University of Edinburgh
School of Informatics

Interactive Architecture

4th Year Project Report
Artificial Intelligence and Software Engineering

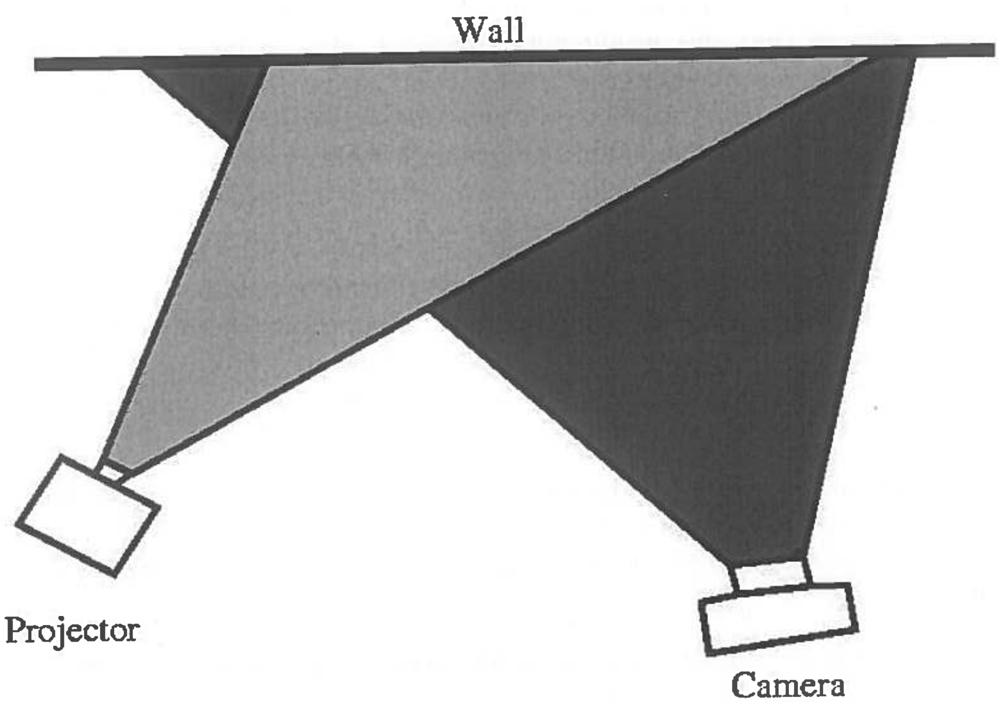
Michael Walters

March 2, 2005

Abstract: This document details the work involved in the 2004 4th year Undergraduate Project titled 'Interactive Architecture'. The work is being carried out by Michael Walters and supervised by Prof. Robert Fisher and is scheduled to last approximately 25 weeks. The project itself implements common image recognition and tracking techniques to allow a camera and projector to be placed at arbitrary positions for use in a visual menu interface. The interface is to be used as part of a visual direction system within a known environment.

'Interactive Architecture' was proposed by Prof. Robert Fisher and much of the guidance has come from him. His experience in vision systems has proved greatly beneficial in the development of this project. Thanks also go to the Informatics department of Edinburgh University, for use of their equipment and in particular to IPAB (Institute of Perception, Action and Behaviour) for the use of their cameras and projector. Further guidance has also been given by Barbara Webb, Tommy French, Ross Benzie, Ebenezer Ademisoye and Nick Johnson.

Interactive Architecture?



1.1 Initial System Proposal

Perhaps the easiest way to explain the initial project proposal is with a walk-through. Under a working version of the proposed system, the user would enter a room or hallway and be detected. This detection process could occur in one of several ways (Camera, push switch, light beam, etc.) and simply activates the system. Once active, the system projects a menu onto a wall nearby the user. This menu provides a list of names which the user can select by pointing to their desired choice. The system then projects directions along the floor instructing the user when they should turn (or continue straight ahead).

Reviewing the initial project proposal in conjunction with an investigation of the hardware available highlighted changes that would be necessary to the project.

