not as interesting as the information, which the Sparse Optical Flow Techniques could offer. In addition, and as was mentioned before, these algorithms present a high computational cost.

Lucas-Kanade Algorithm

The Lucas–Kanade method [Lucas81] is a two-frame differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade. It introduces an additional term to the optical flow, by assuming the flow to be constant in a local neighborhood around the central pixel under consideration at any given time.

The Lucas-Kanade algorithm is based on the following assumptions:

- *Brightness constancy*. A pixel from the image of an object in the scene does not change in appearance as it moves from frame to frame.
- *Temporal persistence or "small movements"*. The image motion of a surface patch changes slowly in time. Then the truthfulness of this

assumption will depend on the frame rate of the camera and the speed of the object to be tracked.

- *Spatial coherence*. Neighbouring points in a scene belong to the same surface, have similar motion, and project to nearby points on the image plane.

The first assumption for the one-dimensional problem can be written as follows,

$$I(x,t) = cte \Rightarrow \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial t} = 0$$

(3.2.1.)

Or,

$$I_x u + I_t = 0 \Rightarrow u = -\frac{I_t}{I_x}$$

(3.2.2.)

According to the previous expression, in the one-dimensional problem, we can compute the velocity of one edge as the relation between the variation of the brightness with time and the variation of the brightness along the x-axis, as shown in Figure 3.2.1.

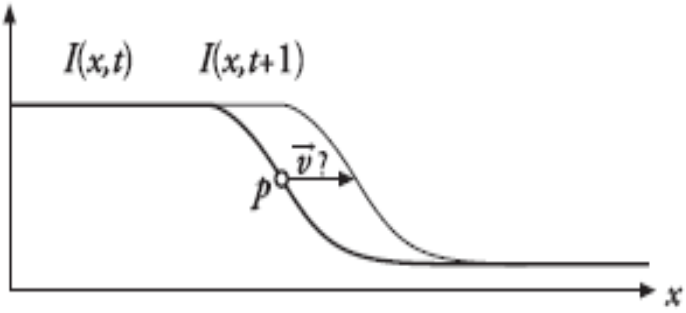


Figure 3.2.1., shows the transition from a high brightness level to a low brightness level along the x-axis. Image taken from [Lucas81].

Because the brightness assumption is not totally true, and the frame rate of the cameras is not as high as desired, in relation to the motion, the velocity computed in the previous way is not exact. However, it is possible to improve the estimation of the velocity by iterating the result, using the Newton's Numeric Method. If the first estimation of the velocity is not good enough, the iteration method will diverge.

The generalization for the two-dimensional case requires adding the y-coordinate to the expression 3.2.2.,

$$I_x u + I_y v + I_t = 0$$

(3.2.3.)

This is a single equation, and there are two unknowns, so the problem is underconstrained, and we can not obtain a unique solution from it. It is only

possible to solve the motion component which is perpendicular to the line described by the flow equation 3.2.3., as it is shown in the Figure 3.2.2.

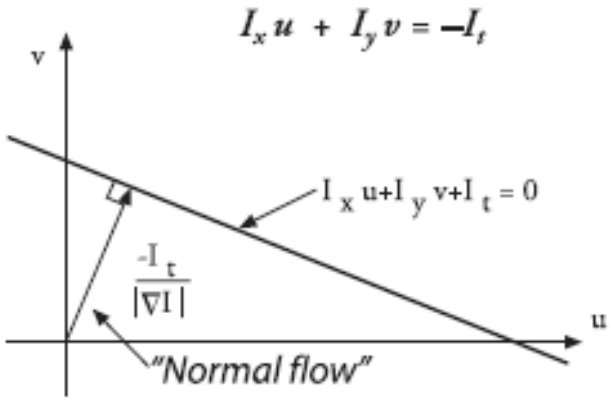


Figure 3.2.2., shows the normal flow of a given pixel in two-dimension. Image taken from [Lucas81].

The problem of the normal optical flow arises when within the window, which is being used to measure motion, is only an edge, not a corner. This is insufficient to determine how the object is moving. This problem is known as the aperture problem and it is illustrated in Figure 3.2.3.

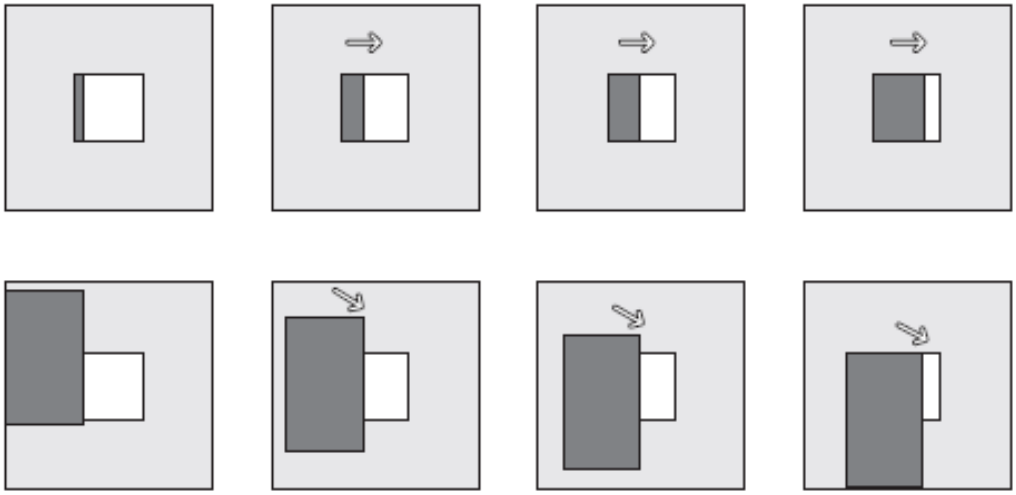


Figure 3.2.3., illustrates the aperture problem. Image taken from [Bradski08].

Using the *Spatial coherence* assumption, the problem can be solved just by taking the surrounding pixels to set up a system of equations, which will be overconstrained and can be solved by least-square minimization. The tracking window should be centered in a corner region from an image, to assure the system of equations can be solved.

Lucas-Kanade algorithm does not work very well when large windows are taken to track large motions, because large windows often break the coherent motion assumption. This problem is solved by the Pyramid Lucas-Kanade Algorithm [Jean99], which tracks first over large spatial scales, using image pyramid, and

then refines the initial motion velocity assumption by working in lower levels of the image pyramid.

The Pyramid Lucas-Kanade algorithm was attempted to use in the problem of people detection in a room without success. It was thought to find points with adequate features to track in the image (*using the OpenCV function cvGoodFeaturesToTrack()*) and track them (*using the OpenCV function cvCalcOpticalFlowPyrLK()*). After that, we could find those with large enough movements between frames and estimate their three-dimensional position.
These points could produce good results in the three-dimensional reconstruction, if we could match these points with their correspondent points in the images captured by the other cameras. However this is not possible in our case. The main assumption in the stereo matching algorithm is, that the most of the scene points must be visible in both views. Since this assumption is not true in our case (*this is shown in Figure 3.2.4.*), we have a group of points being tracked by the optical flow algorithm, but we can not use this information to locate the movement in the three-dimensional world.

Figure 3.2.4.shows two images recorded simultaneously.

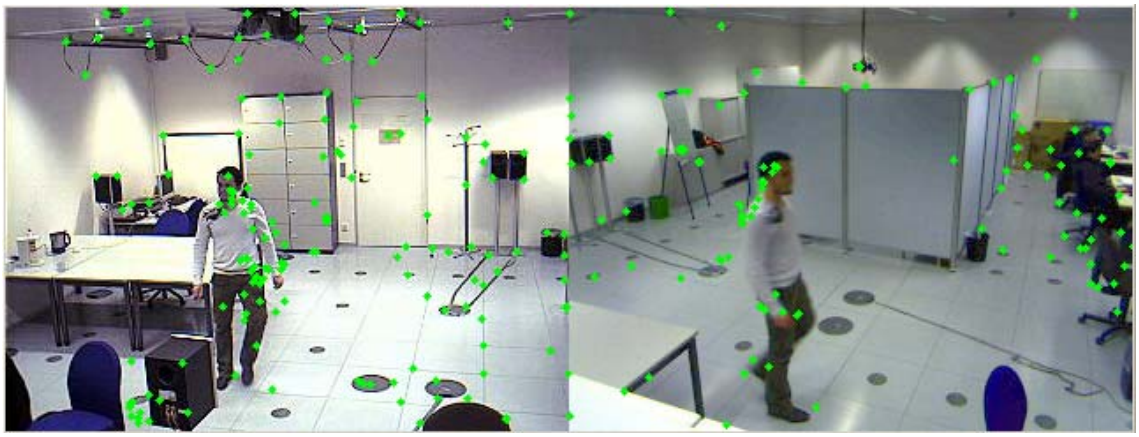


Figure 3.2.5. Pyramid Lucas-Kanade demo image.

3.3. Pattern recognition.

The methods described in previous sections provide the two-dimensional position of moving objects in a scene, in our case in a room. We can not distinguish if the objects in movement are boxes, chairs, dogs, or people.
For this reason, it was attempted to use different kind of techniques which could provide the position of a human body, although it stays stationary.
The OpenCV functions implement several pattern recognition algorithms, which could decide if there is a human body in a scene or not, and let us know its position. The most interesting of these algorithms is the Haar-Classifier algorithm developed by Paul Viola and Michael Jones, commonly known as *Viola-Jones detector* [Viola01]. This algorithm was later extended by Rainer Lienhart and Jochen Mayd [Lienhart02].

The Viola-Jones detector was thought as a new algorithm for robust and extremely rapid object detection. Nevertheless it was motivated by the face detection task. In other detection systems, auxiliary information, such as image differences in video sequences, or pixel color in color images, has been used to achieve high frame rates. This system achieves high frame rates by working only with the information present in a single gray scale image. Nonetheless these alternative sources of information can also be integrated to achieve higher frame rates.

There are four main innovation features of the Viola-Jones detector,

- *Haar-like input features*. They consist in a threshold applied to sums and differences of rectangular image regions.

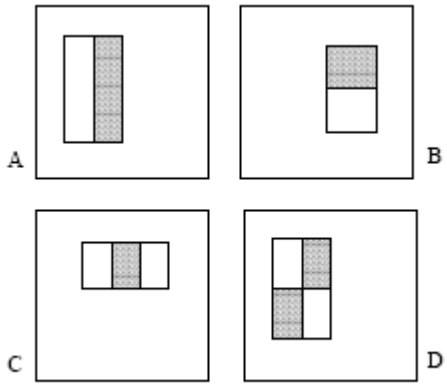


Figure 3.3.1. Rectangle features used by Viola-Jones algorithm. Image taken from [Viola01].

- *Integral image representation*. The integral image representation of one image in a given pixel is computed as the sum of the pixels above and to the left of that pixel, this can be written,

$$ii(x,y) = \sum_{x' \le x} \sum_{y' \le y} i(x',y')$$

(3.3.1)

Where *'ii'* is the integral image and *'i'* is the original image.
The integral image can be computed from an image with a few operations per pixel, by using the following pair of recurrences,

$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

(3.3.2)

Once computed, the Haar-like features can be computed at any scale or location in constant time.

- New method for constructing a classifier, by selecting a small number of important features using AdaBoost. In order to ensure fast classification, the learning process must exclude a large majority of the available features, and focus on a small set of critical features. The feature selection is achieved by a modification of the AdaBoost procedure, which consists in constraining the weak learner, so that each weak classifier returned can depend on only a single feature.

- Novel method for combining successively more complex classifiers in a cascade structure, which dramatically increases the speed of the detector by focussing attention on promising regions of the image. The key insight is that smaller and therefore more efficient, boosted classifiers can be constructed. They reject many of the negative sub-windows before more complex classifiers are called upon to achieve low false positive rates. The overall form of the detection process is that of a degenerate decision tree, which is called *"cascade"*, see Figure 3.3.2. A positive result from the first classifier triggers the second classifier, and so on. A negative outcome at any point leads to the immediate rejection of the sub-window.

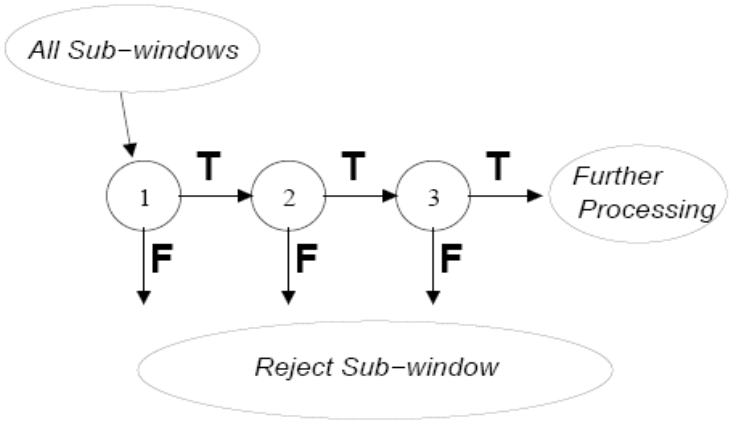


Figure 3.3.2. Cascade detector scheme. Image taken from [Viola01].

Rainer Lienhart and Jochen Mayd extended the Viola-Jones algorithms with two important contributions,

- The basic and over-complete set of Haar-like feature is extended by an efficient set of 45º rotated features, which add additional domain-knowledge to the learning framework and which is otherwise hard to learn.

- Novel post-optimization procedure for a given boosted classifier that improves its performance significantly.

