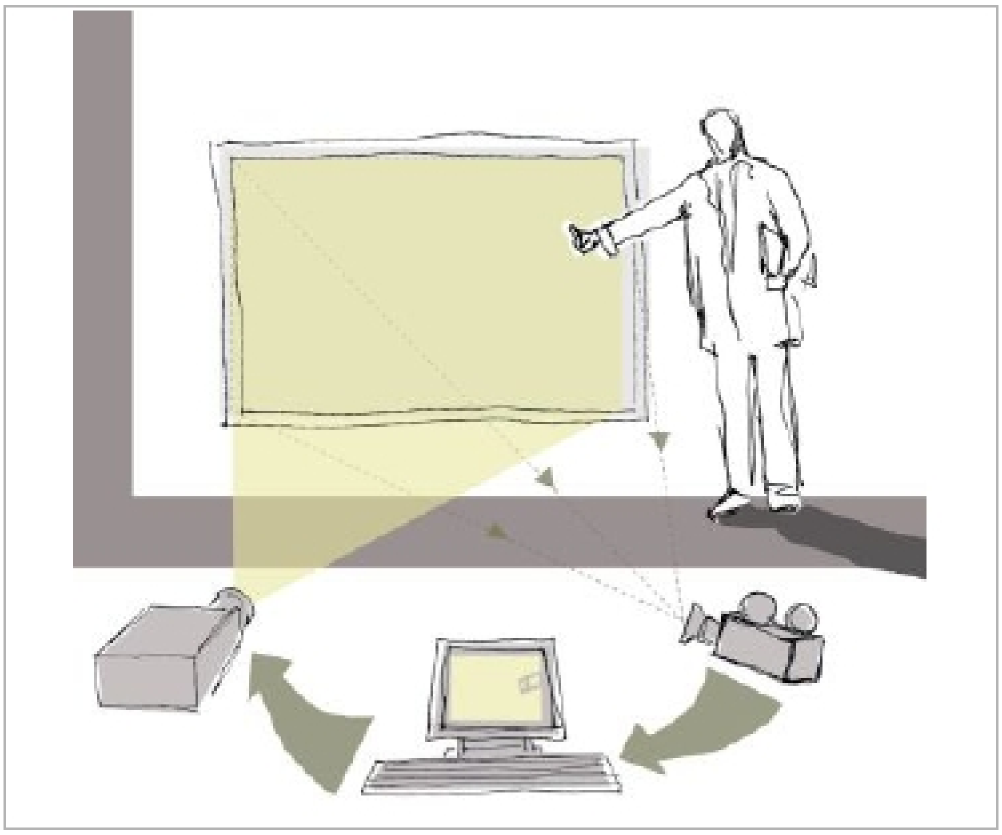
Sheffield Hallam University

Universidaad de Córdoba

MIMAS CALIBRATED CAMERA-PROJECTOR SYSTEM FOR FINGER TRACKING



Artworks by Claudia Amico

Final Year Project
Author: Juan Roldan Ruiz
Supervisors: Dr. Bala P. Amavasai & Jan Wedekind
January 2007

Contents

Acknowledgements

I really would like to express my deep gratitude to Sheffield Hallam University and Escuela Politecnica Superior de Córdoba of giving me the opportunity of making such an interesting and challenging project and the entire professional and personal experience that it involves, and to all the *MMVL* staff, and to Julien Faucher for his priceless mathematical support.

I specially would like to thank to a person with such a personal and professional quality who has accompanied me during the whole project; without him, this project would not have come true. I would like to thank very specially to Jan Wedekind for supporting me during the whole process of the research with his infinite patience and amazing sense of humour and his cooking skills (although he does not like Jamón Serrano).

And finally an extra and very special thanks goes to Claudia Amico for being such an incredible person, and for sharing with me so good vibras. And also for her amazing artworks!

Chapter 1

Introduction

1.1 Motivation

The implementation and potential of Computer Vision [10] and Image Processing [3] is becoming increasingly appreciated in the fields of Information Technology and Artificial Intelligence, due to the advances accomplished in the integration of these areas of knowledge.

In the *Micro-systems and Machine Vision Laboratory (MMVL)* in Sheffield Hallam University, Computer Vision researches are continuously developed with the aim of continue deeping in this challenging area of Computer Science.

This project offers the possibility of converting a planar surface into a tactile board, by replacing the use of traditional devices such as the keyboard and mouse by a more interactive interface.

The aim of this project is to build an interactive camera-projector system capable to perform a pointer tracking whilst it is moved in front of a planar surface such as a white screen, a blackboard or a regular wall.

1.2 State-of-the-art

Similar researches to this project have been developed and have been used as a base for this research. Such as the Zhengyou Zhang project [13] who proposes a flexible new technique for camera calibration, the Magic-Board project [4] that stimulated the idea the camera projector system or the Edward Tse's work [12], which is focused on supporting people's natural interactions over digital surfaces such as large tables and wall display

The search of a more interactive interface between machine and human continues producing other developed camera projector system projects similar to this one, and the way of implement them is also a challenging task, in terms of calibrating the system [13] or finding different ways of tracking an object in a scene [8].

Chapter 2

Specifications

2.1 Hardware set-up

The proposed system can be described as a still camera and a video projector, both of them connected to a workstation and oriented toward a planar surface (called panel from now on).

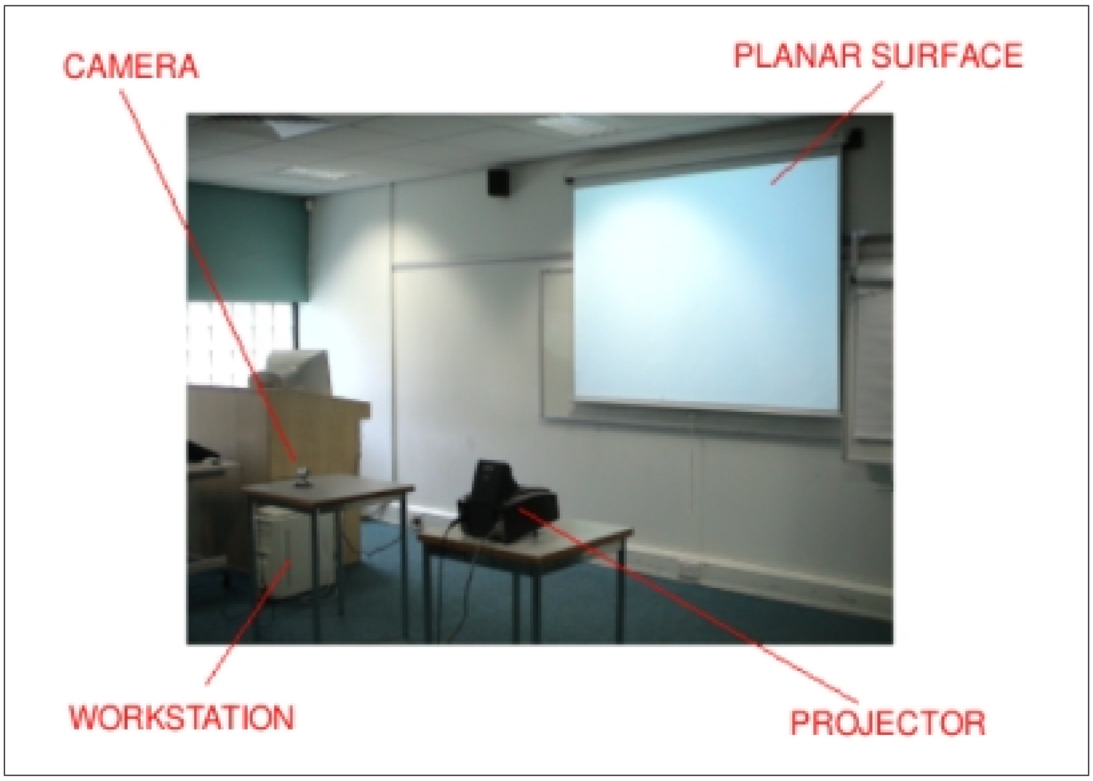


Figure 2.1: *System Components*

The workstation is an Intel(R) Pentium(R) 4 at 2.50GHz with a 256 Mb main memory card and also equipped with a graphic card in order to send the output to the projector. The projector (Figure 2.2) is a CTX EzPro 550 video Projector with a 800x600 pixels resolution working with a frequency of 72Hz which is attached to the serial port of the computer.The camera used (Figure 2.3) is a Logitech Quickcam Pro 4000 with a 640x480 pixels resolution and is plugged-in to a 2.0 USB port of the computer.