The main result of this review was that the emphasis of the project changed. Initially the focus was on a fast, easy to use interactive system. Examining the hardware showed that this would not be possible due to large delays on the image capture (as described later). Further it was agreed that user detection and tracking should only be implemented if there was sufficient time, as adding these components would require an additional camera, which was not available. Also their inclusion at a later date would be easy (due to their modular nature) and the increase in terms of functionality was not enough to justify their implementation.

Once it was agreed that the user detection and tracking would be superfluous to the system, the need for the projection of directional arrows was also considered. As the inclusion of this feature would require a second projector (or a mirror-tilt projector both of which were unavailable) it was decided that this component should also be omitted.

Much discussion focused on the initialisation and set-up of the projected image. Aligning the image to the wall so that it appeared correctly whilst being projected from an arbitrary angle could be done either by hand for each installation or set up automatically.

Due to the hardware available the projected menu system should only project 6 options in a 2x3 grid. This could be easily updated for a real-world installation by increasing the number of options or making the menu system hierarchical.

1.3 Revised System proposal

After the above review the focus of the system shifted from fast interaction to automatic initialisation. There are two main driving forces behind this decision. The first is that whilst there has been a fair deal of work done on interactive gesture based systems, there has been relatively little done on automatically aligned camera-projection systems. The second is that, as mentioned above, the hardware in use would not allow for a fast enough response time for novel users.

The new system centers around the visual menu interface. The translations required to establish a correct projection of the menu are calculated automatically with prompts given where necessary. The user is then presented with a menu from which they can select one of 6 options and their choice is displayed. It is easy to see how such a system could be used in other systems and that it would offer set up each time.

1.4 Potential Issues

It is important to note here that the prototype system within this project and any subsequent installation would be substantially different. The purpose of this project is to demonstrate the functionality behind an eventual installation, and thus the prototype is somewhat lacking in places. Often this is due to the hardware available being unsuitable (a real installation would have purpose built, or at least adapted, hardware).

Because the system uses a projected menu background subtraction is a concern. When the user points to an object the projection will fall on their arm, this may make background subtraction and thus isolation difficult. If this is the case it would be wise to attempt the analysis on a simple menu first.

1.5 Restrictions

In order that this implementation demonstrates the functionality involved whilst not out-growing the scope of the project, several restrictions have been implaced.

- There will only be one user.

  By using a projected menu the system relies heavily on line of sight. Anything that gets between the projector and the wall will disrupt the menu. Further, the interference will be picked up on the camera and disrupt the system's ability to detect the intended target. To minimise this the system deals only with a single user environment. In a full installation this may seem unreasonable, however with the correct hardware and software the system would likely operate fast enough that the likelihood of any interference would be minimised.

- There will be a restricted space.

  Although the system has been designed to be both versatile and adaptive this has been done within reason. The system does not cope well with dynamic backgrounds, or spaces with heavily varying light conditions.

- The camera and projector are reasonably positioned.

  Although the camera and projector are automatically aligned and the system has been designed to accommodate arbitrary positions, this has to be

1.6 Hardware and Software

For the purposes of image capture the system uses a Creative webcam. These cameras can take around 2 frames per second, although access is restricted by the streamer command. The cameras are connected to the computer through a Usb cable, and images were taken at 480x480, although larger sizes are available. There are some temporary controls in regards to brightness and colour levels, however no way of setting them outwith xawtv.

The projector that the system uses is the Sony CX-PSD5. This mini projector plugs into the graphics output, and displays images at 900x700 pixels. This is irregardless of the actual size of the image as it would appear onscreen.

All applications and hardware are run from an Informatics' Dell Workstation running a KDE Linux distribution.

The system runs via Matlab version 6.5.1 installed on the Linux machines provided by Edinburgh University. Matlab is a commonly used mathematical analysis tool, and the majority of the image analysis was done via Matlab's Image Processing Toolbox. This provides several common functions that were used within this project. See appendix 1 for more on the commands used.

The streamer command line program is part of the xawtv package created by Gerd Knorr. Streamer allows command line control over webcams and is used here to take a series of images to monitor the position of the user's hand, and is also used within the calibration. Within this system the '-s' and '-o' options are used to specify the size and output respectively.