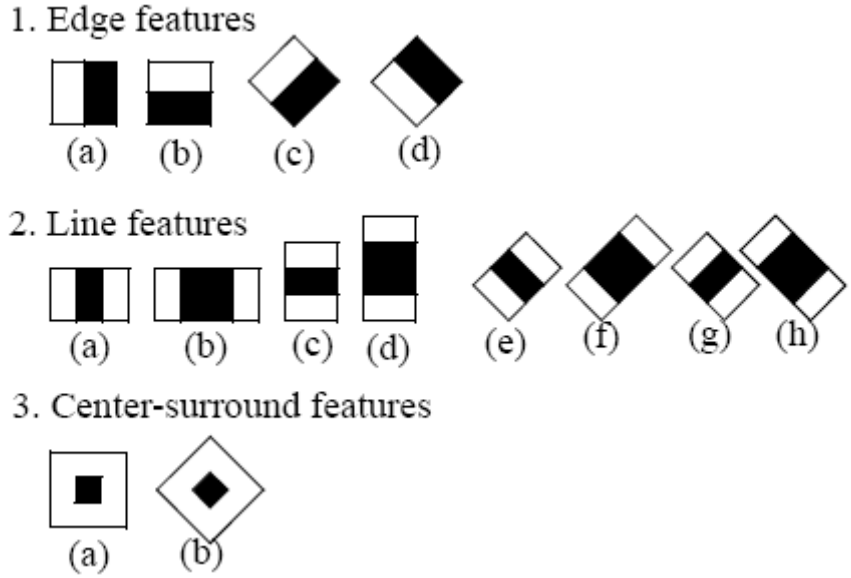


Figure 3.3.3. Set of features modified by Rainer Lienhart and Jochen Mayd. . Image taken from [Lienhart02].

This technique works well on objects that have distinguishing views, that is, front views of faces, backs, sides or fronts of cars. However it does not work well on side views of faces or corner views of cars, this is due to the sub-window which must be learnt, includes part of the changing background.

OpenCV implement all the necessary functions to train and run the Haar-classifier. In addition, there are three trained detectors, available in the files of the OpenCV, which are very interesting for our purpose:

- Upper body detector.
- Lower body detector.
- Full body detector.

These detectors were trained by Hannes Kruppa and Bernt Schiele [Kruppa03]. The upper-body detector is the most useful in our scenario because legs are often hidden and even when they are visible, they are more difficult to recognise.

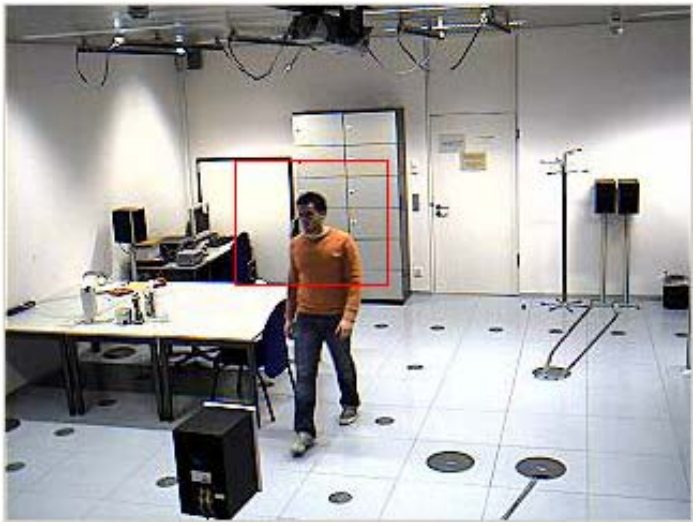


Figure 3.3.4. Detection of upper-body of a person using haar-classifier.

It was not possible to use this algorithm for the three-dimensional estimation of the position of people, because the detection rate was very low even to try to combine it with tracking algorithms. This is because the detector must rely on silhouette information and they include information about the changing background. Nevertheless, it is possible to improve its performance, by training the detector not only with back and front views of the body but also with side views.

Comparing this method with the previous methods based on motion detection, the following conclusions were achieved,

- Because this method is not based on motion, it can detect stationary people.

- Using motion based algorithms makes it usual, that two people walking together are interpreted like a single object; this should not happen when the Haar-classifier is used.

- The Haar-classifier algorithm is an extremely rapid object detector but it is much slower than every motion detection algorithm.

- The detection rate of the upper-body detector is not high enough for our application, it should be taken into account that to estimate the three-dimensional position of each person, we need at least a pair of two-dimensional coordinates in different cameras of the system.

3.4. Background Subtraction.

The background subtraction algorithms are used to isolate parts of objects (foreground) from the rest of the image (background). In this application the background corresponds to the stationary part of the image and the foreground corresponds to the part of the image with motion.

OpenCV does not implement any specific algorithm of background subtraction but it provides a lot of tools to implement them.

In order to define the background, several situations must be taken into account. In our case, we are watching a room and our desired foreground objects are people in the room. Our assumption is that everything large enough with movement in a scene is a person, but it could also be a chair being moved by a person. Somehow the foreground must be updated to include the chair which has been moved and the hole left by the old position of the chair. Other situations, which could produce false detections, are the illumination changes which can be considered as objects of the foreground.

The simplest background subtraction method is to subtract one frame from another and then label any difference bigger than a threshold as foreground. Then we obtain a mask of the same size of the image with non-zero values in pixels belonging to the foreground. Applying a dilate operation and finding the components of the mask which are connected, we get the different objects (people) of the foreground. This simple background subtraction method, combined with tracking techniques based in the Kalman filter, works well but it is very frame rate dependent.

Figure 3.4.1. Person tracked with method based in differencing frames background subtraction combined with a Kalman Filter. The blue box represents the region belonging to the foreground.

The main problem with this method is that the foreground obtained is up to the frame rate, so with a low frame rate the box obtained to represent the objects of the foreground will be larger than the object, and with very low frame rates (one frame per second) it is possible to find two objects belonging to the foreground

which represent the old and the new position of a single object (person) in movement (see Figure 3.4.2).

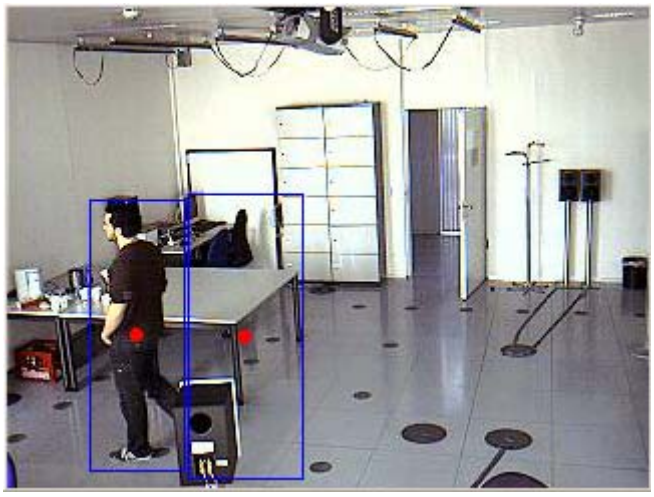


Figure 3.4.2.

Other problem is derived from the fact of the dilate operation must use a kernel big enough to join the pixels of a single object. Sometimes it also joins pixels of different objects, resulting as a single object of the foreground (see Figure 3.4.3). In addition, the point selected to represent each object has been the top-centered of each region with movement. This is influenced by the dilate operation so it will affect the accuracy of the three-dimensional reconstruction.

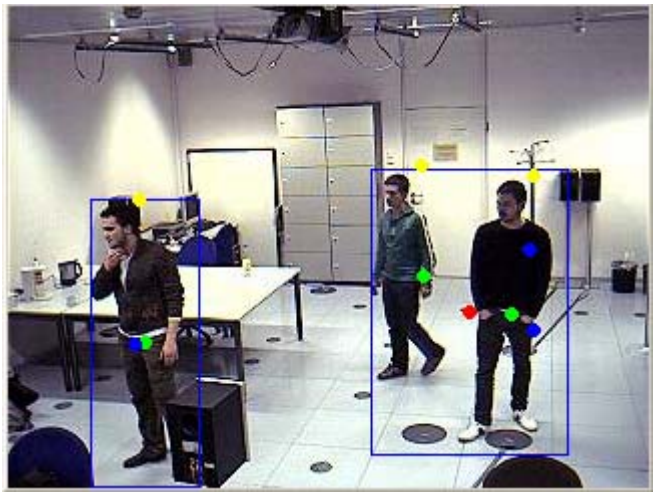


Figure 3.4.3. Two people are considered as an only foreground object.

In order to avoid the problems explained above, a more sophisticated background subtraction method was implemented. It takes the first image of the video stream as background. Then, a mask is obtained by a differencing operation between the background and the current image, followed by a threshold and dilates operations. This mask is used to find the connected components and in this way get the segmented objects of the foreground. The current frame is also subtracted from the previous frame, and a threshold operation is accomplished, giving as result a mask with pixels which change

from one frame to the previous frame. This mask is called “silhouette”, and is used to update a matrix which contains the timestamp of the last movement for each pixel; as it was done in the Motion History Image,

$$MHI(x,y) = \begin{cases} timestamp & if \quad silhouette(x,y) != 0 \\ MHI(x,y) & otherwise \end{cases}$$

(3.4.1)

This matrix is used to search for recent movements in the regions of the foreground. If there is no recent movement in this region, its content is used to update the background. To determine if there was or not recent movement in a region, a threshold is applied to our Motion History Image and the pixels within this region are counted. If there are not enough pixels with recent movement, relative to the surface of the region, then this region updates the background to the correspondent region.

This method has a better behaviour than the previous, because it does not depend on the frame rate, due to the time information that it handles. In addition, the kernel required for the dilate operation is smaller than the previous one. This allows us to distinguish between objects of the foreground even when they are very near (see Figure 3.4.4), and does not distort the pixel coordinates of the point.

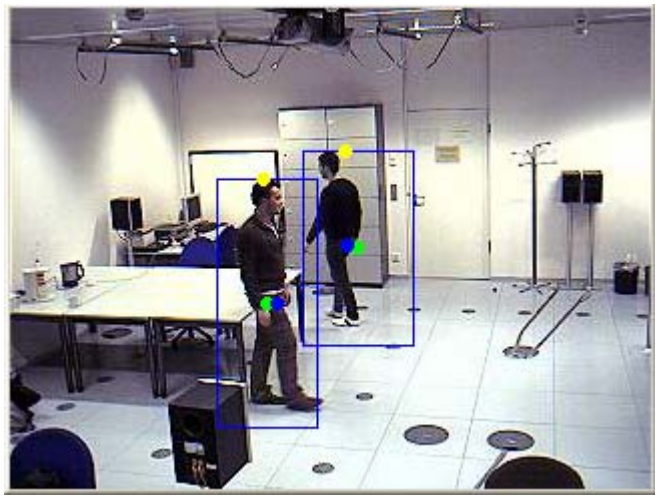


Figure 3.4.4. Two people which appear very near in the image are interpreted as two different objects of the foreground.

In order to make a faster update of the background when a big illumination change happens, if the algorithm detects a big change between the background and the current frame, it updates the whole background until this situation disappears.

The more sophisticated background subtraction technique has been used to measure the position of objects (people) in the current scenes, and these measurements have been used by Kalman filters (see Appendix B. Kalman

Filter) to estimate the position of the objects in the next frame. In this way we are able to track the objects in a period of time.

The point selected to represent each region of the foreground is the top-center point. The aim is to track the movement of this point in a plane, so the state vector will be composed by the position and velocity of the object in the two dimensional system.

$$x_k = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}_k^T \quad \textbf{(3.4.2)}$$

Although the state vector has four components, only the position components can be measured, not the velocity components, so the measurement vector is,

$$z_k = \begin{bmatrix} z_x & z_y \end{bmatrix}_k^T \quad \textbf{(3.5.11)}$$

Assuming constant velocity of the object between frames, and supposing a time between frames equal to *dt*, the transition matrix for our system can be written,

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.5.12)}$$

There is no control inputs in the system considered here, so the matrix *B* does not exist. The matrix *H* which relates to the current state with the measurement is,

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \textbf{(3.5.13)}$$

4. Analysis.

So far, different methods have been described in order to get information in two dimensions from each image, which is taken from a digital camera. It has also been described how to obtain the parameters of the pin-hole model of a camera, and how to use the information from pixel coordinates of the previous methods and the epipolar constraints obtained from the model of the cameras. Through this method, it is possible to obtain groups of points (one point per image) which are supposed to be the same as in the three-dimensional world. It is then possible via triangulation reconstruction to obtain the three-dimensional position of this point. In this chapter I analyze the behavior of the different methods used to get the pixel locations of the objects, the sources of error which have an influence in the system output, and finally I analyze the accuracy of the three-dimensional measurement.

4.1. Analysis of the behavior of the motion detection and tracking algorithms.

In chapter 3, the method, which was used to obtain the two-dimensional information in pixel units of the location of a person (object in movement) in an image, was described. Analyzing the behavior of these methods, I reached the following conclusions:

- **Optical Flow Algorithm.**

At the end of the experiment, the Pyramid Lucas-Kanade algorithm was not successfully implemented to track people. Only a group of points was being successfully tracked by the Pyramid Lucas-Kanade algorithm. The problem arises from the fact that it has not been possible to develop any matching algorithm between points of different cameras. This way it is impossible to accomplish the three-dimensional reconstruction of the points being tracked with this method.

Figure 4.1.1 illustrates the behavior of the optical flow algorithm.

- **Body detection with Haar Classifier.**

This method was studied because it could present some advantages:

- It can detect stationary people.

- Using motion based algorithms, it is usual that two people walking together are interpreted like a single object; this should not happen when the Haar Classifier is used.

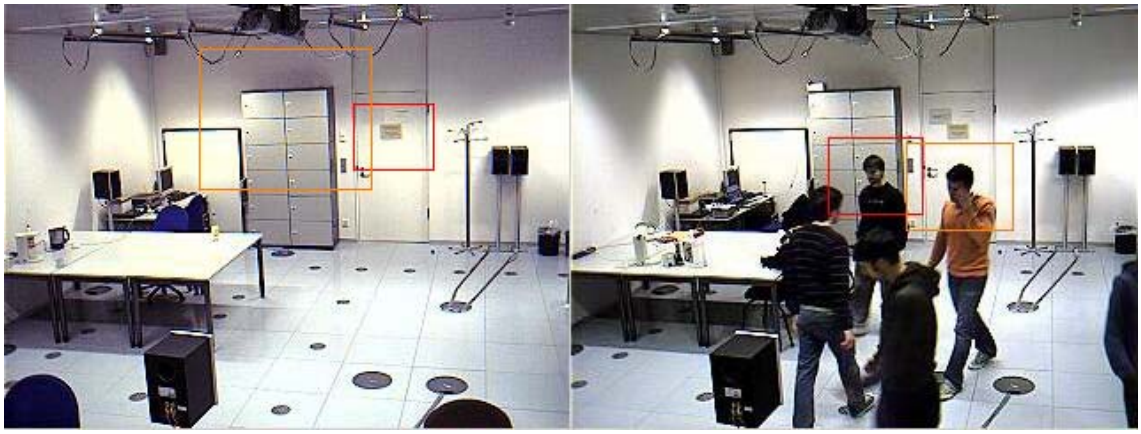


Figure 4.1.2. illustrates the behaviour of the body detector.