Figure 2.2: *CTX EzPro 550 video Projector*

Figure 2.3: *Logitech Quickcam Pro 4000 Camera*

2.2 Software

The operative system which this application has been developed and tested under is a *10.0 SUSE GNU/Linux* distribution and all software applications, toolkits and libraries used during the whole development process of this project are open source software.

This application has been implemented using the computer vision library *MIMAS* and all the libraries that it requires, such as *LAPACK* for mathematical calculations or graphical displaying. For the graphical interface the *Qt4* technology has been used.

2.3 System description

The developed software creates an interactive projector-camera system which is suitable to be controlled avoiding the use of traditional devices such as the keyboard and the mouse. Moreover, the system can reproduce the behaviour of these peripherics.

This system will be able to track a pointer moving in front of the panel, to know its relative position in it and to get the corresponding position in the screen of the computer (called screen from now on). So for each position of the pointer in the panel there will exist a corresponding position in the screen.

In the hypothetical case where the camera and the projector lens were exactly in the same position, were oriented toward the same direction and their frustum had exactly the same values, then no translation would not be needed, because the projected image from the projector would be exactly the same than grabbed one by the camera. But as a result of the location and orientation differences between the camera and the projector lens, the relation in the *2D* space between their positions (and orientation) and the position of the panel should be calculated with the purpose of knowing the relative position of the object in the panel. The region of the panel where the pointer is going to be tracked is called *Panel Active Area (PAA)* which will correspond to the *Screen Active Area (SAA)* of the screen.

Chapter 3

Design

3.1 System initialization

First of all, the initialization of the system must be done, which basically means the execution of two steps: placing the system components in the scene and performing the calibration of the system. The former one does not require any technical specification as the relative positions of the camera, project and screen are completely arbitrary. The only aspect to be considered when placing the camera is the fact that, as the pointer must always be visible for the camera, no object should interpose between the camera and the pointer. Thus, if the camera is very angled placed with the screen, the pointer holder's body may hide the pointer to the camera if it has been placed in the same side of the screen where the pointer holder used is in that moment.

Once the projector and the camera are set in a still position pointing to the panel, the calibration can be made.

3.2 Camera-projector calibration

The goal of the camera-projector calibration is to find the relation between the panel position and the projector and camera positions in the space, which means the relation between the coordinates of the points projected from the projector to the panel and the observed points, which are the ones grabbed by the camera. So in order to get that relation between the 2D scene and the image [11], the plain to plain homography matrix [6] is required.

As can be seen in Figure 3.1, the homography H transfers the points from one view to the other. In this system, that transfer

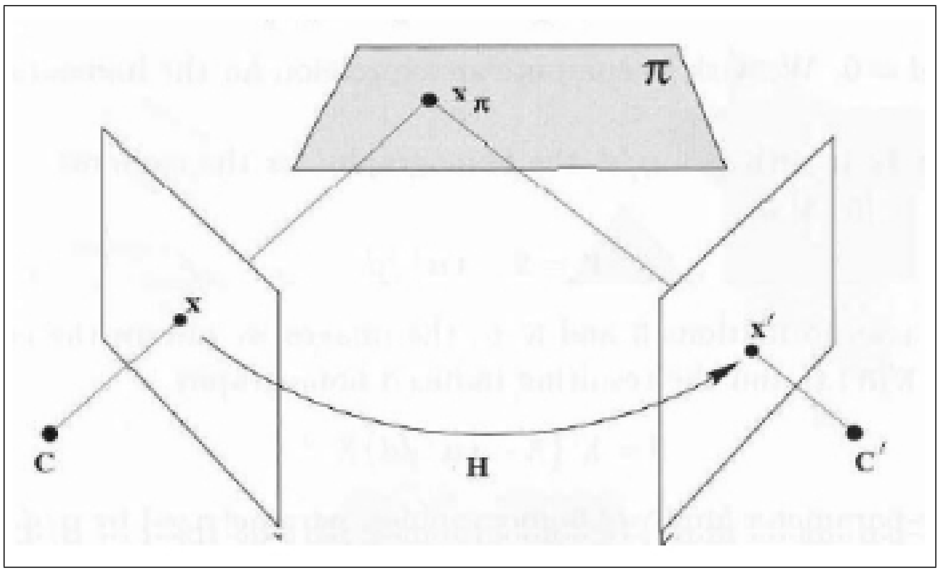


Figure 3.1: *Plain to plain Homography*

will be made from the points projected by the projector and the points observed or grabbed by the camera.

3.2.1 Homography matrix

The mathematical process to obtain the homography matrix [9] has been developed as follows:

As can be seen in Figure 3.2 , is the camera centre and the point X on the world plane is imaged as x in the image plane. Euclidean coordinates (X, Y) and (x, y) are used for the world and image planes, respectively.

The world and image points are represented by homogenous coordinates (3-vector) as:

$$X = (X, Y, W)^T \tag{3.1}$$

$$x = (x, y, 1)^T \tag{3.2}$$

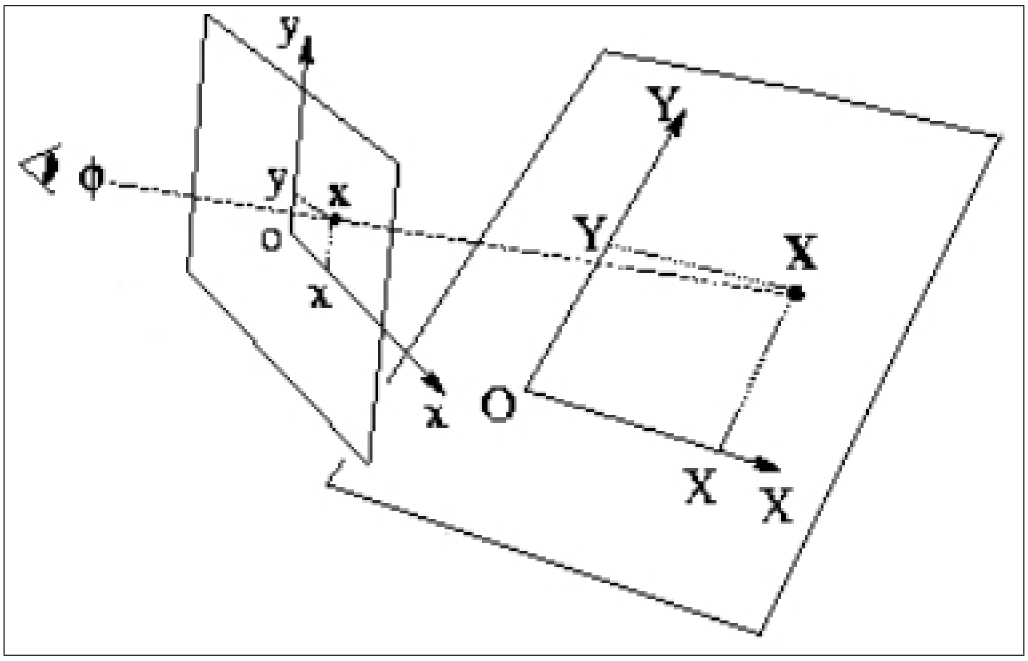


Figure 3.2: *Image plane and world plane*

The scale of the matrix does not affect the equation, so only the eight degrees of freedom corresponding to the ratio of the matrix elements are significant.