1.6.1 Hardware Issues

1.6.1.1 Camera

The Creative webcam has been a constant hindrance to the system due to its temperamental nature. Often the brightness and saturation of the camera will change without any apparent reason. This threw out many of the processes and accounts for the majority of the anomalies within the system. Consequently the system has been built to withstand this and is both reliable and adaptable.

Also, the camera suffers from radial distortion causing the edges to curve. As this was not noted until fairly late in the development of the system, it has not distortion [1].

Unfortunately due to hardware upgrades within the University, the Creative webcams were, for a while, misplaced, and it looked as though the project may have to continue using the new Logitech cameras. Whilst the Logitech cameras are far superior, with much higher resolutions, better auto-focus, constant light levels and a better lens, the University does not have the correct driver installed for their use. Thus the maximum size image available was 320x240, which greatly affected the menu. The system was altered to allow for this, however midway through alterations a Creative webcam was found, and re-implemented within the system.

1.6.1.2 Projector

Here the differences between this prototype system and a full implementation are more keenly felt. The computers used allow for only one display interface at a time. This means that when using the system either the monitor or the projector can be used but not both. This caused irritation in the testing and development stages, however, does not hugely effect the actual use of the system. Ideally there would be two displays; a monitor and the projector. The monitor would be used by the technician to set up and observe the system, whilst the projector would display the user's menu within the interactivity stage.

The projector suffers further due to the KDE Linux operating system. The projector provides little control over the size of the image it is displaying, and will only display an 900x700 portion of the screen. This is irregardless of how the screen is displayed on a monitor. Things are further complicated due to the fact that under KDE it is considerably difficult to alter the screen size. The two settings that the projector will display only show a small area of the screen, which has had to be accounted for.

1.6.2 Software Issues

1.6.2.1 Matlab

Unfortunately the Matlab 'Image Acquisition Toolbox', which allows Matlab to capture images from webcams, was not available. This would have allowed the system to run much faster during the pointing stage. As a result the system had to use the Unix 'streamer' command.

Within the system the 'streamer' command line is used to capture images. Streamer offers two options, the first is to specify a timeline over which multiple images are to be taken. Under Matlab these are all taken before proceeding to the next Matlab instruction. This means that all of the images except the final one would be redundant; the user would have moved on, before the system could process their actions. Alternatively, streamer allows one image to be taken, but then the connection to the webcam is closed. Opening this connection incurs a minimum delay of 1 second each time, and thus the user would have to point for a longer time at their target before receiving feedback.

The solution is sadly to 'hope for the best'. Tests show that users take approximately 2 seconds to point to a menu item naturally, and a survey showed that users would be willing to pause for a further second. The runtime of the pointing-analysis is around 2 seconds. This means that it is likely the user will be on the target as the photo is taken. By using feedback the user will wait longer, as they expect the result of their actions to be displayed.

1.7 Previous Work

The majority of the work done has been on two separate components, 'Gesture based Interactive Menu Systems' and 'Image and Projection Alignment', with a strong bias towards the former.

Initially the idea of using a more complicated interface system was explored and thus considerable reading was done in regards to gesture based interfaces. The attractions inherent in such systems are easily understood: a natural, intuitive control method and the only extra hardware needed is a camera. Initial attempts at implementing gesture recognition revolved around glove-based systems, but as these systems already require a camera it makes more sense to explore the vision based systems. Thus work was done based on the static pose of the hand, however with advances in the field, and greater accessibility due to falling hardware costs, work progressed onto interpreting dynamic hand gestures. Now there are numerous successful gesture recognition systems. [2] [3].