Despite these advantages, the Haar Clasifier was not used to estimate the three-dimensional position, because the detection rate of the upper-body detector is very low for this purpose. In order to be able to compute the three-dimensional location, the same body should be detected in at least two images.

- **Motion Templates.**

The two-dimensional information provided by the motion templates algorithm was useful to compute the three-dimensional position of objects in movement. After analyzing the behavior of this method,

- Motion templates algorithm provides the position of an object in movement, not only people.

- When several objects are too close together, they can be interpreted as an only object.

- The body of a person is sometimes interpreted by the detector as different objects.

- With low frame rates (2 fps) the hole that is left from the object in movement can be detected as other object.

- The points obtained with this method are not exactly at the top of the object in movement.

Figure 4.1.3. illustrates the behavioral features of the Motion Templates method.

- **Background Subtraction and Tracking based on Kalman filter.**

The information of the location of the foreground object in the image, which had been obtained through the background subtraction method, was successfully used to compute the three-dimensional position of these objects. The behavior of this method is very similar to the behavior of the Motion Templates but with some differences:

- The objects of the foreground are objects in movement, not only people.
- It distinguishes different objects when they are very near.
- Its frame rate is totally independent.
- The points provided by this method are in the top-center of the object.
- It is the fastest method.

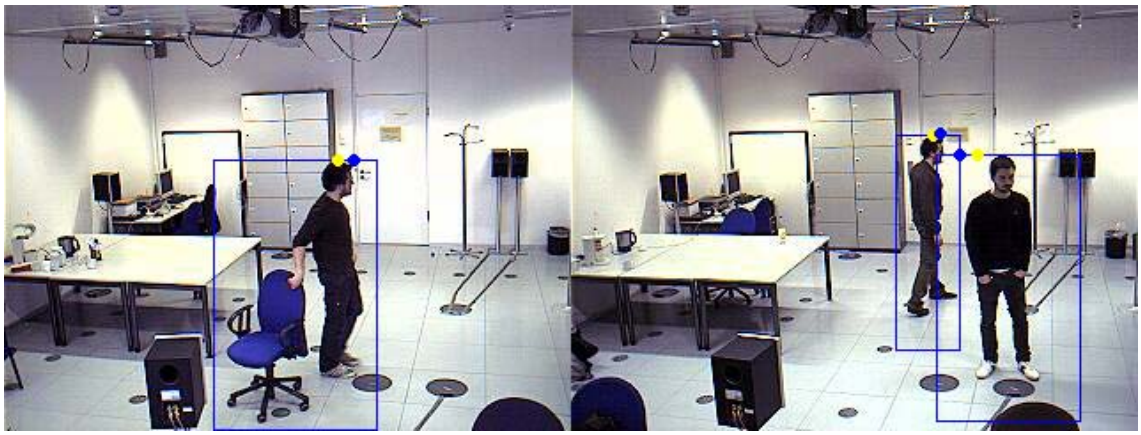


Figure 4.1.4. illustrates the behavioral features of the Background Subtraction method.

- **Runtime analysis.**

Although the aim of this application is not to work in real time, a part of it may be used in that context. The runtimes of the different algorithms in a single core

processor 1.5 GHz with an image size of 320x240 pixels have been calculated and the results are shown in the Table 4.1.

Method	Runtime (ms.)
Optical flow	55
Body detection	500
Motion templates	57
Background subtraction	3

Table 4.1 Comparison between runtimes.

4.2. Sources of error

In this the section different sources of error will be analyzed, but befote that it should be kept in mind that in this system is very difficult to compute a value for an error because the detection algorithms used to compute the objects location in the images are based in motion detection and the measurement process of the instant position of one person in movement in a room could be complicated. In addition, the objects are represented by a single point located in its top and this is not a fix position in the three-dimensional world but it depends on the relative position of the object and the cameras.

It is possible to distinguish three sources of error those which are relative to the calibration process, those which are relative to the location provided by the detection algorithms, and those which are relative to matching process among points of images from different cameras.

- **Errors related to the calibration process.**

The calibration process let us compute the camera extrinsic and intrinsic parameters, according to a model of the camera, as it was explained in Chapter 1, it was used the pin-hole camera model without taking into account the distortion lens parameters because the error introduced by this simplification of the model was considered negligible. Although the model of the camera match perfect with its behaviour, it provide us its position relative to a chess board frame (see Chapter 1) and this position must be related to the world-reference-frame, and the measurement of this relation will also have an influence in the global error.

Summarizing the errors derived of the calibration process:

- The goodness of fit of the model.
- The accuracy of the algorithm to compute the model parameters.
- The accuracy of the measurements of the relative position between the board-reference-frame and the world-reference-frame.

- **Errors related to the detection algorithms.**

The algorithms which compute the pixel coordinates of the people (objects in movement) in each image from the different cameras produce two kinds of error:

- Errors related with the behaviour of the algorithm, that is, when the algorithm interprets one object as if there were two different objects, or vice versa, when the algorithm interprets two objects as an only object.
- Errors related with the accuracy of the location, the ideal case is that the points computed for the detection algorithms in each image would be the projection of the same three-dimensional point, for this reason it was

chosen the top-center point to represent each object and instead of its center of mass because if this point would have been chosen it would have introduced large matching error when some parts of the people are hidden by stationary objects (chairs, tables,…). Even so, the points computed by the detection algorithms depend on the projection of the objects in each camera and they do not match exactly with the same point in the real world and due to the fact of taking the top points they are usually over the top of the object and the influence of this error increase with the distance of the object to the camera.

- **Errors related to the matching algorithm.**

The matching algorithm implemented for this application it is based only in the epipolar constraint among the points computed by the detection methods in the images of each camera, but there is no matching algorithm implemented in the way of features matching or correlation matching. The reason for not implementing these kinds of matching algorithms is that they assume that most scene points are visible from each viewpoints and the relative position of the cameras make this assumption false as it is shown in the Figure 4.2.1.

Figure 4.2.1. shows two frames taken simultaneously from two cameras.

The matching algorithm consists of linking those points visible from each camera and which minimize the distance to the epipolar lines of the other cameras. So the matching algorithm fails when links points of images which are not correspondent and then a non desired three-dimensional point is computed.

4.3. Analysis of the accuracy of the three-dimensional reconstruction.

In this subsection it is analyzed the accurate of the three-dimensional measurement, in order to accomplish this analysis videos with a person doing small movements around nine known positions were recorded, although the exact location of the people it is unknown, it must be within a small known region (60x60 cm) of the room. Two videos were recorded with different frame rates (2 and 5 fps), and the data were analyzed taking into account two and three views. Figure 4.3.1. shows the nexus lab, the location of the world-reference-frame, and the location of the cameras used to record the videos to be analyzed. Figure 4.3.2 represents the isometric projection of the nexus lab, the location of the world-reference frame and the three camera-reference-frames. Notice that the grid of the ground of the isometric projection has been drawn to represent the stabs of the floor of the nexus lab, and the nine yellow ones are the regions were the person was recorded.

Figure 4.3.1. shows the location of the word-reference-frame and the cameras used in the accurate analysis process in the nexus lab.

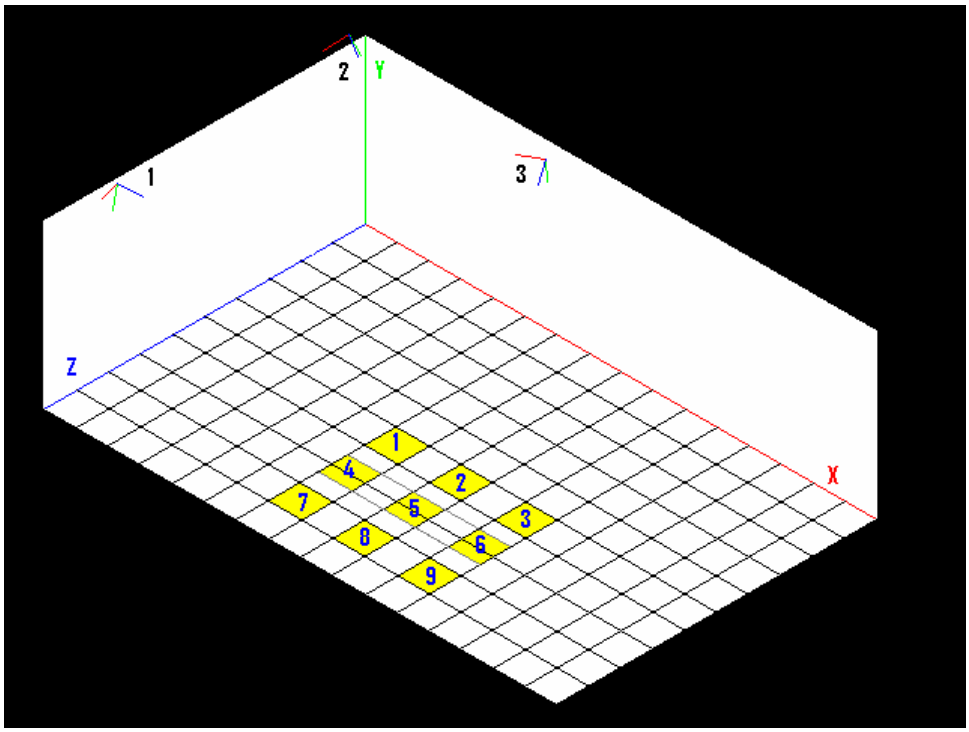


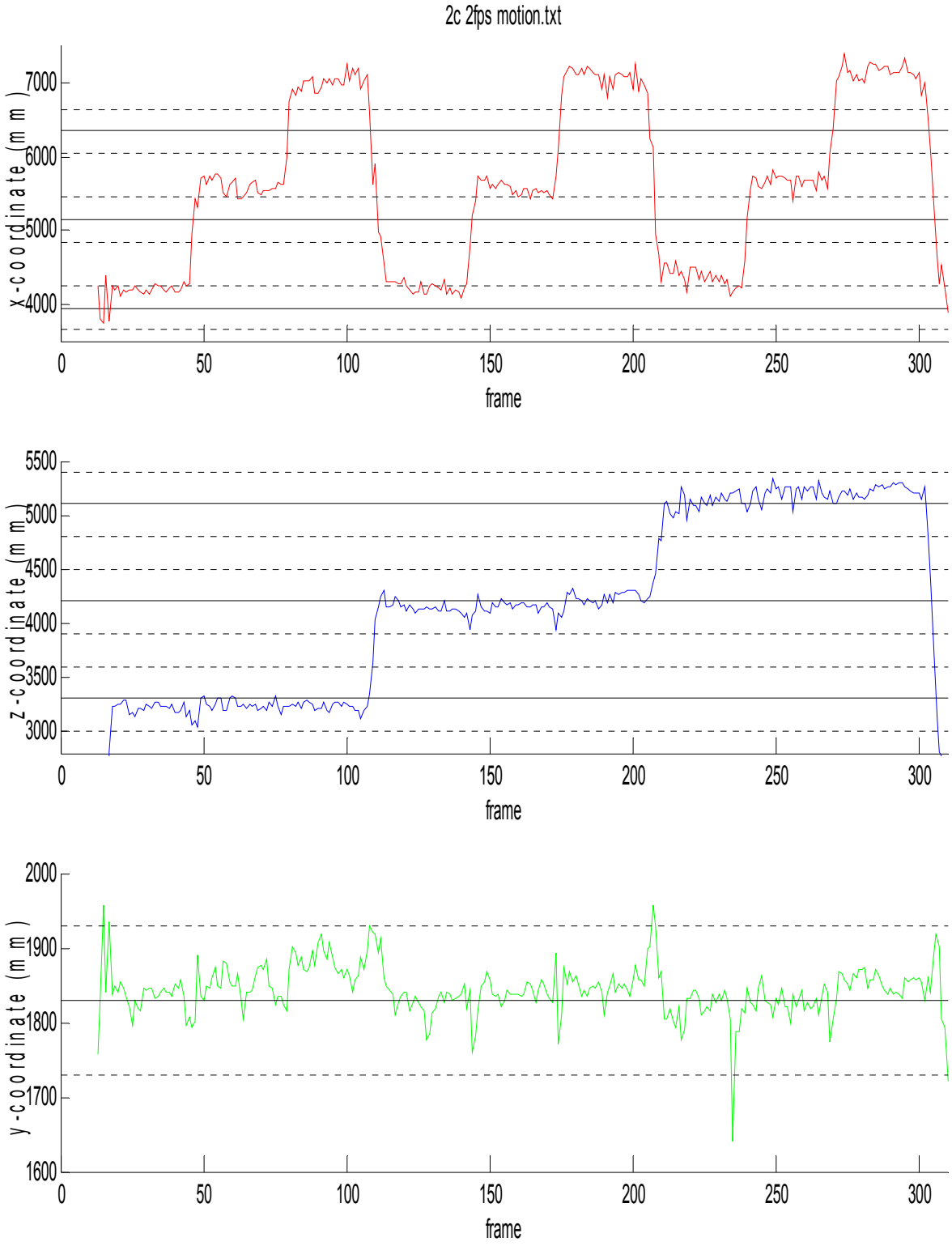
Figure 4.3.2 shows the isometric representation of the nexus lab, the world-reference-frame, the three cameras-reference-frames, and the location of the regions used in the accurate analysis process. (The grid in the plane XZ represents the slabs of the nexus lab)

In the Table 4.3.1.are the world-reference-coordinates of the boundaries and the midpoints of the nine regions in yellow shown in the Figure 4.3.2. The y-coordinate correspond to the height of the person which is 1830 mm.