Camera model is completely specified once the matrix is determined. The matrix can be computed from the relative positioning of the two planes and camera centre. However, it can also be computed directly from image to world point correspondences.

$$\lambda\, x' = H\, x \tag{3.3}$$

Two-dimensional version of the plane to plane homography where H is a 3×3 Homography matrix can be seen in the equation 3.4.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{3.4}$$

$$H \in \mathbb{R}^{3\times 3} \tag{3.5}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.6}$$

From equation 3.6, each image to world point correspondence provides two equations linear in the H matrix elements. For n correspondences we obtain a system of $2n$ equation in 8 unknowns. If $n = 4$ then an exact solution is obtained. Otherwise, if $n > 4$, the matrix is over determined, and H is estimated by a suitable minimisation scheme.

The covariance of the estimated H matrix depends both on the errors in the position of the points used for its computation and the estimation method. The homogenous solution has been the standard method used for estimating H. The solution is obtained using *Single Value Decomposition (SVD).*

Writing the H matrix in vector form as:

$$H = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T \tag{3.7}$$

The homogenous equations for n points become $Ah = 0$. Being A the $2n \times 9$ matrix:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 & -X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 & -Y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 & -X_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 & -Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nX_n & -y_nX_n & -X_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nY_n & -y_nY_n & -Y_n \end{pmatrix} \tag{3.8}$$

It is a standard result of linear algebra that the vector h that minimises the algebraic residuals $\mid \Delta h \mid$, subject to $\mid h \mid = 1$, is given by the eigenvector of least eigenvalue of A^TA. This eigenvector can be obtained directly from the SVD of A. In the case of $n = 4$, h is the null-vector of A and the residuals are zero.

3.2.2 Pattern recognition by cross-correlation

Correlation is a robust and general technique for pattern recognition and is used in many applications, such as automatic target recognition. To locate a point in an image the use of patterns recognition has been made, and in order to achieve this recognition, cross correlation technique has been applied.

Cross correlation [1] [5] is a standard method of estimating the degree to which two series are correlated. Considering two series $x(i)$ and $y(i)$ where $i = 0, 1, 2 \cdots N - 1$, the cross correlation r at delay d is defined in equation 3.9.

$$r = \frac{\sum\limits_i \left[(x(i) - mx) \cdot (y(i-d) - my) \right]}{\sqrt{\sum\limits_i (x(i) - mx)^2} \sqrt{\sum_i (y(i-d) - my)^2}} \tag{3.9}$$

Where mx and my are the means of the corresponding series. If the above is computed for all delays $d = 0, 1, 2 \cdots N - 1$ then it results in a cross correlation series of twice the length as the original series.

$$r(d) = \frac{\sum\limits_{i} \left[\left(x\left(i\right) - mx \right) \cdot \left(y\left(i - d\right) - my \right) \right]}{\sqrt{\sum\limits_{i} \left(x\left(i\right) - mx \right)^2} \sqrt{\sum_{i} \left(y\left(i - d\right) - my \right)^2}} \tag{3.10}$$

Thus, the ranges of the delay d will be between 0 and N, which is the number of points. This process of pattern identification inside the image uses cross correlation of the image with a suitable mask, when the mask and the pattern being sought are similar the cross correlation will be high. The mask is itself an image which needs to have the same functional appearance as the pattern to be found. The mask is centered at every pixel in the image and the cross correlation calculated, this forms a *2D* array of correlation coefficients.

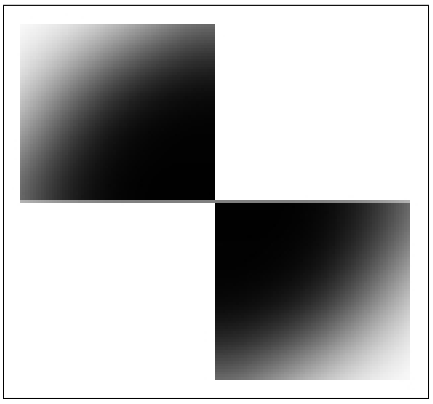


Figure 3.3: *Used pattern*

When this searching process is done, a grey-level image is generated, where white pixels will mean high correlation and black pixels will mean low correlation.

The correlation method has not been implemented in a straightforward way due to it does not work efficiently in terms of distortion permission and speed. Straightforward correlation works well only when the grabbed image (which is the test scene in this case) contains an exact copy of the reference image. Moreover, it is highly sensitive by the noise produced by illumination changes or viewing geometry changes such as rotations or scale changes. This is why a more efficient method had been used, an efficient method based on Fast Fourier Transforms in order to compute the discrete Fourier transforms. The aim of this choice is to reduce the number of computations needed for making the correlation method.

3.2.3 Obtaining the point pairs

Once the the Homography matrix structure is known, a N number of point pairs are required to find the value of all the unknown variables and consequently solve the equations and get the homography matrix for this particular situation. In this case, five point pairs are needed, according to the explanation given in section 3.2.1.

The process to obtain the five point pairs coordinates is shown in Figure 3.4.

First of all, a set of five different point coordinates in the SAA must be set. Then, for each of these five points P_i (let call them P_i being $i = 1..5$) of the SAA, the procedure to obtain the corresponding five points $P_{i'}$ of the PAA and therefore get the required five point pairs consists of creating a blank image and

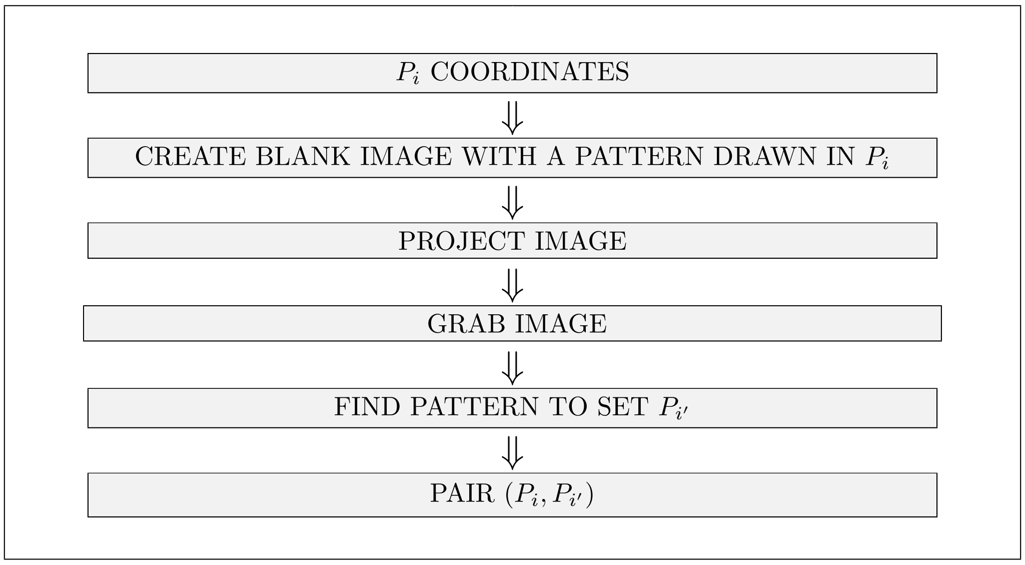

P_i COORDINATES
CREATE BLANK IMAGE WITH A PATTERN DRAWN IN P_i
PROJECT IMAGE
GRAB IMAGE
FIND PATTERN TO SET $P_{i'}$
PAIR $(P_i, P_{i'})$

Figure 3.4: *Procedure to obtain a point pair*

draw the chosen pattern in it in the P_i coordinates so the centre of the pattern coincides with these coordinates.