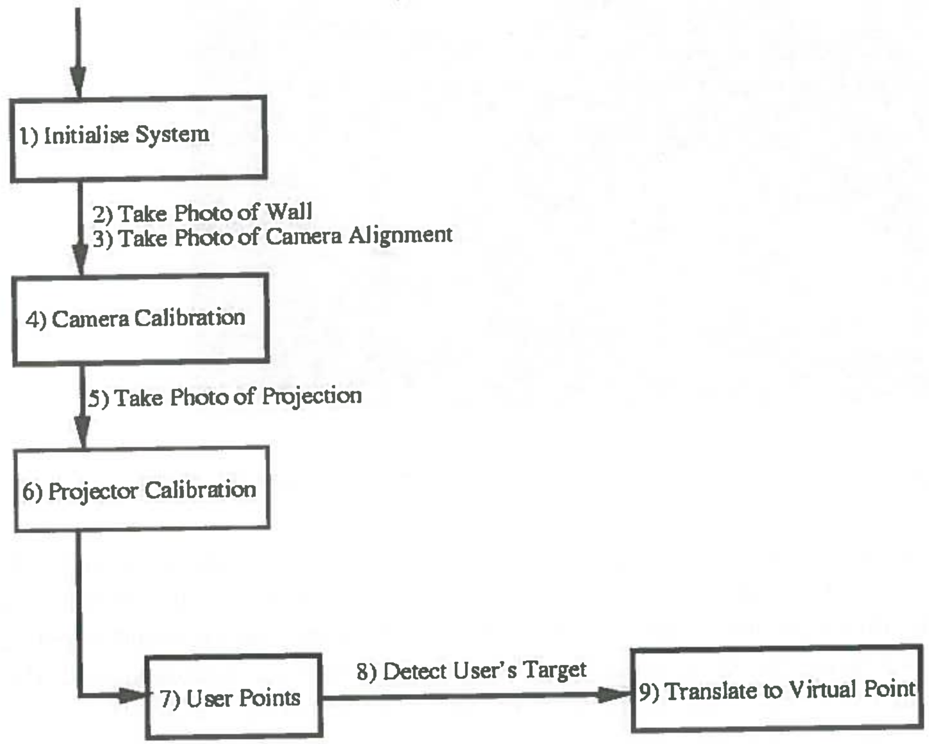
Whilst this success is admirable and could well have been integrated within this project, the scope of the project did not require it, and thus reading was refocused towards image analysis and tracking. This reading would help when developing methods to analyse and track of the user's hand. Paul Rosin's papers on 'Thresholding for Change Detection' [4] were an invaluable resource here, and gave an insight into techniques used to identify adequate thresholds. Further work on

The work done by Ahmed Elgammal et al. [5] showed that even in highly complex scenes it was possible to do very successful background subtraction. This reinforced the idea that background subtraction would be the most natural method to use in the isolation of the user's hand. In their paper Elgammal et al. present a technique for creating a model of the background based on *"kernel density estimation of the probability density function of the intensity of each pixel."* Although this system proved very successful it is slightly excessive in a static environment such as ours. With more time available this method would however, allow for the system to work in a dynamic environment (assuming the movement was not between the camera and the user).

Both *"Multiple View Geometry in Computer Vision"* [6] and *"Computer Vision: A Modern Approach"* [7] provide information on projective geometry and the necessary transformations required to project from one 2-D plane to another.

In "Auto-Calibration of Multi-Projector Display Walls," by Raij et al. [8] several projectors are aligned to create (the illusion of) one image. This and work done by Rahul Sukthankar et al in *"Smarter Presentations:.."* [9] formed the basis of the image alignment for this system.