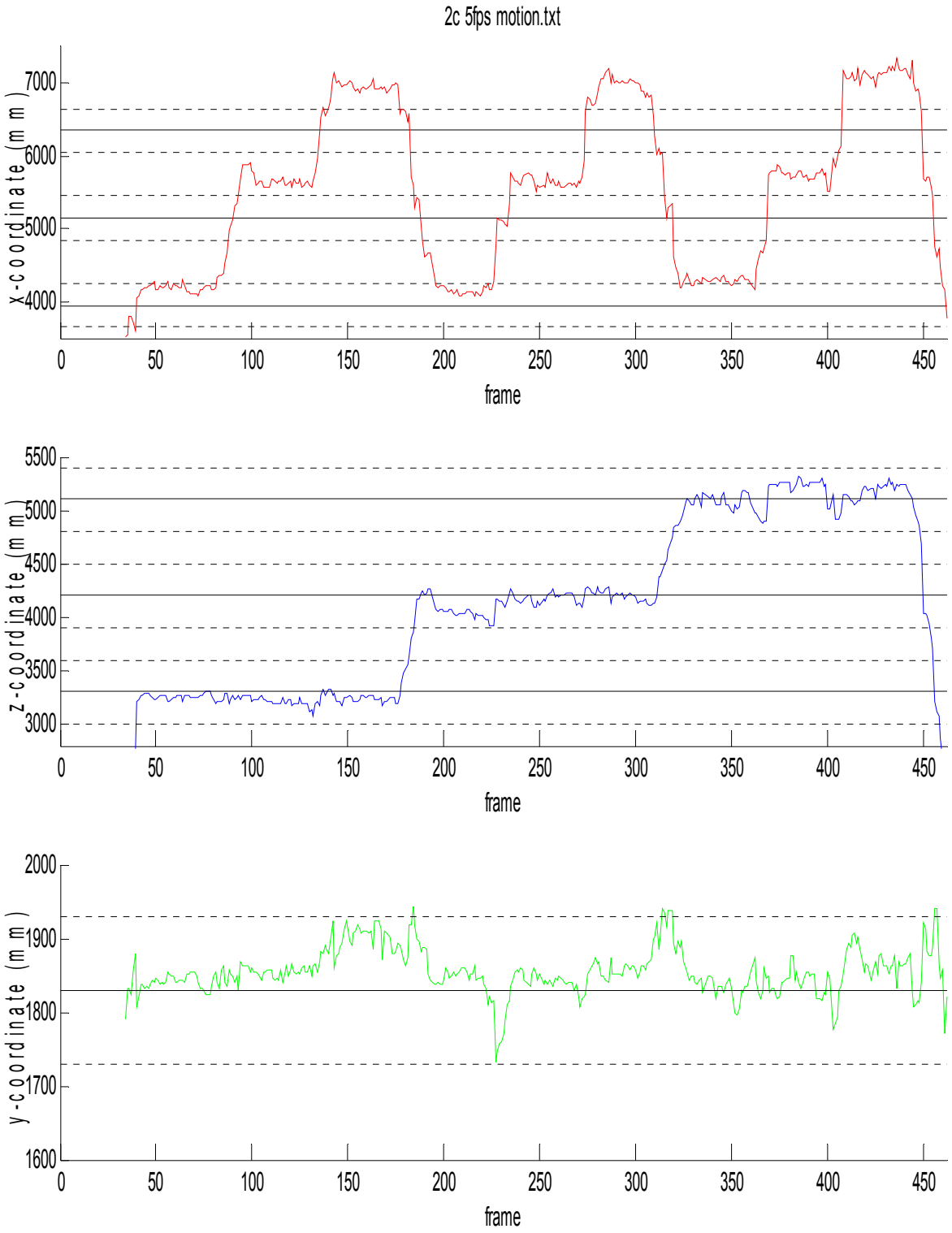
	X (mm)			Z (mm)		
	min	mean	max	min	mean	max
1	3650	3950	4250	3000	3300	3600
2	4850	5150	5450	3000	3300	3600
3	6050	6350	6750	3000	3300	3600
4	3650	3950	4250	3900	4200	4500
5	4850	5150	5450	3900	4200	4500
6	6050	6350	6750	3900	4200	4500
7	3650	3950	4250	4800	5100	5400
8	4850	5150	5450	4800	5100	5400
9	6050	6350	6750	4800	5100	5400

Table 4.3.1. Boundaries and midpoints of the test regions.

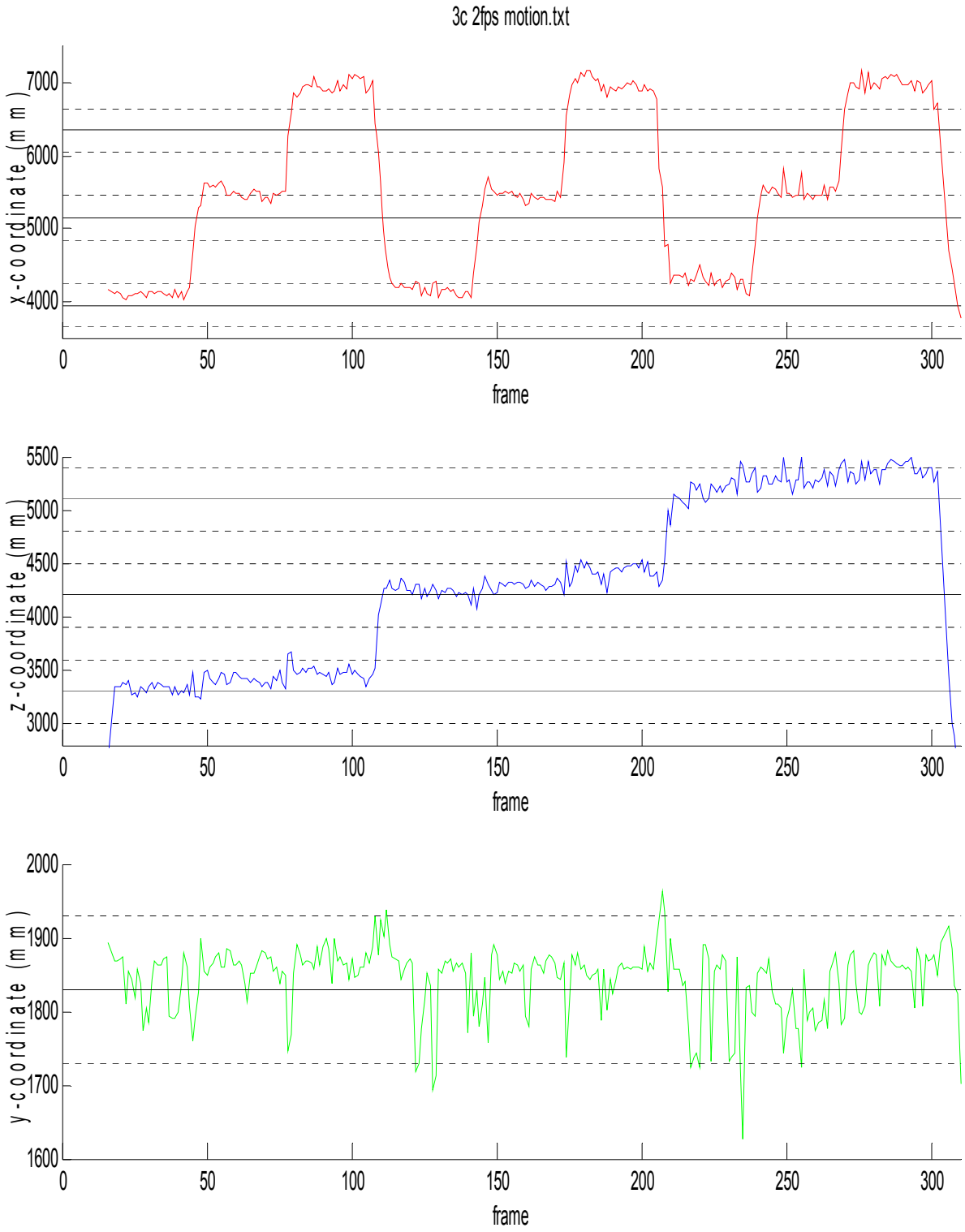
The Graphs 4.3.1 - 4.3.8 show the output of the system with different configurations when the person goes from the first to the last region staying within each region for a period of time. The black solid lines shows the mid point of the region and black dotted lines the boundaries of the regions in each coordinate.



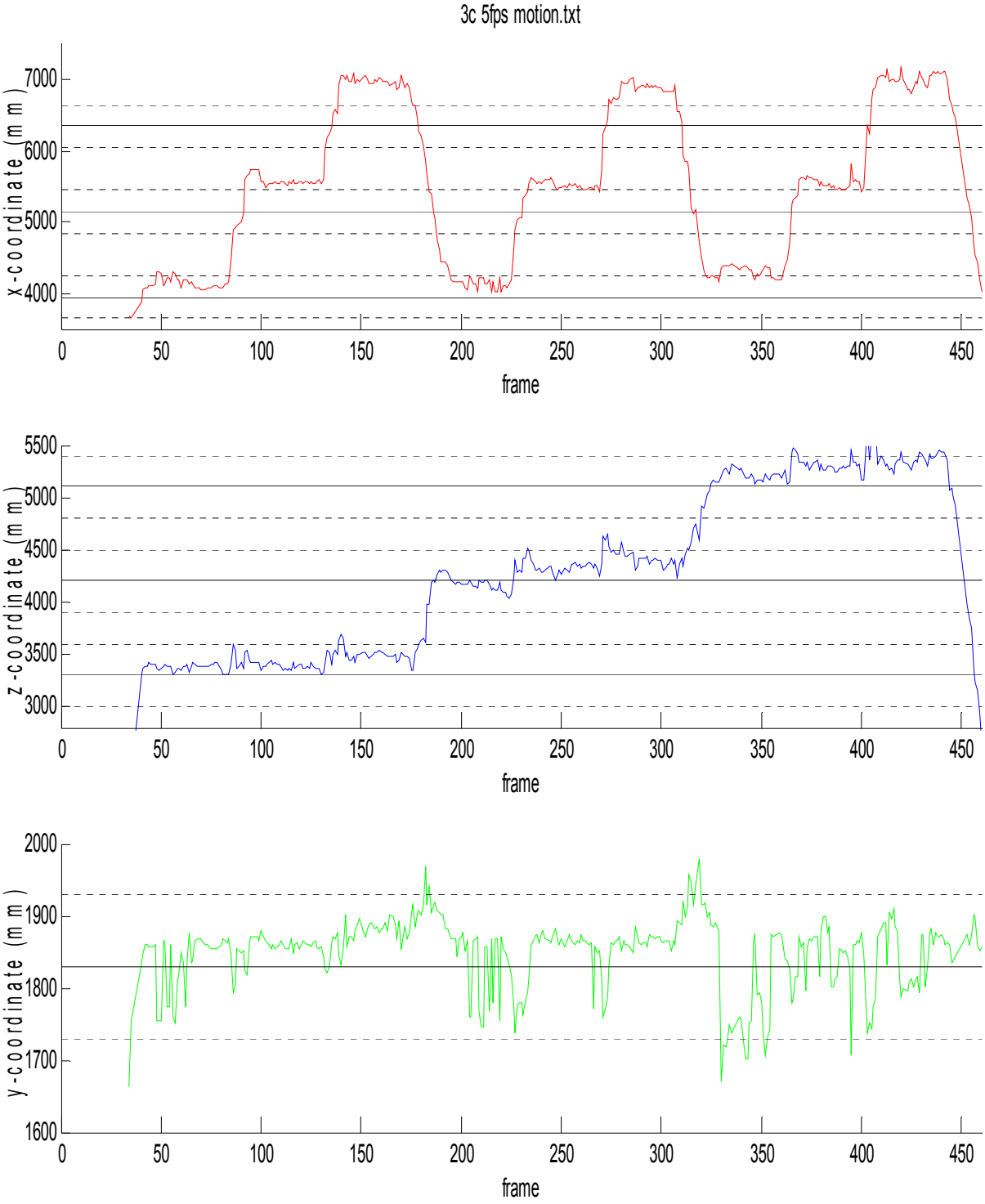
Graph 4.3.1. Output of the system using 2-cameras, 2 frames-per-second, and Motion-Template method.