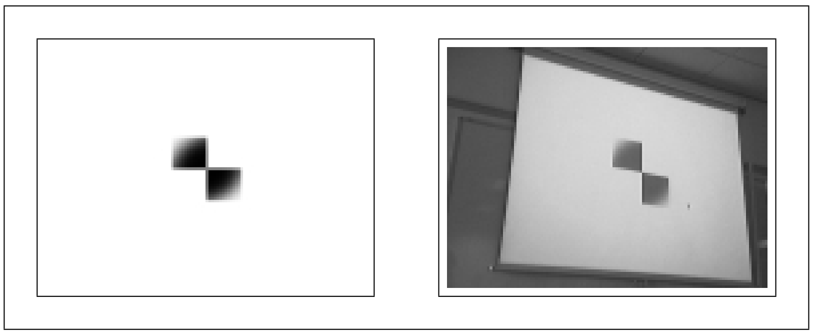


Figure 3.5: *Projected image and grabbed image*

Then, the created image is projected against the planar surface and grabbed with the camera. As can be seen in Figure 3.5, these two images (the projected and the grabbed one) show the pattern in different positions and perspectives. While in the first image the pattern centre is placed in P_1 coordinates, in the second image the pattern centre lies at $P_{1'}$ coordinates.

Finally, the grabbed image is processed by a pattern recognition procedure in order to know the coordinates of the centre of the pattern in that image, which is $P_{i'}$. Now the retrieval of the point pair $(P_i, P_{i'})$ is achieved and this process will be done once for each P_i point. As Figure 3.6 shows, the pattern is projected five times, once per each $_i$ point. It is projected and grabbed as well an image with no pattern in it in order to make an difference image.

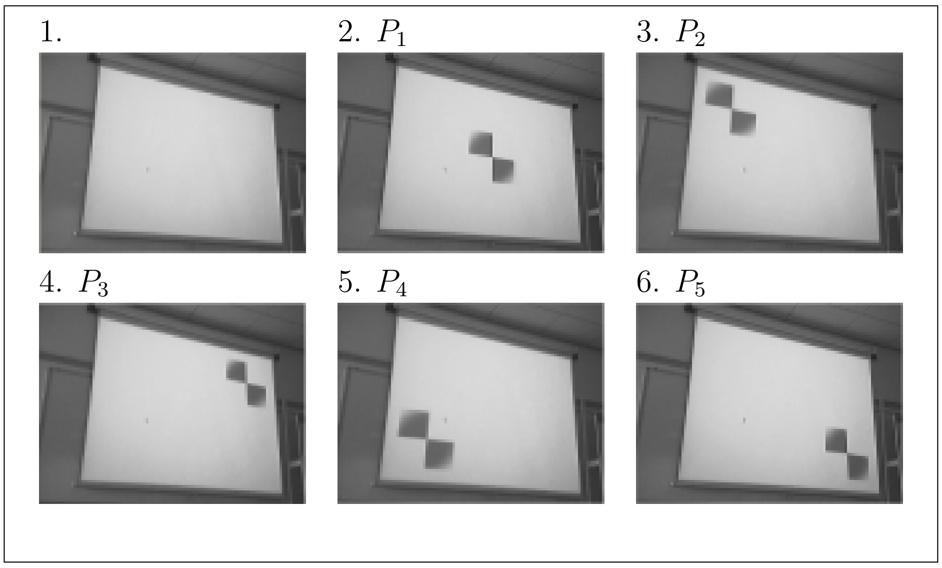


Figure 3.6: *Projecting the patterns*

Every image originated by the difference between each grabbed image (those which contain the projected pattern) and the scene image, that is the grabbed image which no pattern in it, will be the one to be processed by cross-correlation (see Section 3.2.2) with the pattern image in order to search the pattern in it.

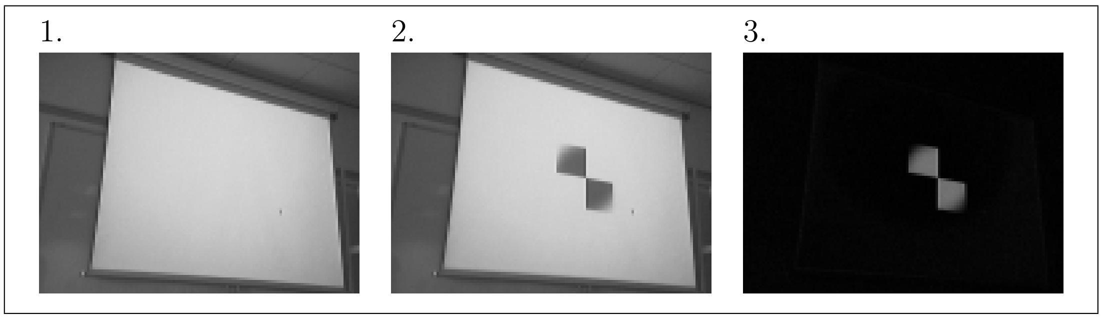


Figure 3.7: *Pattern-scene image difference*

As can be noticed in the difference image (see third image of Figure 3.7), the pattern is shown in negative mode, so the pattern image is also converted to negative mode before making the search (Figure 3.8).

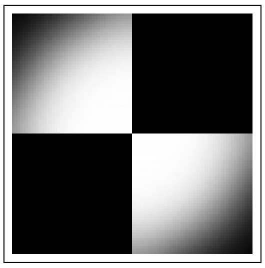


Figure 3.8: Used pattern in negative mode

3.3 Pointer tracking

There are different techniques for recognising an object in a scene, such as cross-correlation, colour segmentation or the comparison of different pictures. The developed system applies the colour segmentation technique because it was proved to be a very capable method for the desired real-time performance and also because its implementation is not as complex as in the other alternatives.

3.3.1 Locating the pointer by colour segmentation

In order to apply the colour segmentation to an image with the intention of separating the desired object from the background, the colour difference between these two elements must be sufficiently noticeable. Thus, the image can be thresholded and a binary image can be generated, excluding the background by giving to the background pixels and to the object pixels different values correspondingly.

The process of recognizing the pointer in the scene is divided in some steps as Figure 3.9 shows.