System

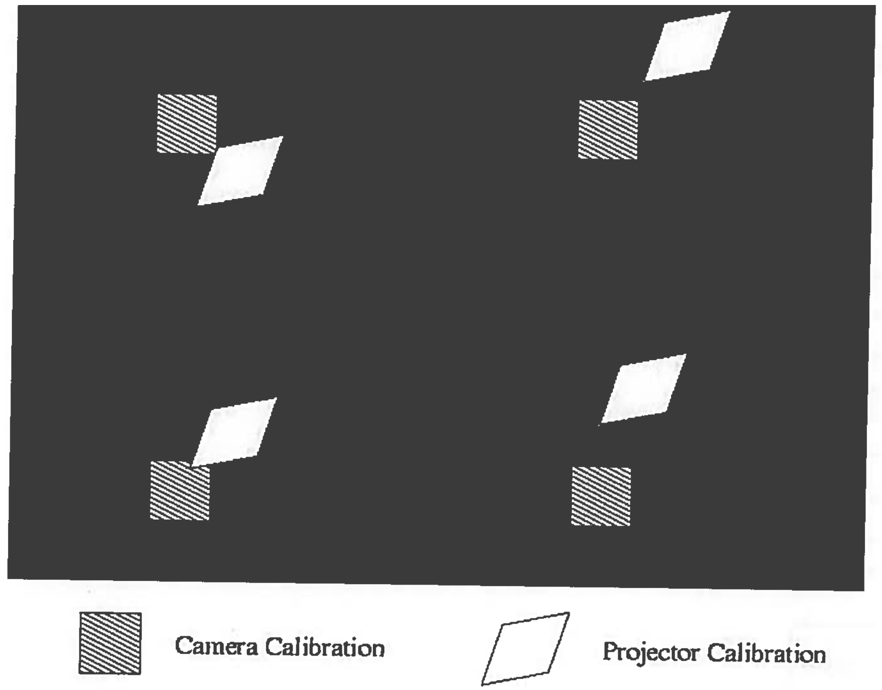


2.1 Calibration

The calibration procedure will be carried out by the technician that installs the hardware. It is assumed this person has little or no experience with image analysis, and thus this stage is as automated as possible. Prompts, where necessary, are direct and easily understood. Calibration need only be carried out once, when the system is installed.

When installing the hardware, it is essential that the camera has a full view of the scene, including the entire area into which the menu is to be projected. Also the projected image should not be greater than 120cms to maintain correct alignment.

The first step is to boot up the pc and load the required files. Obviously in a real world installation this would be loaded on chip, and would launch automatically.



The technician then follows the prompts. The first task is to take a photograph of the background (called 'bg'). Then, following the prompt, the orientation markers should be placed at the corner of the desired menu location and another photograph taken (called 'camera'). These should then be removed and the projector turned on and used as the display. With the projector calibration image shown, the next photograph should be taken (called 'projector'). After taking the camera and projector photographs, the system will prompt the technician to check that it has correctly isolated the calibration points. If the system can not identify them, the user should manually select them by clicking on-screen.

With the points selected the system will display a test image via the projector which should be aligned to the wall. A final confirmation is required, before the system proceeds to the Pointing stage.

2.2 Pointing

This is the main stage of the system and will be used as soon as calibration has been completed. Here the user will point at the menu to indicate their desired target. The projection will provide feedback and will display the calculated target, once the user has hovered over it for long enough (about 2 seconds). If the system is incorrect, feedback should be provided to allow the user to compensate for 20 iterations (about 1 minute).