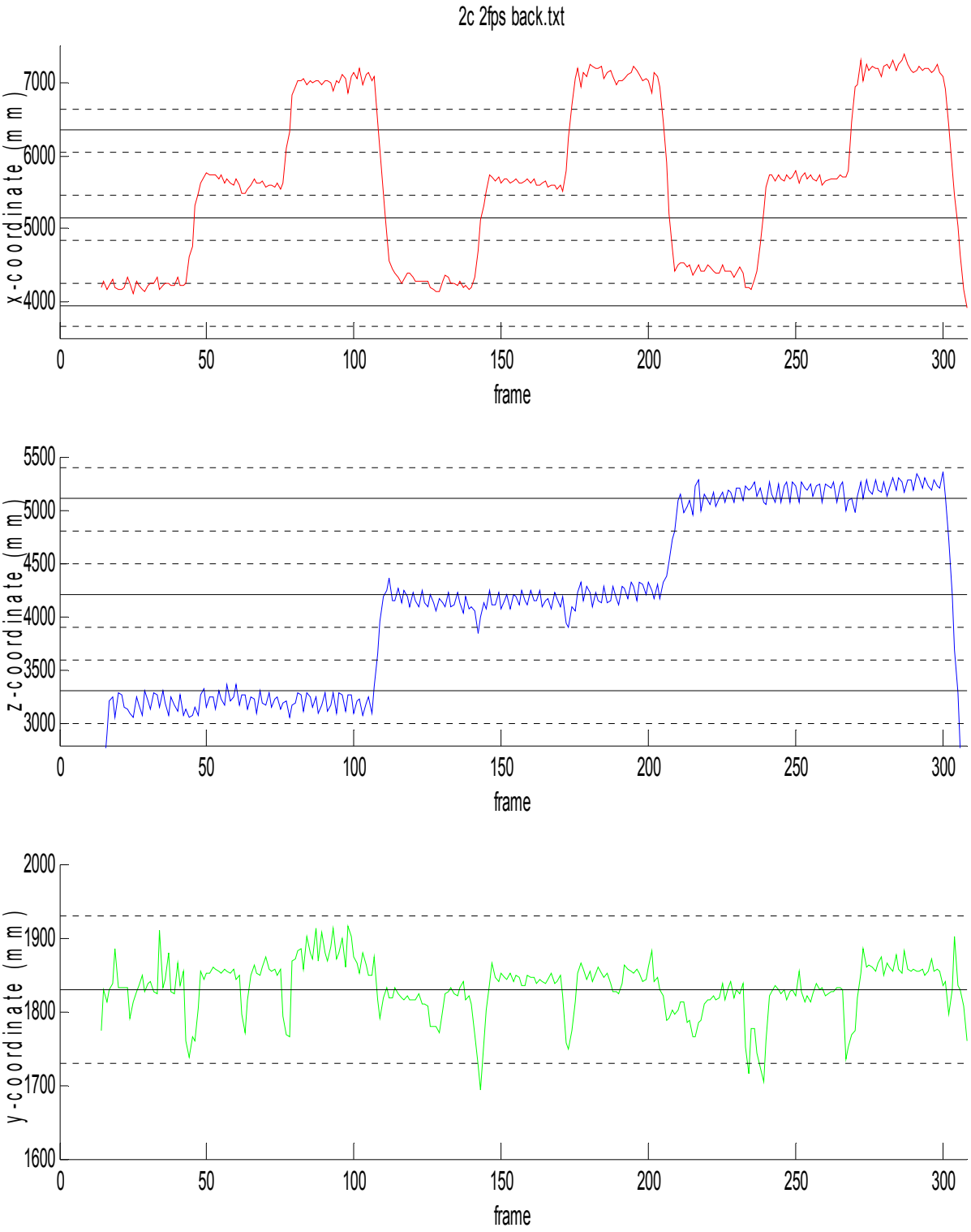
Graph 4.3.2. Output of the system using 2-cameras, 5 frames-per-second, and Motion-Template method.



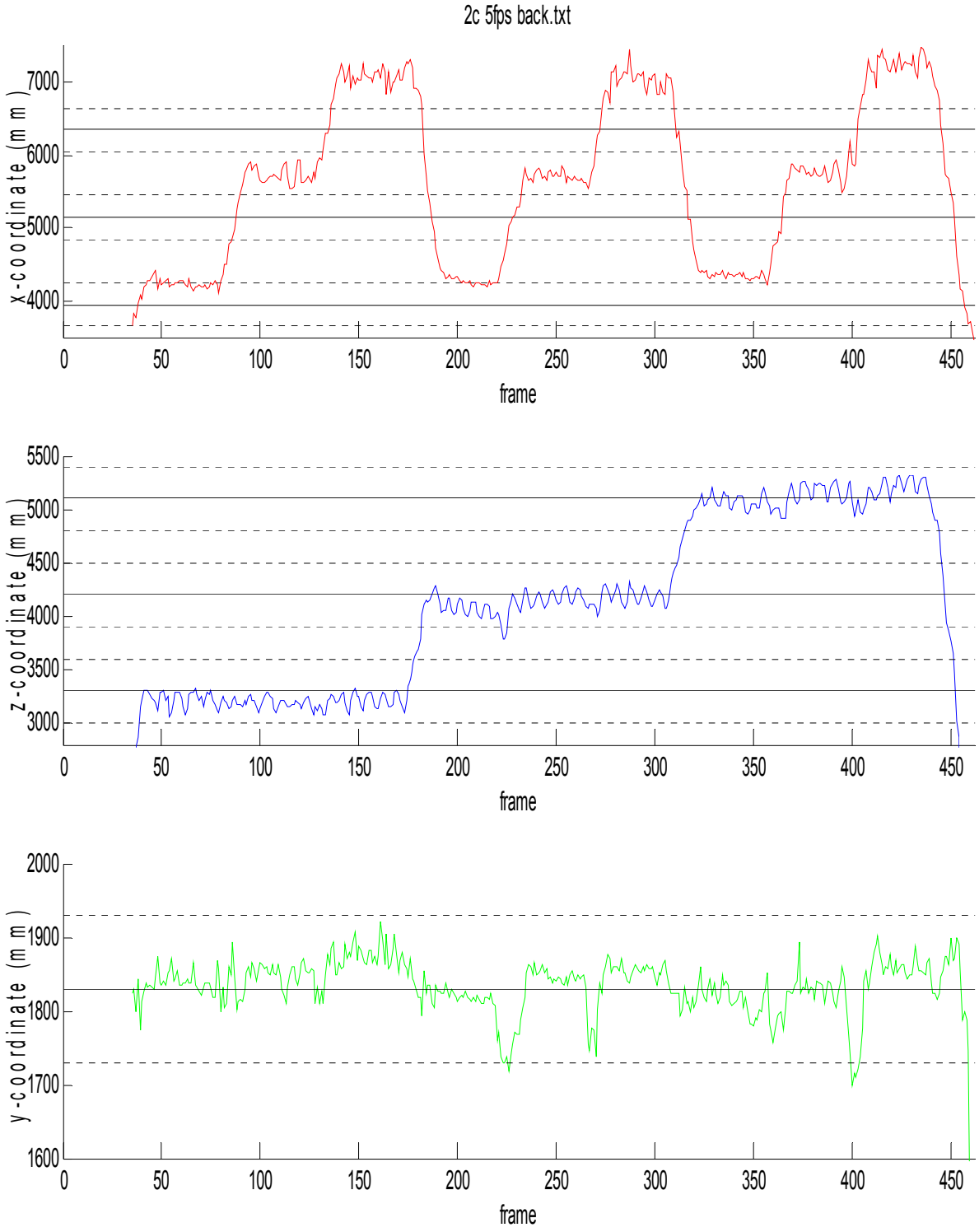
Graph 4.3.3. Output of the system using 3-cameras, 2 frames-per-second, and Motion-Template method.



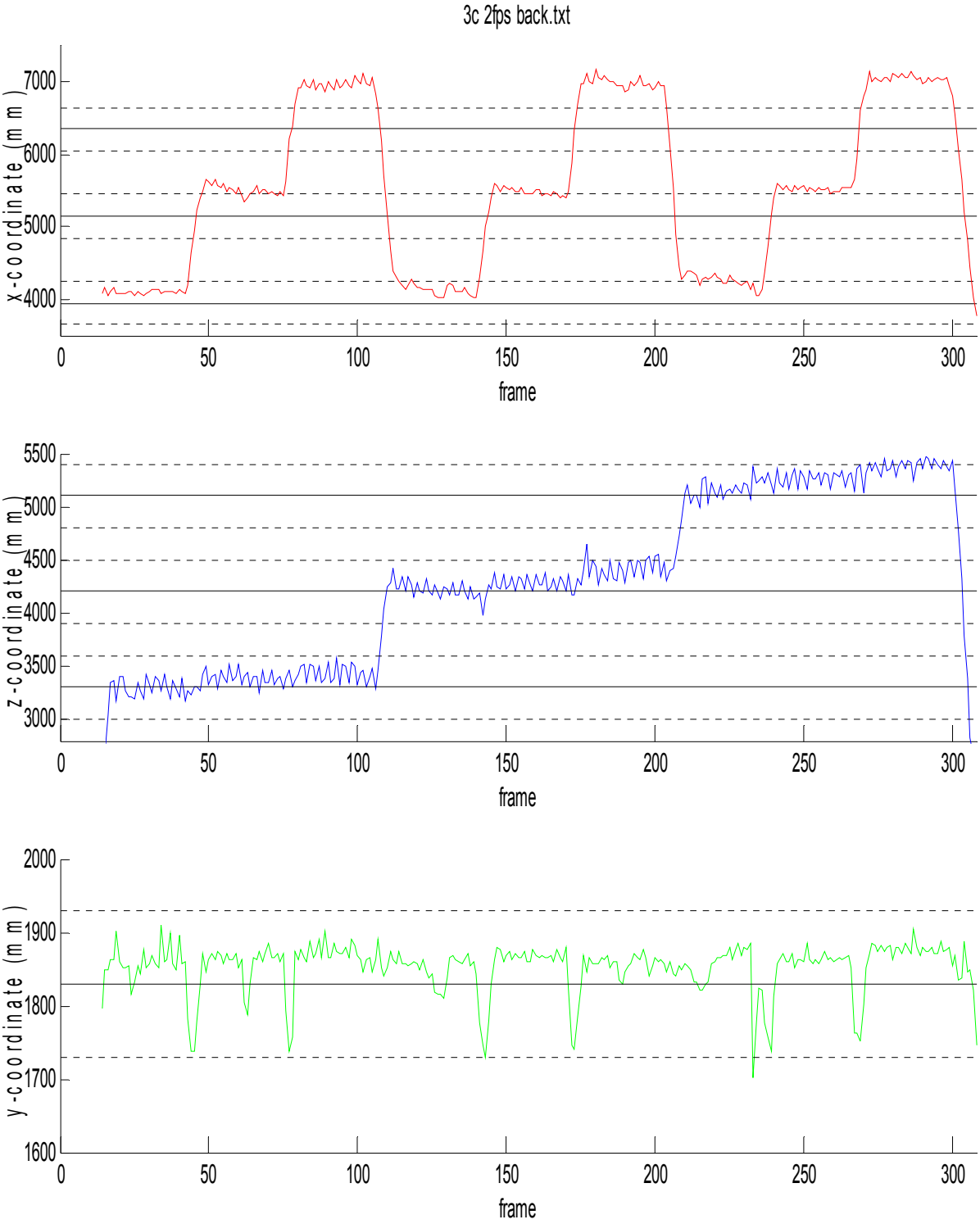
Graph 4.3.4. Output of the system using 3-cameras, 5 frames-per-second, and Motion-Template method.



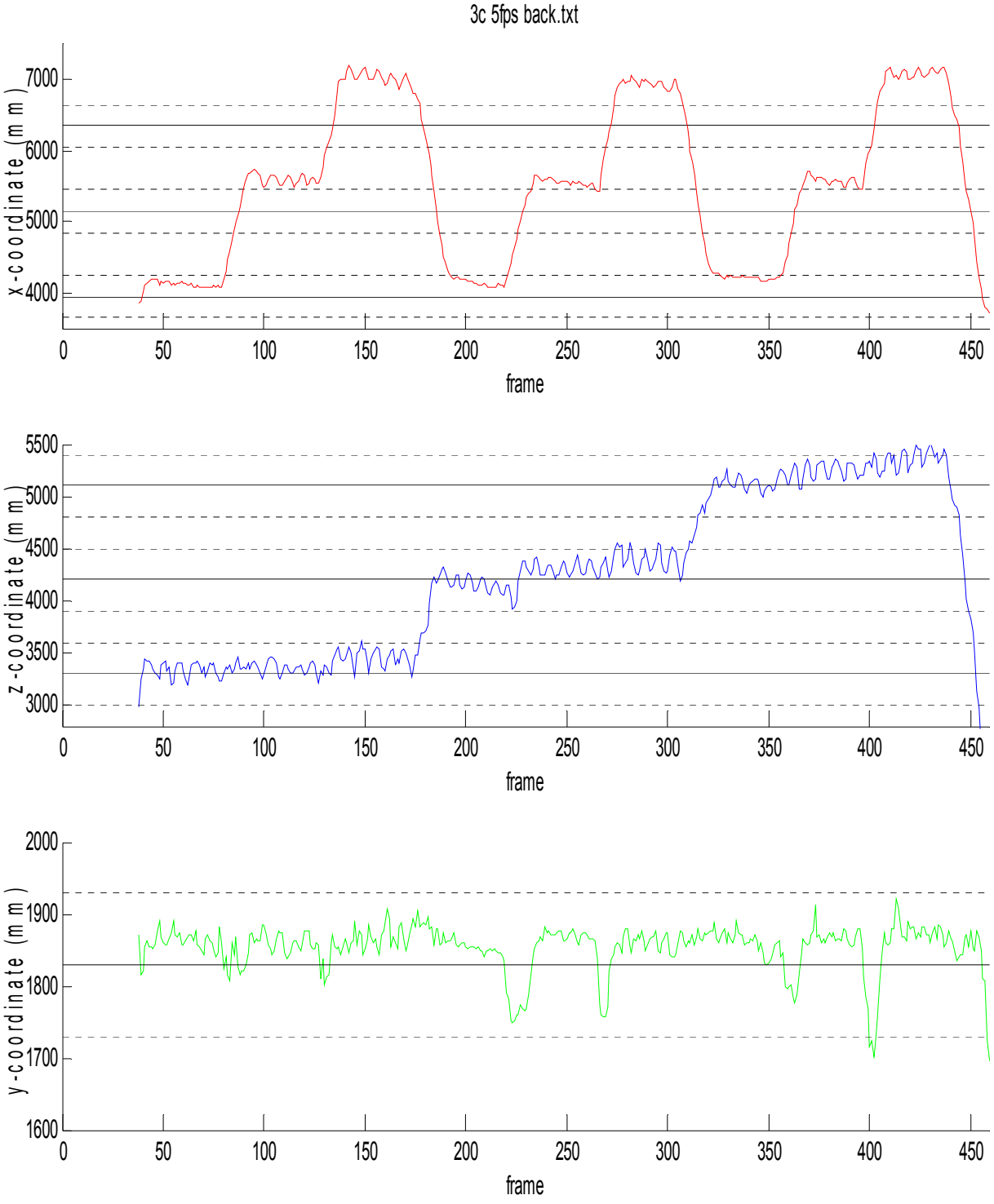
Graph 4.3.5. Output of the system using 2-cameras, 2 frames-per-second, and Background Subtraction method.



Graph 4.3.6. Output of the system using 2-cameras, 5 frames-per-second, and Background Subtraction method.



Graph 4.3.7. Output of the system using 3-cameras, 2 frames-per-second, and Background Subtraction method.



Graph 4.3.8. Output of the system using 3-cameras, 5 frames-per-second, and Background Subtraction method.