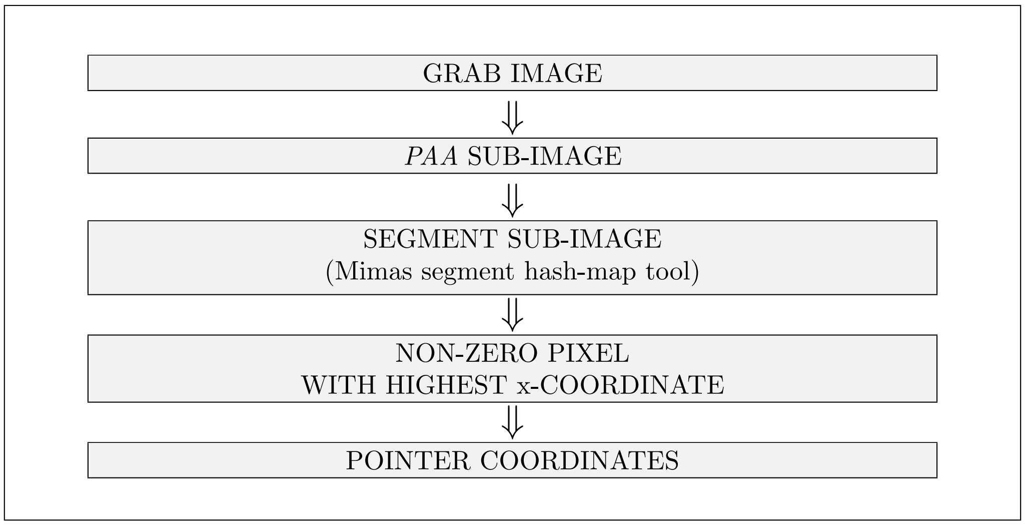


Figure 3.9: *Pointer recognition*

The recognition process begins with grabbing an image with the camera and storing

The object used as the pointer is an orange marker (Figure 3.10), is differentiated from the background applying the colour segmentation (see Figure 3.12).

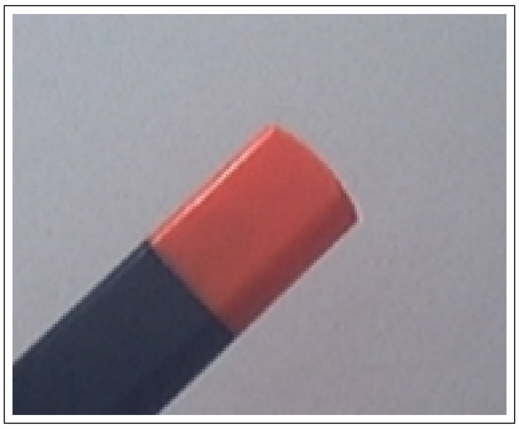


Figure 3.10: *Original image of the pointer*

The Mimas segment hash-map tool was used for the colour segmentation process. A comparison between each pixel of the image and a reference model (Figure 3.11) is performed. Reference image which must be retrieved previously from a taken image of the pointer (Figure 3.10) in the same light conditions that the system is going to be working in afterwards.

Figure 3.11: *Reference image*

This model contains the colours which want to be found in the image as part of the desired object. Each pixel of the image that does not have the same *RGB* colour values of any pixel in the reference model will get a zero value, so it will be seen as a black pixel.

Once the segmented image is retrieved, the location of the pointer is considered as the pixel located at the highest and furthest right in the image that has a non zero value.

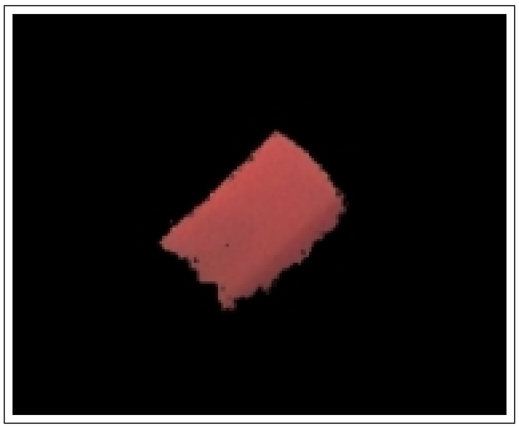


Figure 3.12: *Colour-segmented image of the pointer*

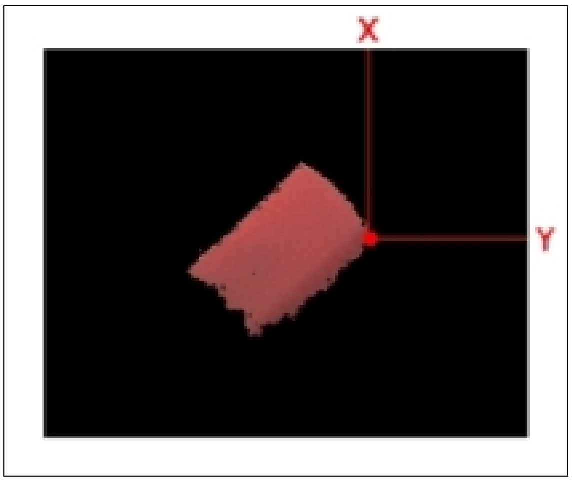


Figure 3.13: *Locating the pointer*

The pointer pixel is not searched in the whole grabbed image (Figure 3.14), only inside a sub image of it, which is delimited by the four corners of the *PAA* (see Figure 3.12), so only that area of the grabbed image is be processed. And each time the pointer is searched, the first step to be done is to know whether the candidate to be the pointer pixel resides inside the *PAA*. The solution for this problem is achieved by computing the corresponding *SAA* coordinates, by applying the inverse of the Homography, to the *PAA* coordinates and checking whether the point lies within the projected image (Figure 3.16).

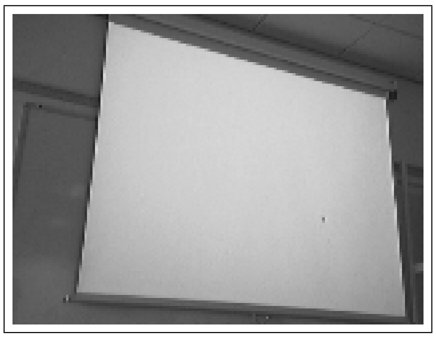


Figure 3.14: *Grabbed image*

As a result of this, all possible noise outside the *PAA* is avoided because no pointer candidate is considered if it lies outside the *PAA*.

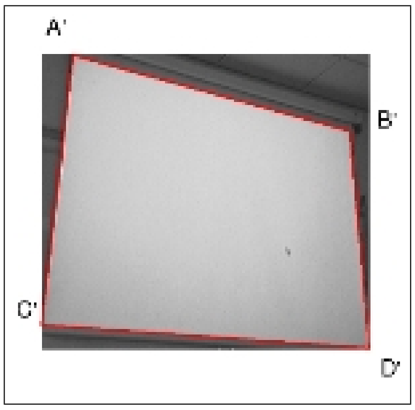


Figure 3.15: *Processed sub image*

The pointer tracking is finally achieved by processing in real time the images grabbed by the camera as previously explained and applying the Homography matrix to the location of the pointer in the grabbed images.

3.3.2 *PAA* to *SAA*

Once the location of the pointer in the grabbed image is determined, the following step is to get its corresponding pixel in

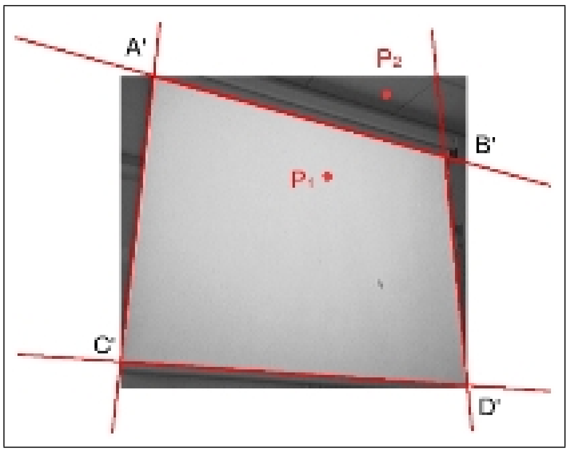


Figure 3.16: *Inside/outside* PAA

the *SAA*. This process is made automatically since the Homography matrix between the image plane (*PAA*) and the world plane (*SAA*) is already known.