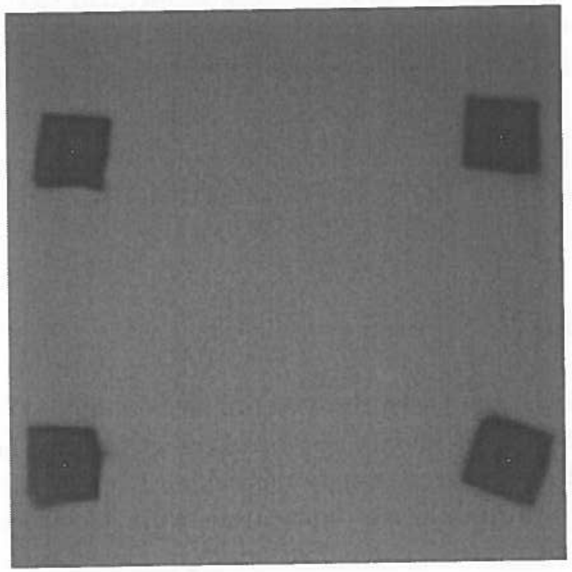
3.1 Calibration

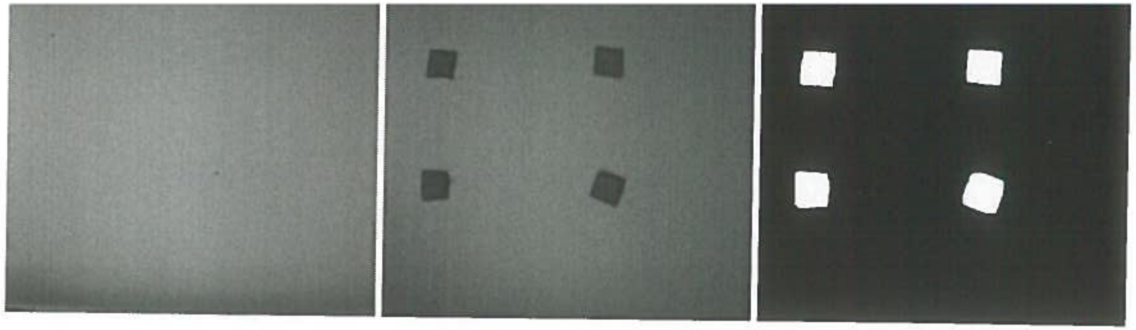
3.1.1 Description

Calibration, as described above, covers the task of setting up the system so that the camera and projector angles are accurately compensated for. In the design of this part it was important that all stages be as automated as possible, and that when user intervention is required adequate instructions are given. This motivated a prompt based system, which leads the user through the set-up step by step. To combat the temperamental nature of the webcams, users are asked for confirmation after each image and after major calculations.

The calibration works by identifying four points in the captured image, which are then mapped to four known points. Here, four cards are arranged in to a square. These four cards are isolated in the scene and the median of the points of each card found. This gives four points; the midpoints of each piece of card. These points are then passed to the 'transformgenerator' method which uses them to describe a mapping from one set to the other. Below the two main components, Isolation and Transformation, are explained in greater depth.

3.1.2 Isolation





3.1.2.1 Approach

To accurately calculate the transformations, the transformgenerator method requires the position of four points on the captured image which correspond to four previously identified points. These are gained through the same method for both the projector and the camera transformations.

The initialisation procedure requires the user to take three photographs. The first is of the wall or other surface onto which the menu is to be projected. The second is of the same wall with an orientation card or four markers, which will define the area of the menu. The final image is of the wall (without orientation card) with the projector orientation image projected into its correct place.

In both camera and projector isolation stages the same technique is used. First the background is removed via the Matlab command 'imsubtract'. This subtracts every element in the background image array from the corresponding element in the camera or projector image array and returns an array of the difference. Whilst in theory this would completely isolate the squares, in reality fluctuating light levels and other noise mean that thresholding is required to properly isolate the shapes.

Once these shapes have been isolated each square is classified and subsequently averaged to locate the centre of mass which is then passed to transformgenerator to create the relevant transform.

3.1.2.2 Problems Encountered

Thresholding here was done with the Matlab method 'im2bw' which takes a value between 0 and 1 as the threshold. The precise level of thresholding was difficult to decide as adaptability was strongly desirable. Because of this, the threshold value is calculated within the system, rather than hardcoded. If the squares are not correctly identified on the first attempt, the system prompts the user