In the previous graphics the position measured by the system with the different configurations has been shown. It can be noticed that the error in the x-axis is the highest and it increases with the distance of the object to the cameras 1 and 2 and it decreases when the camera 3 is used to compute the three-dimensional coordinate, this is due to the rays from the cameras 1 and 2 becomes more and more parallel when the point they are pointing to is further, so small errors in the correspondence become big errors in the reconstruction this is illustrate in the Figure 4.3.3.

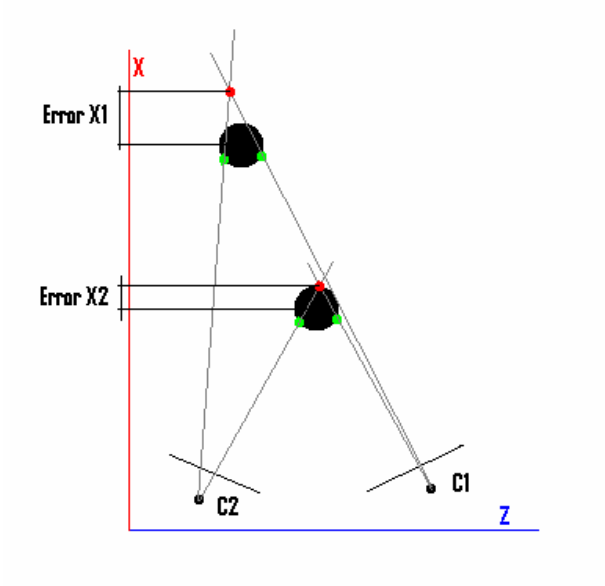
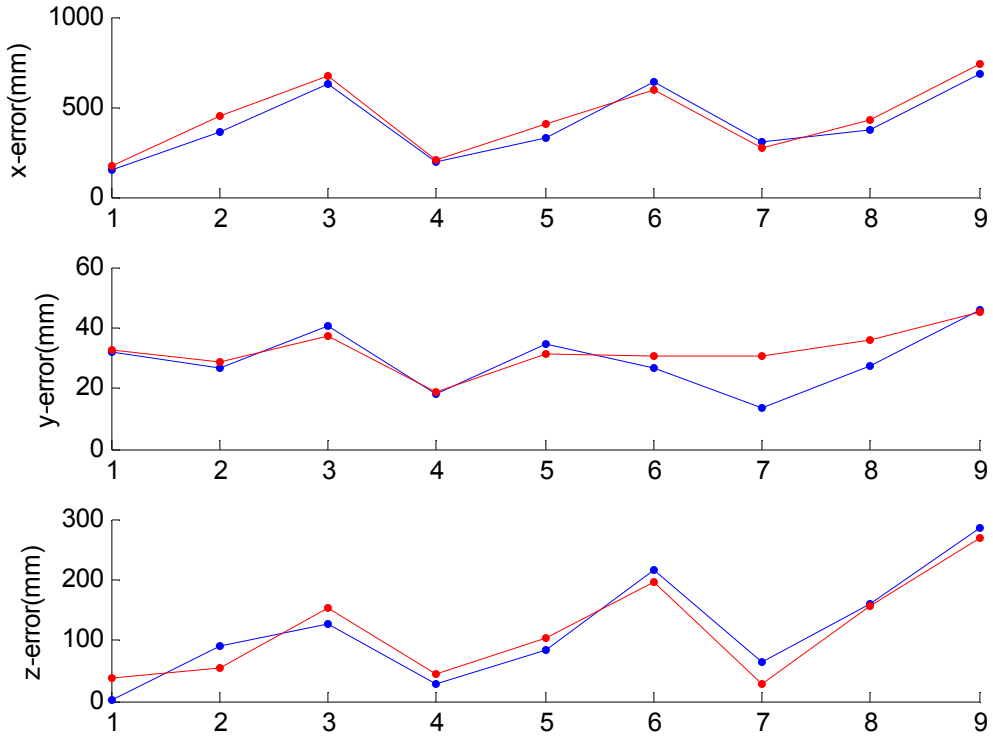


Figure 4.3.3. Shows the relation between the distance to the camera centers and the error

In order to compare the results with the different settings of the system the measurements have been separated according to the regions they belongs to, and the absolute mean errors have been computed taking the midpoint of each region as real value.

- **Error comparison between different frame rates.**

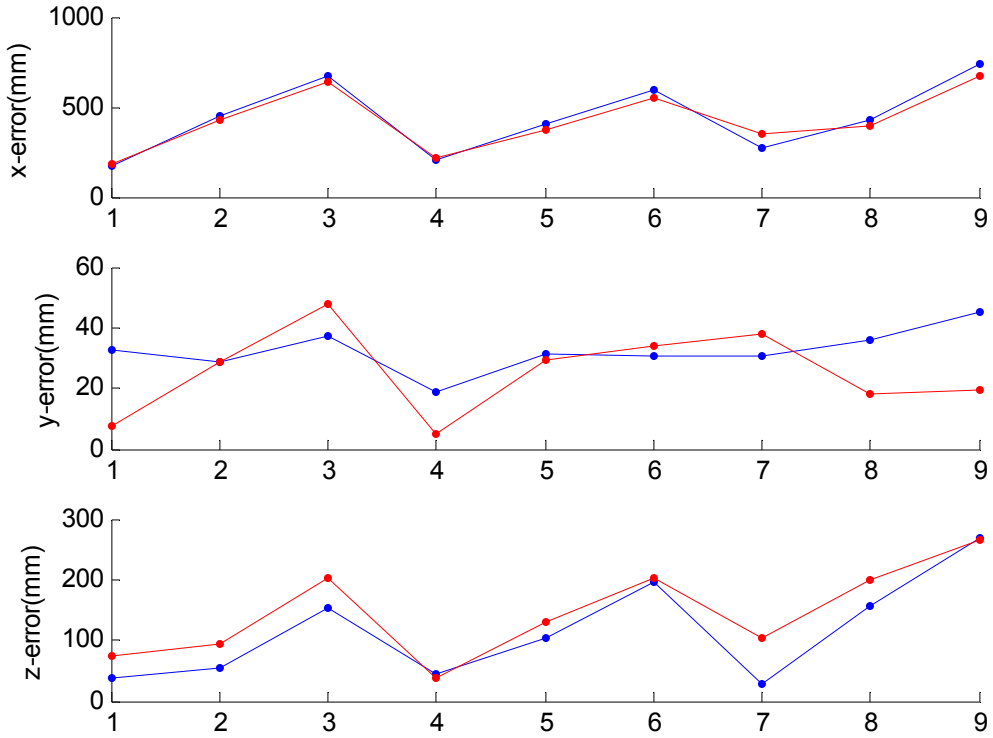
Analyzing the results with different frame-rates, it seems that the frame rate does not have an influence in the results of the program but they do, not in the error measured with this process, but in the behaviour of the algorithm based on frame differencing. The Figure shows the comparison between the errors of the system output using 2 and 5 fps and background subtraction method, and there is any obvious relationship between the error and the frame rate.



Graph 4.3.9. Shows the absolute mean error of the system output in each region using background subtraction method, 3 cameras, 2 fps (blue) and 5 fps (red).

- **Error comparison between different methods.**

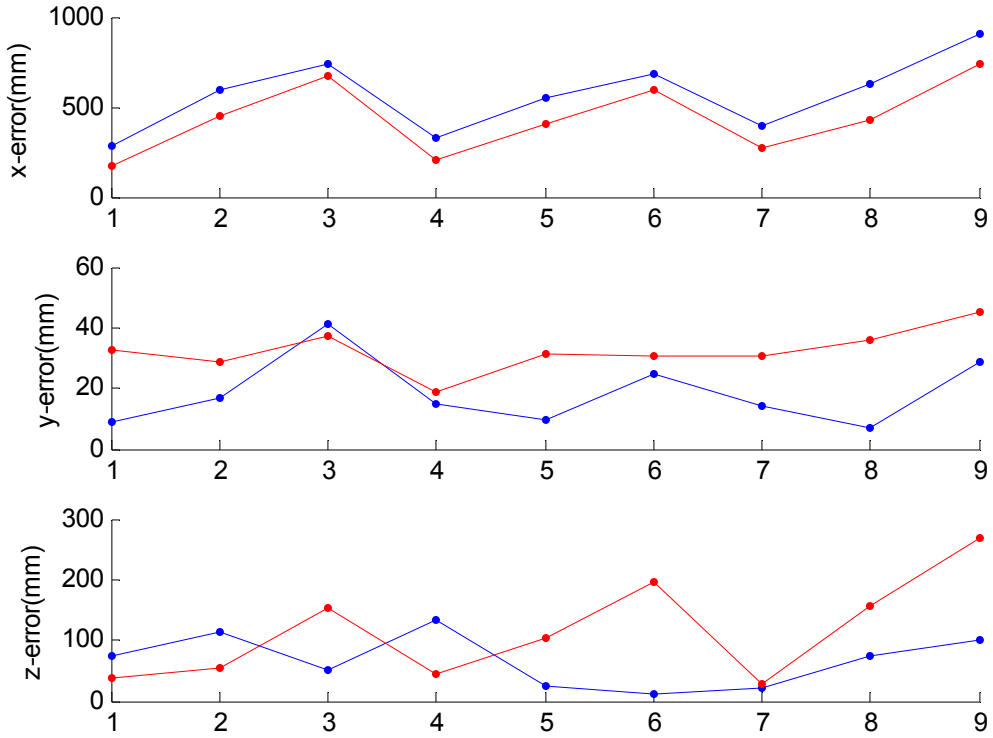
The error using Background Subtraction and Motion Templates methods it is also very similar. . Graph 4.3.10.shows the absolute mean errors of the system using the same frame rate and number of cameras but the different methods.



Graph 4.3.10. Shows the absolute mean error of the system output in each region using 3 cameras 5 fps and Background Subtraction method (blue) and Motion Template (red),.

- **Error comparison between two and three cameras.**

Graph 4.3.11. shows the absolute mean error of the system using the cameras 1 and 2 (blue) and the cameras 1, 2 and 3, it is noticed that when the third camera is used to compute the three-dimensional location of the person the error in the x-axis decreases while the others coordinates remain constant or increase, this is due to the relative position of the third camera, which focus is almost perpendicular to the focus of the other cameras (see Graph 4.3.11.).



Graph 4.3.11. Shows the absolute mean error of the system output in each region using Background Subtraction method 5 fps, 2 cameras (blue) and 3 cameras(red).

- **Distance between correspondent rays.**

The distance between correspondent rays can be understood as a measurement of the goodness of the matching between points of images of different cameras, and in the ideal case should be zero.

In order to analyze the goodness of the matching among points, the distance between each pair of rays for each measurement has been computed and stored. Graphs 4.3.12 and 4.3.13 shows the distance between each pair of rays using three cameras, it can be noticed that the distance between the rays from the cameras 1 - 2 is much shorter than the distance between the rays from the cameras 1 - 3, and 2 – 3, this is due to the projection of the three-dimensional world is more similar in cameras 1 and 2 than the cameras 1-2 and 3, and this has an influence in the matching goodness since we take the top-center point of the object in movement to represent the hole object, this situation can be seen in the *Figure 4.3.4.*..

Comparing the mean values of the distance between rays using Motion Templates and Background Subtraction methods can be noticed that the distance is longer when the first method is used. This is due to the Motion Templates used in our system it is based in frame differencing to get a silhouette this produces that the point chosen to accomplish the three-

dimensional reconstruction is influenced by the trace left of the tracked object along different frames.

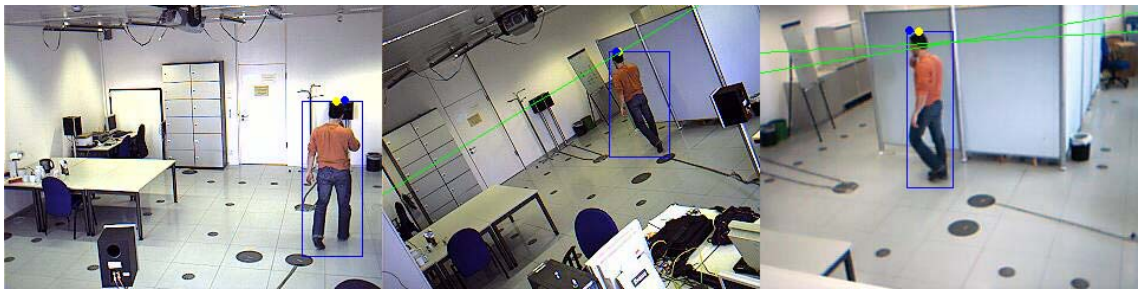
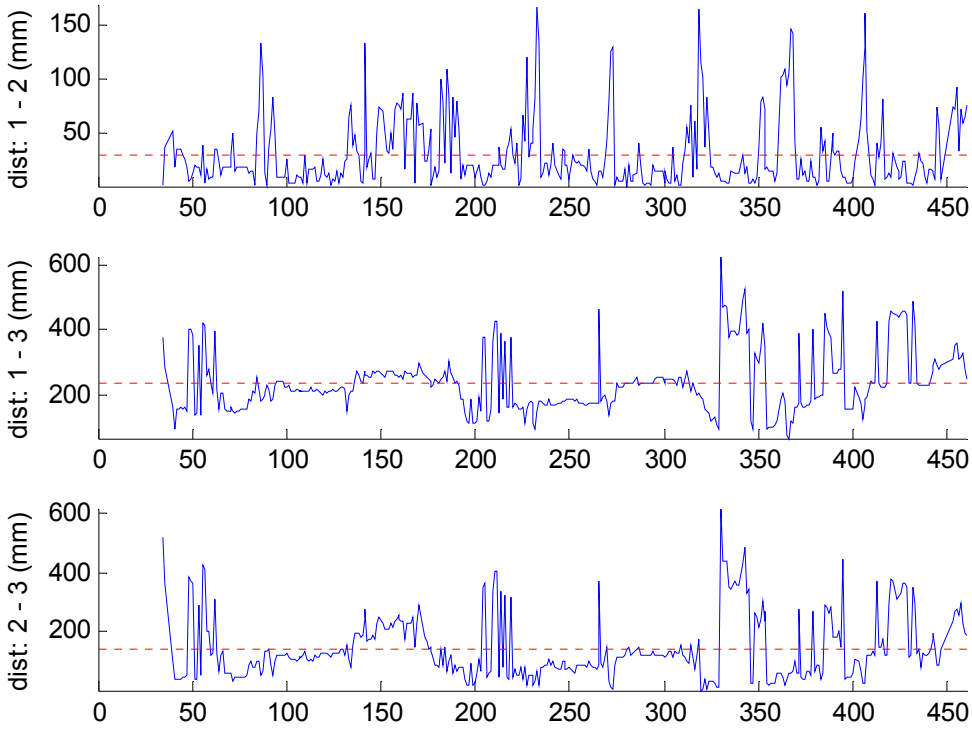
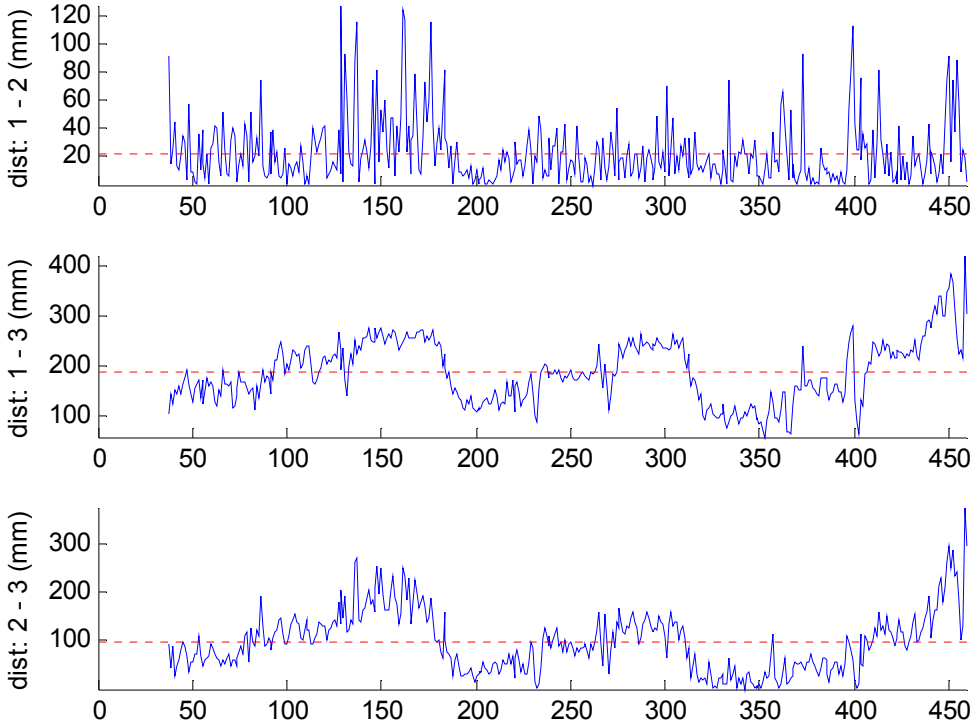