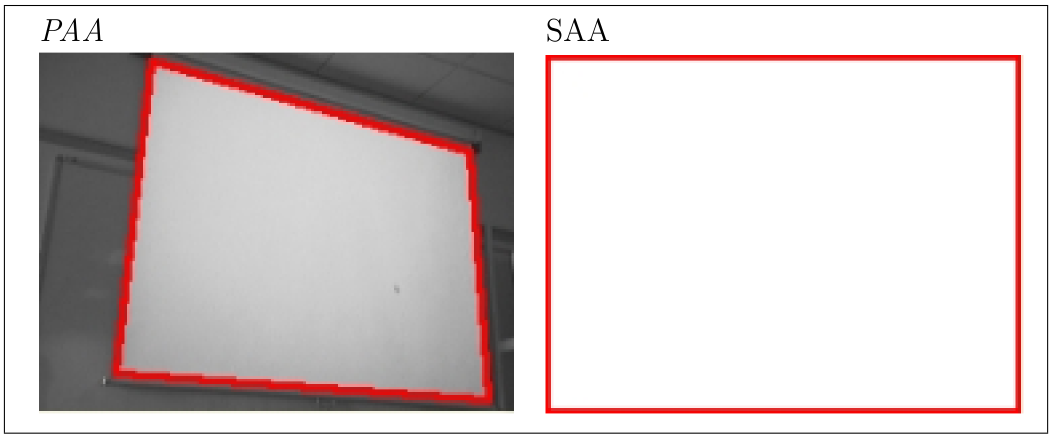


Figure 3.17: *PAA* and *SAA*

For each retrieved pixel in the *PAA*, the equation 3.3 must be used in order to get its corresponding pixel in the *SAA*.

3.3.3 Pointer states

In order to know in real time the actual position of the pointer and a brief history of its location during a certain amount of time, some states have been defined:

- **Not ready**: The system is initialization time and the tracking has not began, so the pointer state can not be determined yet.
- **Not found**: The pointer is not located inside the SAA.
- **Moving**: The pointer has been located inside the SAA. This state is applied to the pointer in two situations.
- **Stopped**: The first one occurs when its current location if far enough from its location history so it can be considered that a noticeable movement has happened. And the second one happens when the pointer has not location history enough in that moment to be compared with its current location, due to the fact that it has not been time enough inside PAA. This state is applied to the pointer when its current location is near from its last measured location history. This nearness is determined by the application of the standard deviation [2] applied to these location values.

As can be seen in Figure 3.18, the first state that can be defined for the pointer is the Not ready state, which will be defined at the initialization of the application. Then, the pointer state will change according to the given previous definitions.

In order to implement the sequence of the changing pointer states, a vector of coordinates is defined to store the produced location history during an established amount of time. For every retrieved point, which is added to the end of the vector, the

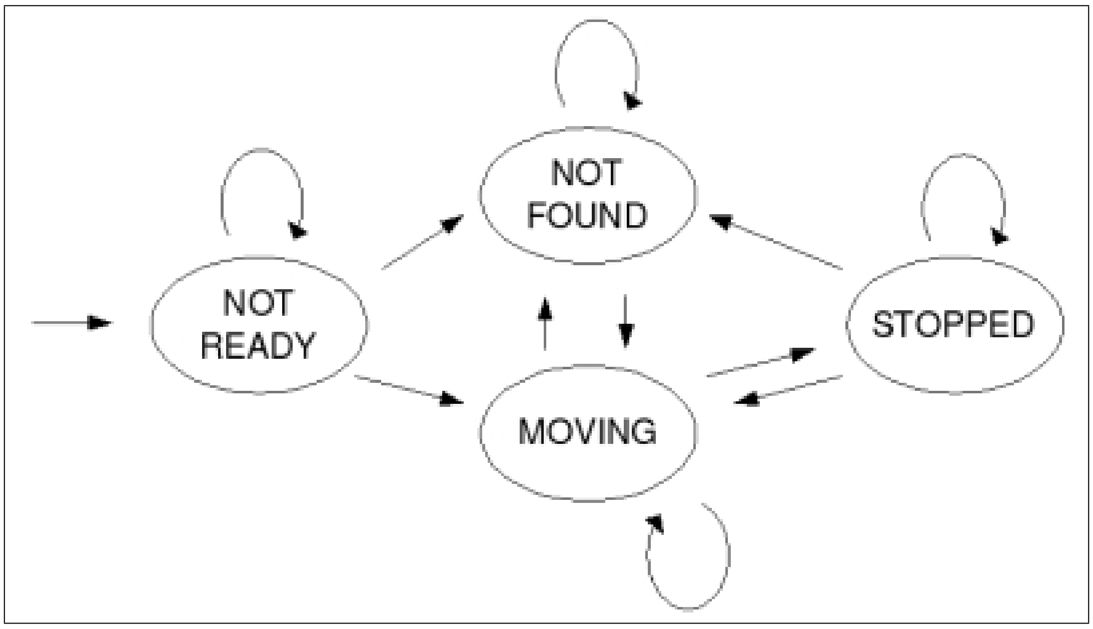


Figure 3.18: *Pointer states*

actual pointer coordinates are stored in this new vector element. The vector will be empty at initialization time and when the actual pointer state corresponds to *Not found*. In this case, the vector is cleared because no valid coordinates can be stored.

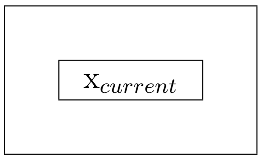


Figure 3.19: *The current point is added to the initial / cleared vector*

A certain amount of time is defined to control the time difference between the moment when the first element of the vector was added and the moment when the new element is going to be added to the vector. As shown in Figure 3.19, if this difference is not as big as the defined time, the only action made to the vector will be the addition of the new element.

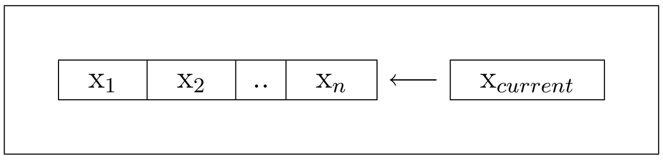


Figure 3.20: *Adding a point to the vector when location history is not enough yet*

In other case, when the time difference is bigger than the defined time, the first element of the vector is removed. As consequence of this, the vector always contains the location history of the pointer location only during the desired amount of time.

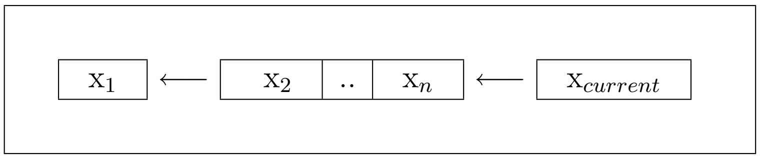


Figure 3.21: *Adding a point to the vector when location history is already enough*

Once the location history of the pointer location is stored, which means that the vector is sufficiently full, the processing of the stored data beings. The actual pointer state is the aim of this processing, which will be *Moving* or *Stopped* depending on the relation among the elements of this vector. This relation corresponds to comparison between the stardard deviation and a threshold of this location history.