Figure 4.3.4. Shows the person being tracked, the yellow points in each image are the matched points and the green line the line projection of the previous to the current camera.



Graph 4.3.12. shows the absolute (blue) and mean (red) distance between each pair of rays, using the Motion Templates method, 3 cameras and 5 fps.



Graph 4.3.13. shows the absolute (blue) and mean (red) distance between each pair of rays, using the Background Subtraction method, 3 cameras and 5. fps.

➢ Summary.

- It has been tried to compute the three-dimensional location of people in a room using four different methods to get the two dimensional information from the image of each camera, and it has been accomplished only with two the Motion Templates and Background Subtraction methods.

- Both methods have reported similar errors, although the Background Subtraction method using a Kalman filter to track the foreground objects is faster, frame rate independent and behaves better.

- The measurement error in the x-axis of the world-reference-frame is the largest and increase with the distance due to the location and orientation of the cameras 1 and 2. This error decreases when the camera 3 is used to compute the three-dimensional location.

- The system works well when there is only a person in the room but it does not work when several people appear together or occluded in the video, due to there is no matching algorithm between points of the different images.

- According to the results obtained we can conclude that the method with the best behaviour and accuracy is the Background Subtraction Method.

5. Summary and outlook.

➢ Summary.

The aim of this thesis is the detection of people in images captured from multiple cameras, estimate their three-dimensional location, based on the information obtained from the cameras, and investigate the accuracy of the three-dimensional location.

In order to achieve this aim, it was necessary to obtain the relative location between the camera-reference-frames and a known reference-frame. This was accomplished according to what was explained in chapter 2 of this work.
Combining this information with the location of the people in each image captured from the different views, the three-dimensional reconstruction can be accomplished. The different methods to obtain the location of people in an image were studied in chapter 3. The last part of this work, chapter 4, is dedicated to investigate the accuracy of the systems designed to compute the three-dimensional location of people.

➢ Outlook.

There are several ways to improve and extend the results of this thesis. Some of them are,

- Using higher camera resolutions, this will increase the accuracy of the location, and at the same time the runtime of the algorithms.

- Adding more views of the scene, the three-dimensional point computed will be more accurate.

- Improving the tracking people algorithms to allow the system to compute the three-dimensional location of more than one person. To achieve this, the algorithm should be able to track the people when they are partly-hidden by other people.

- Methods based on pattern recognition are also very interesting, because they do not need people in movement, and in that way are totally frame rate independent.

Bibliography.

[Beauchemin95] S. S. Beauchemin and J. L. Barron, “The computation of optical flow”, 1995.

[Berthold81] Berthold K.P. Horn and Brian G. Rhunck, “Artificial Intelligence”. 1981.

[Beauchemin95] S.S. Beauchemin and J.L. Barron, “The Computation of Optical Flow”.

[Bobick96] A. Bobick and J. Davis, “Real-time recognition of activity using temporal templates,” 1996.

[Bradski98] G. R. Bradski, “Computer video face tracking for use in a perceptual userinterface,” 1998.

[Bradski00] G. Bradski and J. Davis, “Motion segmentation and pose recognition with motion history gradients”, 2000.

[Bradski08] Gary Bradski and Adrian Kaehler,”Learning OpenCV”.

[Brown66] D. C. Brown, “Decentring distortion of lenses”, 1966.

[Brown71] D. C. Brown, “Close-range camera calibration” , 1971.

[Davis97] J. Davis and A. Bobick, “The representation and recognition of action using temporal templates”,1997.

[Davis99] J. Davis and G. Bradski, “Real-time motion template gradients using IntelCVLib”, 1999.

[Horn81] B. K. P. Horn and B. G. Schunck, “Determining optical flow”, 1981.

[Harley00] Richard Hartley and Andrew Zisserman “Multiple View Geometry in Computer Vision”, 2000.

[Höferlin07] Markus Höferlin “Stereovision durch Kombination einer omnidirektionalen und einer perspektivischen Kamera”, 2007.

[Huang95] Y. Huang and X. H. Zhuang, “Motion-partitioned adaptive block matching for video compression”, 1995.

[Jean99] Jean-Yves Bouquet, “Pyramidal Implementation of the Lucas Kanade Feature Tracker”, 1999.

[Kinkeldei07] Patrick Kinkeldei “3D Objektlokalisierung anhand verteilter Messungen”, Mai 2007.

[Kruppa03]Hannes Kruppa, Modesto Castrillon Santana and Bernt Schiele, “Fast and Robust Face Finding via Local Context”, 2003.

[Lienhart02] Rainer Lienhart and Jochen Maydt, “An Extended Set of Haar-like Features for Rapid Object Detection”, 2002.

[Lucas81] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision”, 1981.

[OpenCV] Open Source Computer Vision Library (OpenCV), *http://sourceforge.net/projects/opencvlibrary/*.

[OpenCV Wiki] Open Source Computer Vision Library Wiki, *http://opencvlibrary .sourceforge.net/*.

[Trucco98] Emanuele Trucco and Alessandro Verri,“Introductory Techniques for 3D Computer Vision”,1998.

[Viola01] Paul Viola and Michael Jones “Robust Real Time Object Detection”, 2001.

[Welsh95] G. Welsh and G. Bishop, “An introduction to the Kalman filter”, 1995.

[Zhang00] Z. Zhang, “A flexible new technique for camera calibration”, 2000.

Appendix A. Camera calibration method.

There are several methods to obtain the pin-hole camera model parameters, that is, intrinsic parameters: focal lengths and the origin of the image; and extrinsic parameters: rotation and translation matrix, which relate the camera-reference-frame with an external known reference-frame. OpenCV chose one method which works well on planar objects; this is based on Zhang's method.

The most common planar pattern is a chessboard. This way, by collecting several images of a chessboard and extracting the corner points within the chessboard, it is possible to collect a homography **H** for each view of the pattern. The homography matrix **H** can be set to the camera's intrinsic matrix **M** multiplied by a combination of two rotation matrix columns, $\mathbf{r}_1$ and $\mathbf{r}_2$, and the translation vector **t**, including a scale factor. This yields:

$$H = [h_1 h_2 h_3] = sM[r_1 r_2 t] \quad \textbf{(B.1)}$$

Then $\mathbf{h_1}$, $\mathbf{h_2}$, $\mathbf{h_3}$, can be written as,

$$\begin{aligned} h_1 &= sMr_1 \quad or \quad r_1 = \lambda M^{-1} h_1 \\ h_2 &= sMr_2 \quad or \quad r_2 = \lambda M^{-1} h_2 \\ h_3 &= sMt \quad or \quad t = \lambda M^{-1} h_3 \end{aligned} \quad \textbf{(B.2)}$$

Here λ =1/s,

The vectors, $\mathbf{r}_1$ and $\mathbf{r}_2$, are orthonormal, this implies:

$$r_1^T r_2 = 0, r_1^T r_1 = r_2^T r_2 \quad \textbf{(B.3)}$$

Using these properties of orthonormality, in expression (B.2), the following constraint can be written,

$$\begin{aligned} h_1^T M^{-T} M^{-1} h_2 &= 0 \\ h_1^T M^{-T} M^{-1} h_1 &= h_2^T M^{-T} M^{-1} h_2 \end{aligned} \quad \textbf{(B.4)}$$

Let define **B**, as:

$$B = M^{-T}M^{-1} = \begin{bmatrix} \frac{1}{f_x^{\,2}} & 0 & \frac{-c_x}{f_x^{\,2}} \\ 0 & \frac{1}{f_y^{\,2}} & \frac{-c_y}{f_y^{\,2}} \\ \frac{-c_x}{f_x^{\,2}} & \frac{-c_y}{f_y^{\,2}} & \frac{c_x^{\,2}}{f_x^{\,2}} + \frac{c_y^{\,2}}{f_y^{\,2}} + 1 \end{bmatrix}$$

(B.5)

Using **B,** considering that matrix **B** is symmetric, the generic term $\mathbf{h_i^T B h_j}$, can be written,

$$h_i^T B h_j = v_{ij}^T b = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix}^T \begin{bmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \\ B_{23} \\ B_{33} \end{bmatrix}$$

(B.6)

Using this definition for $\mathbf{v_{ij}^T}$, the constraints (B.4) may be expressed as:

$$\begin{bmatrix} v_{12}^{\,T} \\ (v_{11} - v_{22})^R \end{bmatrix} b = 0$$

(B.7)

If **K** images of the pattern have been collected, it is possible to stack **K** of these equations together,

$$Vb = 0$$

(B.8)

The camera intrinsic parameters can then get the closed-form solution of the matrix **B**:

$$\lambda = B_{33} - (B_{13}{}^2 + c_y(B_{12}B_{13} - B_{11}B_{23}))/B_{11}$$
$$f_x = \sqrt{\lambda / B_{11}}$$
$$f_y = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}{}^2)}$$
$$c_x = -B_{13}f_x^2/\lambda$$
$$c_y = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}{}^2)$$

(B.9)

The extrinsic parameters are then computed from the equations (3.2), and $\mathbf{r_3}$, can be computed as the cross product of $\mathbf{r_1}$ and $\mathbf{r_2}$.

Appendix B. Kalman Filter.

The Kalman Filter was introduced by Rudolph E. Kalman in 1960, he described a recursive solution to the discrete-data linear filtering problem. The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator, which is optimal in the sense that it minimizes the estimated error covariance, when the presumed conditions are met.

The Kalman filter tries to estimate the state $x \in \Re^n$ of a discrete time process, which is governed by the linear stochastic difference equation,

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \quad \textbf{(B.1)}$$

With a measurement $z \in \Re^m$ that is,

$$z_k = Hx_k + v_k \quad \textbf{(B.2)}$$

Where w_k represents the process noise and v_k represents the measurement noise. Both are assumed to be independent, white, and with normal probability distributions,

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R) \quad \textbf{(B.3)}$$

The *n x n* matrix *A* in the expresion (B.1) is known as the transition matrix and relates the state at *k-1* to the state at *k*, in the absence of noise. The *n x l* matrix *B* relates the optional control input $u \in \Re^l$ to the state *x*. The *m x n* matrix *H* in the equation (B.2) relates the state *x* to the measurement *z* at time *k*. All the matrices might change with each time step or measurement, but they are assumed constant.

Let define $\hat{x}_k^- \in \Re^n$ as a priori state estimate at step *k*, given knowledge of the process prior to step *k* and $\hat{x}_k \in \Re^n$, as a posterior state at step *k* given measurement *z* at step *k*. Then the a prior and a posterior estimate errors can be defined as,

$$e_k^- = x_k - \hat{x}_k^-,$$
$$e_k = x_k - \hat{x}_k \quad \textbf{(B.4)}$$

The estimate error covariances are then,

$$P_k^- = E[e_k^- e_k^{-T}],$$
$$P_k = E[e_k e_k^T] \quad \textbf{(B.5)}$$

The following expression computes a posterior state estimate, as a linear combination of a priori estimate and a weighted difference, between an actual measurement and a measurement prediction,

$$\hat{x}_k = \hat{x}_k^- + k(z_k - H\hat{x}_k^-)$$
(B.6)

In this expression, the *n x m* matrix *k* is the gain or blending factor, which minimizes the posterior error covariance. This minimization can be accomplished by substituting equation (B.6) into the posterior error definition (B.4) and then into the posterior covariance (B.5), taking the derivative of the result with respect to *K,* and setting the result equal to zero, we obtain,

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$
(B.7)

Equation of the Kalman filter can be divided into two groups,

- *Time update equations*. They are used to estimate the process state at some time, so they are responsible for projecting forward the current state and error covariance estimates to obtain the a priori estimates for the next time step. These equations can also be thought of as *predictor* equations.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$
$$P_k^- = AP_{k-1}A^T + Q$$
(B.8)

- *Measurement update equations*. After the estimation process, they obtain feedback in the form of measurements. These equations can be thought of as *corrector* equations.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$
$$P_k = (1 - K_k H)P_k^-$$
(B.9)

The first task during the measurement update is to compute the Kalman filter gain, K_k, then the measure of the process is obtained, z_k, and then a posterior state is estimated by incorporating the measurement. The final step is to obtain a posterior error covariance estimate.

After each time and measurement update pair, the process is repeated with the previous posterior estimates used to project or predict the new a priori estimates.

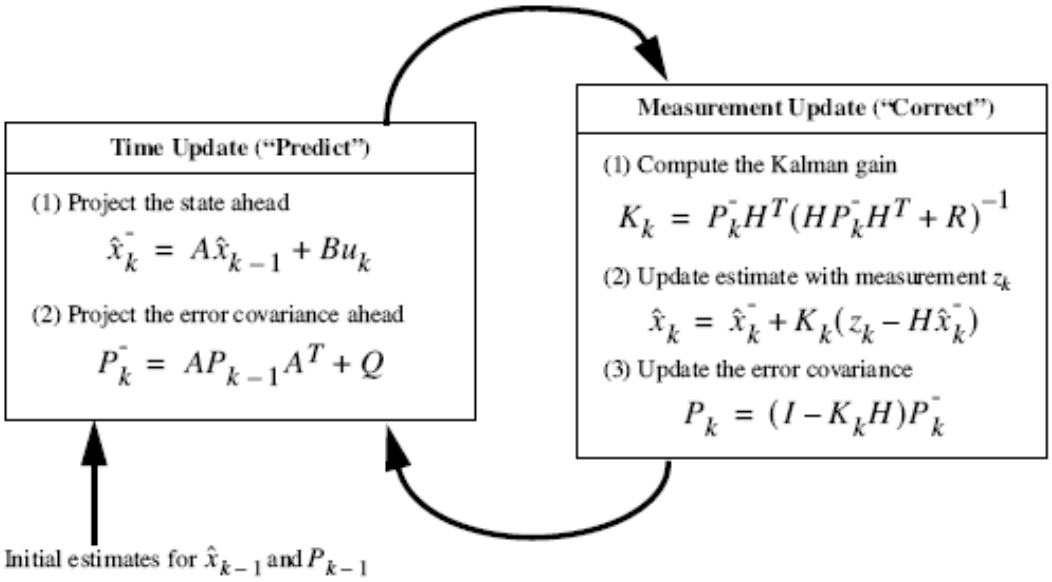


Figure B.1. Scheme of the Kalman filter algorithm. Scheme taken from [Welsh95]

Appendix C. Applications.

This appendix is a manual, which describes the different applications developed for this thesis.

Camera Calibration.

This application has been developed to compute the pin-hole model camera parameters, the intrinsic matrix, the rotation matrix, the translation vector and the distortion coefficients.

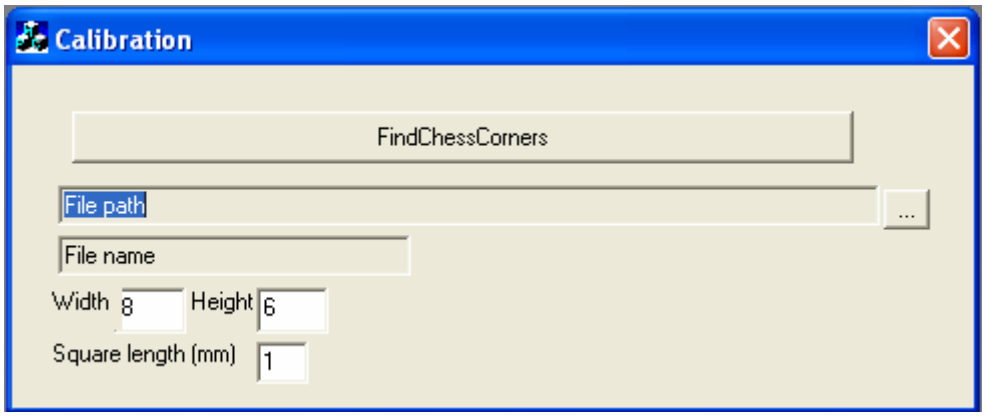


Figure C.1. Camera Calibration GUI

Before running this program, several images of a chessboard should be taken and saved in a directory (at least 12 images of an 8 by 6 chessboard pattern). This application only saves the rotation and translation matrices for the main image, the one which is selected clicking the button "…". The application uses all the images which are in the same directory with the same name, followed by a specific number (less than 100) to compute the parameters.

The inputs are:

- Width: Number of interior corners in the rows of the chessboard pattern.
- Height: Number of interior corners in the columns of the chessboard pattern.
- Square length: Length of a square of the chessboard expressed in millimetres.

The application saves automatically the parameters in "xml" files, in the same directory as the pattern images, and an image of the main pattern with the chessboard-reference frame represented on it (see Figure C.2).

Figure C.2. shows the projection of the computed reference-frame.

The button “FindChessCorners”, let you check if the corners of the chessboard were successfully detected, if there is any invalid image it can be deleted, and the process must be repeated (see Figure C.3).

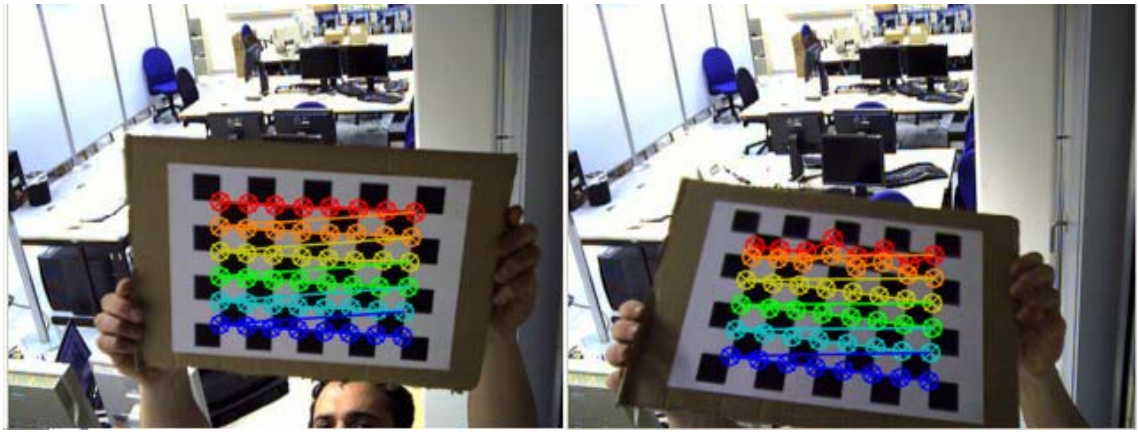


Figure C.3. shows a correct detection (left) and a wrong (right) of the chessboard corners.

Draw Isometric.

The following program is a simple application, which computes the location and draws the isometric projection of the camera-reference-frames in the world-reference-frame, given the translation and rotation matrices which relate:

- The camera-reference-frame and the pattern-reference-frame (computed with the previous application).

- The world-reference-frame and the pattern-reference-frame. To obtain these matrices, some measurements must be accomplished.

*To select the matrix click the button “Load matrix”.

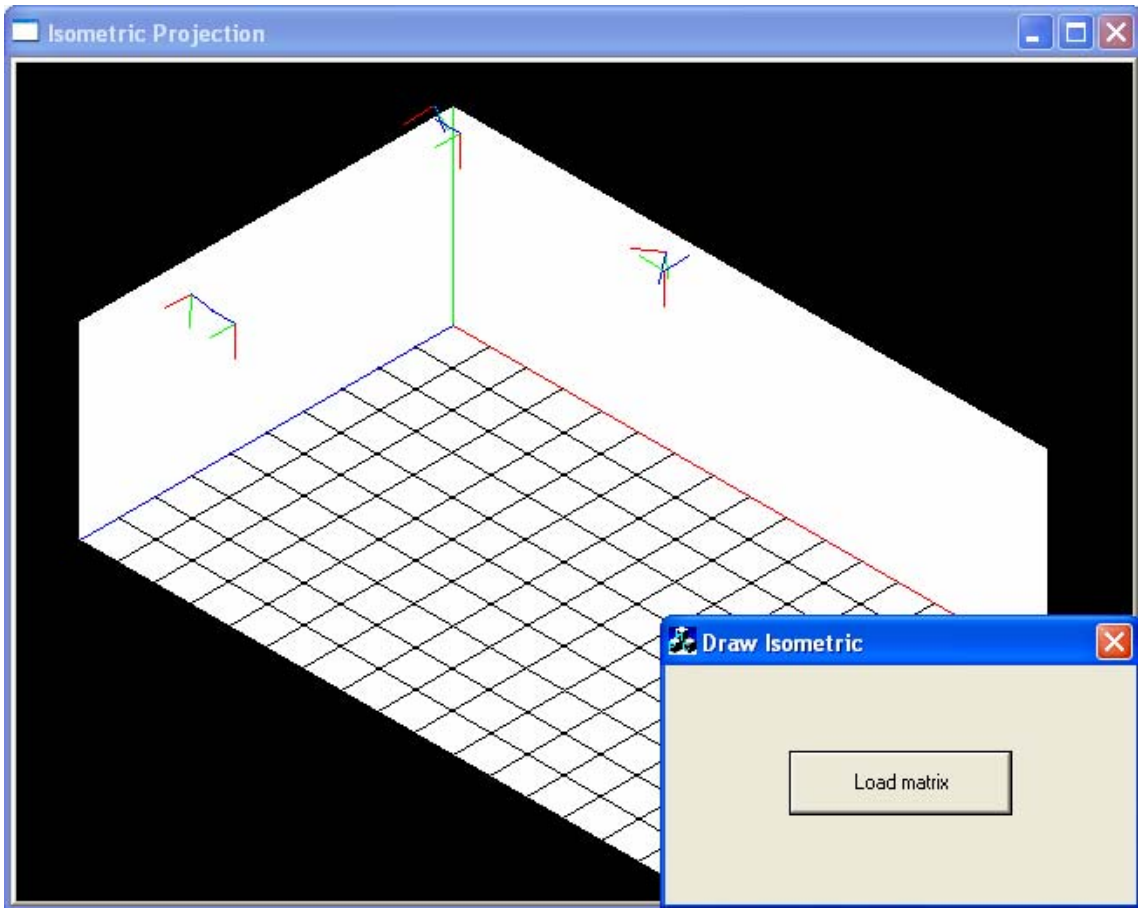


Figure C.4. Draw Isometric GUI.

Applications to compute the 3D location.

There are two applications which compute the three-dimensional location of the people. One of them uses the Motion Templates Method to obtain the two-dimensional location of people, and the other uses the Background Subtraction Method. However their graphical user interfaces are the same.

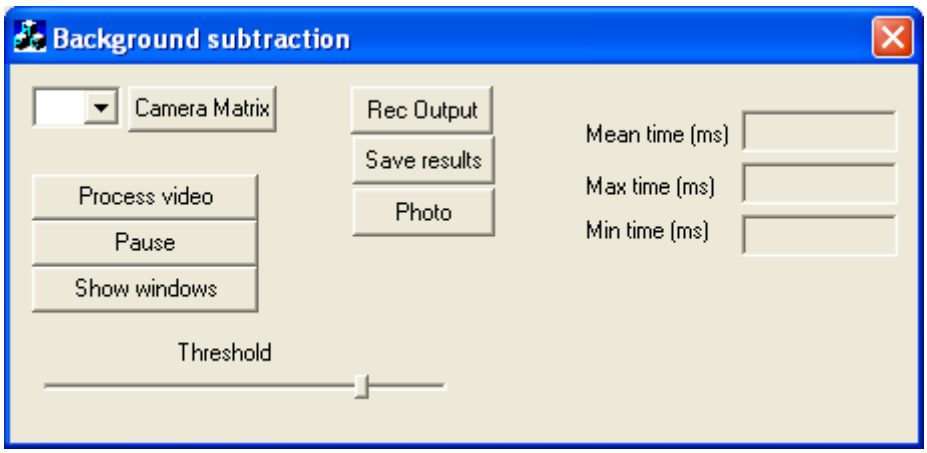


Figure C.5 Generic GUI for 3D location

The size of the videos used for this application must be 320*n x 240, where n is the number of cameras used to record the videos. The videos must be uncompressed.

How to make the program work:

- First, select the number of cameras in the combo box. This number must be less or equal to the number of cameras used to record the opened video. If the number of cameras selected is less than the number of cameras used to record the video, only the first number of cameras selected will be taken into account.

- Once the number of cameras is selected, it is time to introduce the camera matrix. By clicking the “Camera Matrix” button, a file dialog box will appear to choose the cameras matrices (they must be in xml files).

- After selecting the camera matrices, you can choose the video to process clicking in the button “Process video”.

- The “Pause” button allows you to stop the video sequence.

- The “Show windows” button opens the output windows.

- The “Rec Output” check box must be remained pressed, before the video is processed, to record the output windows in a video file.

- The “Save results” check box must be pressed, before the video is processed. This records the measured three-dimensional coordinates of an only object.

- The “Photo” button captures the output windows and allows you to save them in the desired directory.

- The threshold slider allows you to modify some internal parameters; its position should not be modified.

- The mean, max and min runtime per frame of the detection algorithm is shown in the right text boxes.

MOBOTIX-WEBCAM Record application.

This application has been developed to record simultaneously images from two IP-Mobotix cameras and an optional WebCam. The format videos recorded are uncompressed. The application also saves a “.dat” file with the exact timestamp of each frame. Before beginning to record the videos:

- The IP-address of the desired IP-cameras must be set.

- The frame rate must be chosen in the combo box.

- The destination file of the video file must be specified.

Figure C.6. REC application GUI.