The standard deviation of a random variable X is defined as:

$$\sigma = \sqrt{E((X - E(X))^2)} = \sqrt{E(X^2) - (E(X))^2} \tag{3.11}$$

Where *E(X)* is the expected value of X. Knowing that the mean of X is:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + \cdots + x_N}{N} \tag{3.12}$$

The standard deviation can be simplified to:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{3.13}$$

This procedure, instead of simply calculating the arithmetic mean, will reduce considerably the possible noise produced by the peaks in the location history.

Once the *2D* coordinates average of the location history is known, this average is compared with a threshold. If this difference is not as big as a predefined *2D* distance, the actual pointer state is considered as *Stopped*. In other case, the actual pointer position will be defined as *Moving*.

3.3.4 Mouse events

After being capable of knowing in real time the pointer state, the next step is to transfer this knowledge to the mouse control in order to emulate the mouse behaviour and actions, such as moving the cursor inside a screen or making a click event.

This transfer was finally achieved through the use of an *X11* fake events library [7], and the control of the mouse behaviour in a Linux KDE environment was achieved.

POINTER STATE	***MOUSE EVENT***
Not ready	None
Not found	None
Moving	The mouse cursor is updated with the actual pointer coordinates
Stopped	A click event is issued (If no other previous click event has been issued within a certain amount of time)

Table 3.1: Pointer state and mouse event relation

The relation between the actual pointer state an the originated mouse event is described in Table 3.1 .

Chapter 4

Results

4.1 Setting up the system

During the process of setting up the sytem, some aspects about how the system elements are placed in the scene (Figure 2.1) and other system conditions may influence in the way that the system will work, so they should be considered with appropriate attention. For example, the camera should be focused toward the screen so the screen which can be seen in the grabbed image, fills as much as possible that image, considering that the four corners of the screen should also be included in the image.

Another relevant aspect are the light conditions where the system is placed. The darker the environment is, the more likely objects in the foreground will be recognised.

4.2 Applications

As this report has explained, this application has achieved the integration with the *KDE* Desktop. Therefore, any desktop application which can be managed with mouse clicks, cam be also be manipulated through this software. As can be seen in Figure 4.1, the system allows to browse the KDE main menu.

4.2.1 Browsing the file-system

One of the real applications of this software is the task of browsing the file-system. The file brower used was *Conqueror*. As Figure 4.2 shows, tasks such as opening folders and opening files are completely achieved.

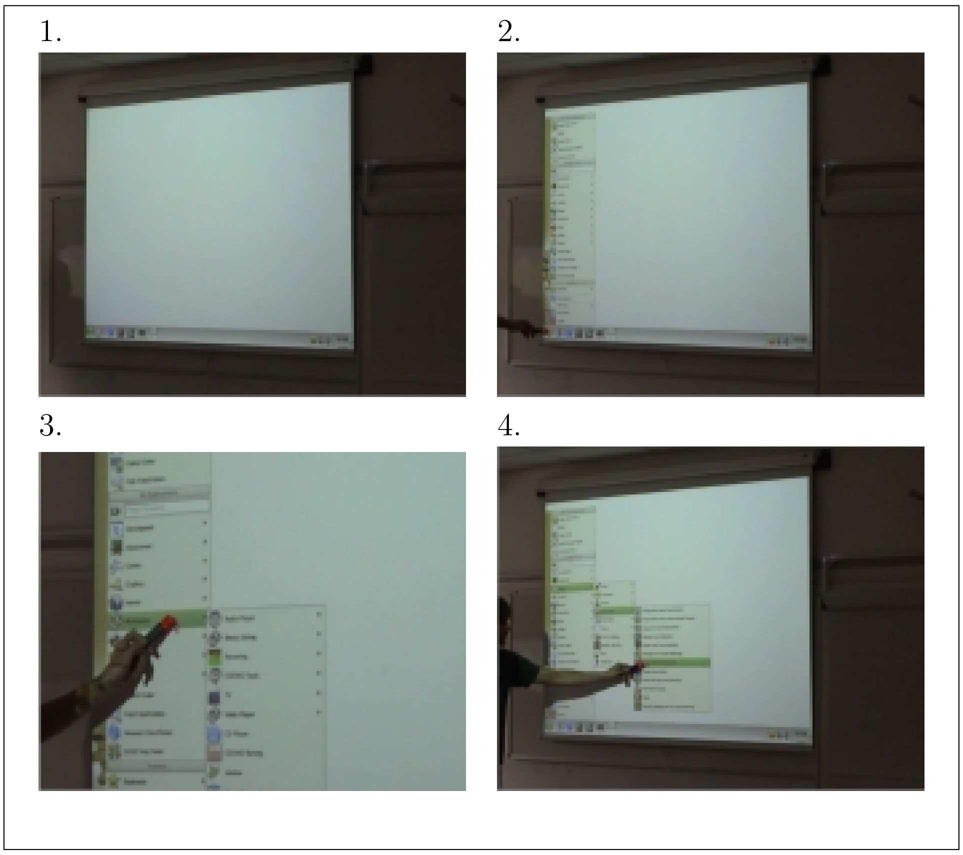


Figure 4.1: *Browsing KDE menu*

4.2.2 Playing Reversi

As Figure 4.3 shows, it is possible to play Reversi. It is another example of how a click-based application can be can be controlled by this interactive system.

4.3 Overall performance and caveats

General performance behaviour has resulted to be quite satisfactory since the system calibration was accomplished successfully and a good level of stability during the pointer traking was achieved.

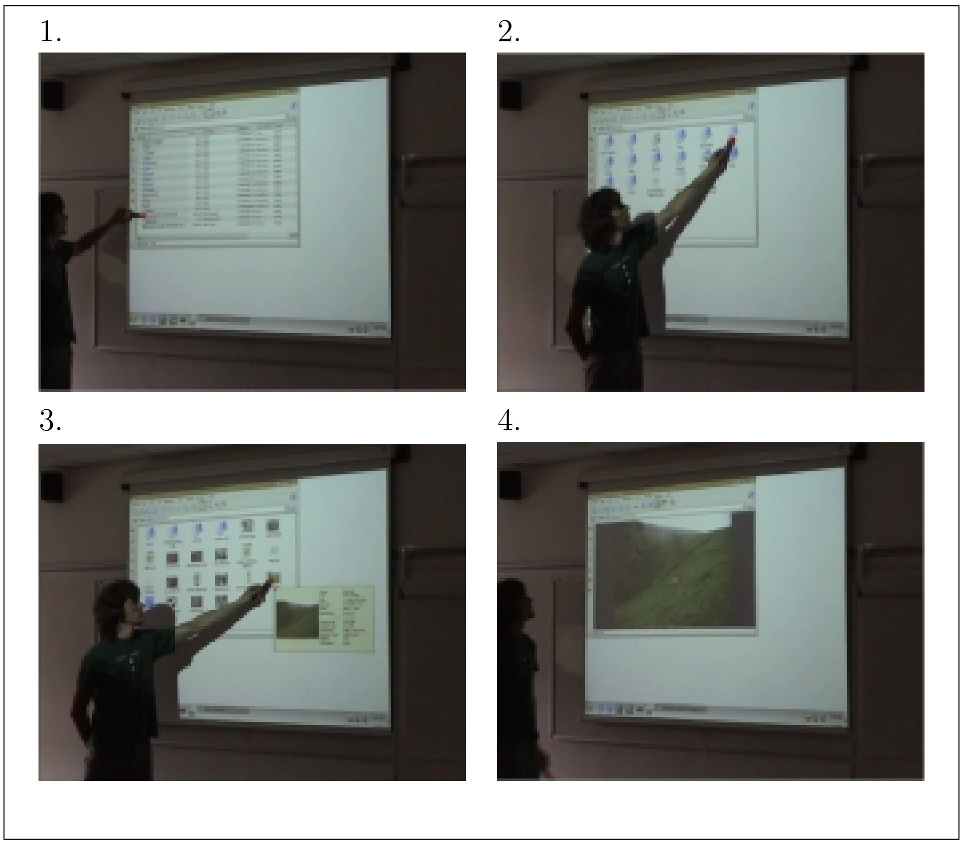


Figure 4.2: *Browsing the file system*

Some caveats:

- Produce all mouse events, such as single and double left click, right click, dragging, mouse wheel click and mouse wheel turns.
- Improvement in the system real-time speed. Maybe by reducing the searching area. Thus, the pointer would only be searched in an small area of interest instead of searching the whole image.
- Allow multiple users (multiple pointers) interact with the system.
- Elimination of the noise produced by background elements

with pixel colour values similar to pointer colour values. Recognizing the pointer not only by its colour but also by its shape may solve the problem.

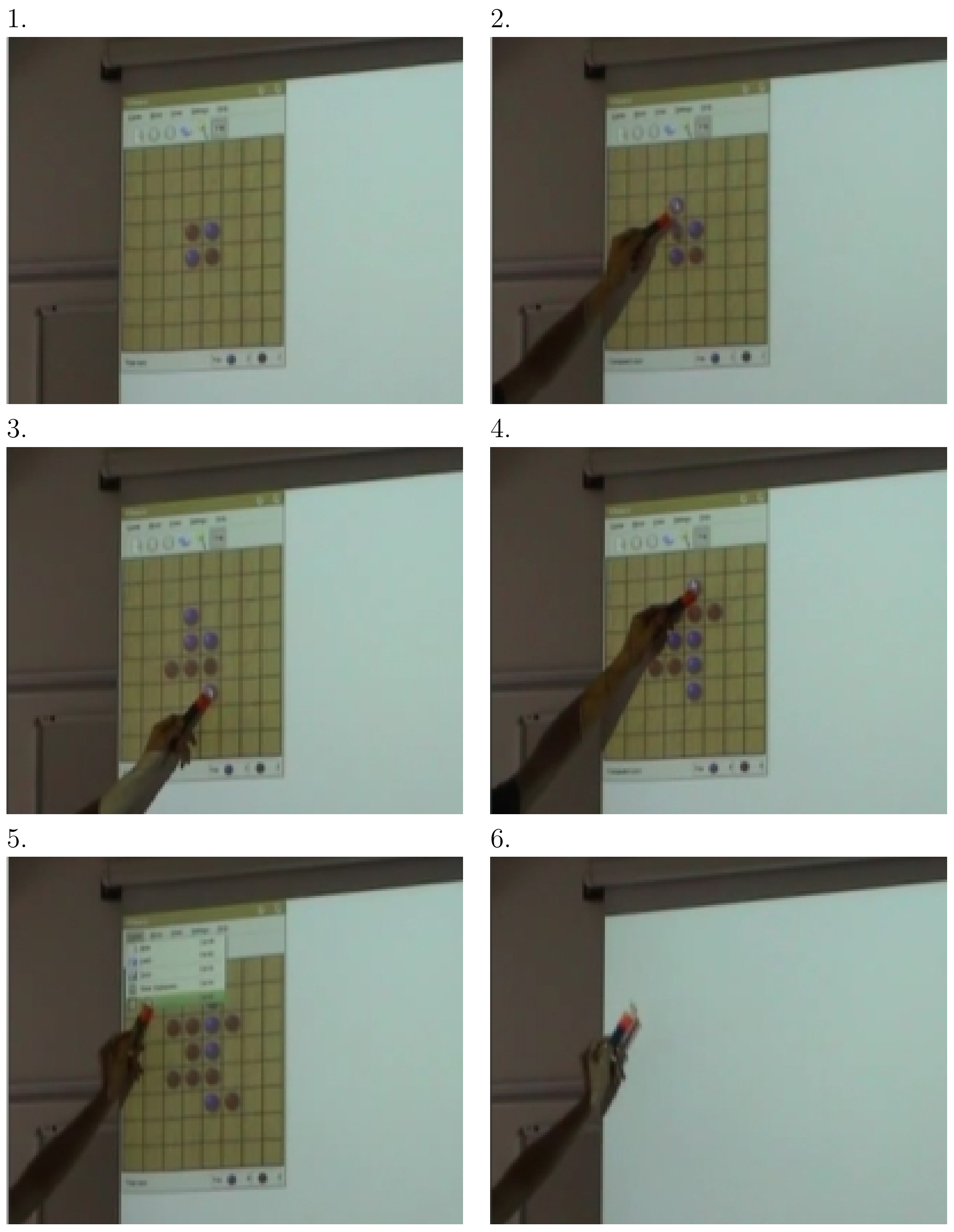


Figure 4.3: *Playing Reversi*

Chapter 5

Conclusions

The elaborated interactive camera projector system offers a wide range of applications such as oral presentations and lectures, and the possibility of converting a regular planar surface into a touching panel just with a PC, a video projector and a webcam. It can even convert a planar TFT screen into a touching panel just by plugging a webcam into the PC and orient it toward that screen.

The fact that the application can be easily integrated into a Linux desktop environment, opens the possibilities of managing desktop applications avoiding the use of the mouse, offering a more interactive interface between human and computers.

Bibliography

[1] Cross-correlation - wikipedia, the free encyclopedia. Website. http://en.wikipedia.org/wiki/Cross-correlation Last accessed December 20th, 2006.

[2] Standard deviation - wikipedia, the free encyclopedia. Website, 2007. http://en.wikipedia.org/wiki/Standard_deviation Last accessed January 4th, 2007.

[3] Low A. *Introductory to computer vision and image processing*. Mc Graw Hill, 1991.

[4] François Bérard. The magic board project. Website. http://iihm.imag.fr/demos/magicboard/ Last accessed April 7th, 2006.

[5] Paul Bourke. Cross correlation. Website. http://local.wasp.uwa.edu.au/~pbourke/other/correlate/ Last accessed June 19th, 2006.

[6] Antonio Criminisi. Computing the plane to plane homography. Website. http://www.robots.ox.ac.uk/~vgg/presentations/bmvc97/criminispaper/node3.html Last accessed Jule 27th, 2006.

[7] Kieron Drake. Test extension library. Website. http://www.xfree86.org/current/xtestlib.html Last accessed June 20th, 2006.

[8] James L. Crowley et al. Finger tracking as an input device for augmented reality. Report, 1995. International Workshop on Face and Gesture Recognition.

[9] Julien Foucher. Mimas camera calibration. Website. http://vision.eng.shu.ac.uk/mediawiki-1.4.10/index.php/Mimas_Camera_Calibration Last accessed June 16th, 2006.

[10] Jain R.et al. *Machine Vision.* McGrawHill, 1995.

[11] Imagineer Systems Ltd. Computing a homography between 2d scene and image. Website. http://gandalf-library.sourceforge.net/tutorial/report/node109.html Last accessed June 16th, 2006.

[12] Edward Tse. Interactions inventor. Website. http://edwardtse.com Last accessed June 10th, 2006.

[13] Zhengyou Zhang. A flexible new technique for camera calibration. Website. http://research.microsoft.com/%7Ezhang/calib/ Last accessed May 12th, 2006.

List of Figures

List of Tables