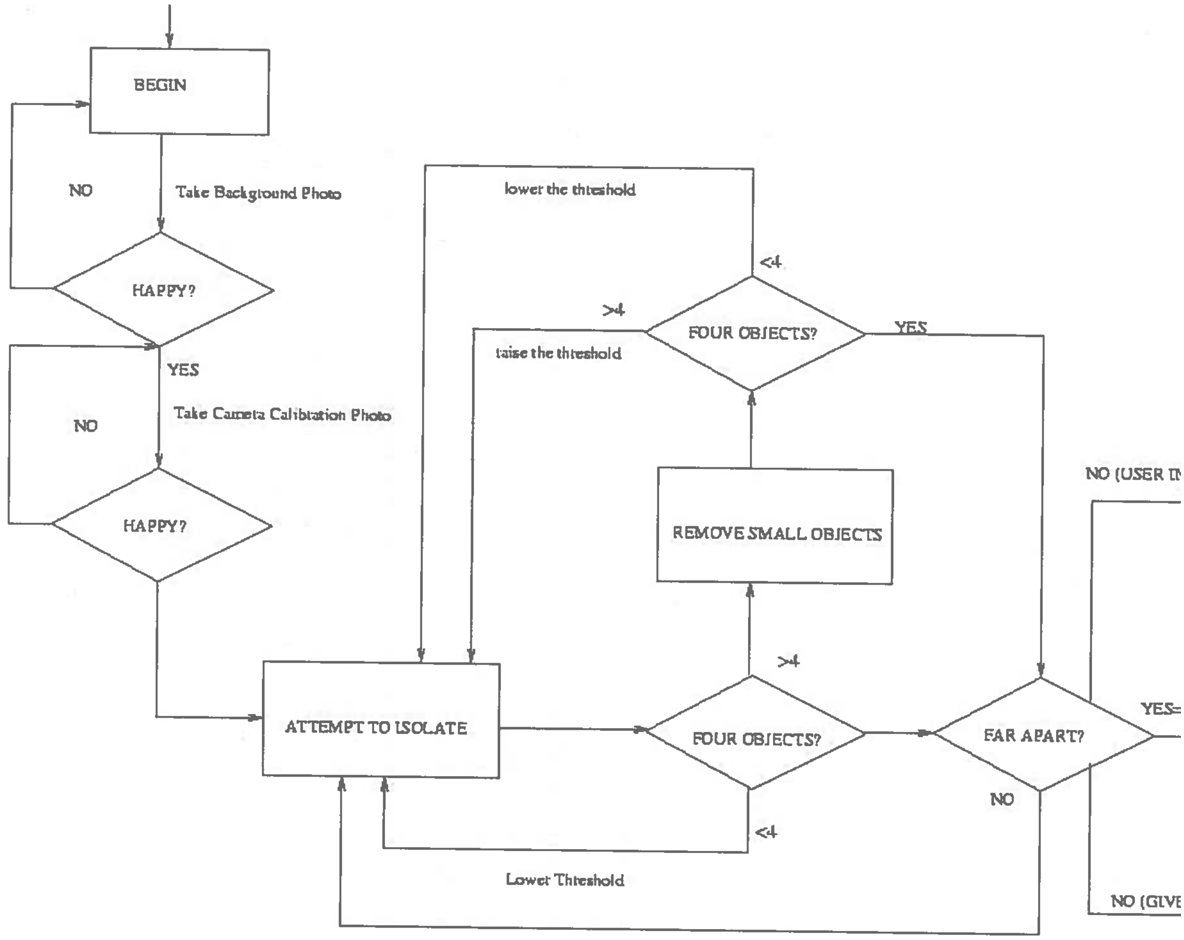
the calibration points will not be detected by the system, [illegible]
rigorous calculations, by selecting the points via their mouse. Otherwise the system will attempt to find the correct calibration points by altering the threshold in accordance with the current number of detected objects. Thus if there are too many objects the threshold is increased, too few, and it is decreased. The amount of alteration is constantly decreasing so as to allow a more accurate figure to be reached, whilst not impeding the result. If, after 20 iterations the points have still not been adequately identified then the user is prompted for further guidance.

After the image is thresholded the 'bwlabel' method is used to identify the number of shapes. This was often far greater than 4 in tests, as the thresholding had caused gaps in the calibration shapes. To combat this the image was dilated (to 'fill in' any potential gaps) with a 10x10 pixel square. This combination of thresholding and dilation resulted in very accurate results.

Whilst the above changes made drastic improvements to the isolation ability, tests still illustrated some errors. If the background or viewpoint had changed slightly, the 'imsubtract' command returned background elements as objects. To get rid of these (and any noise), a method was introduced to remove shapes smaller than the assumed calibration shape. A list of connected shapes was found from the thresholded image, and those below a set size were removed from the image. This minor alteration greatly improved results.

Unfortunately the points were still sometimes incorrectly chosen, and a sanity check was put in place to ensure that the results were more realistic. This works by checking that the calibration points are at least 50 pixels away from one another, but it can be overridden by the user.

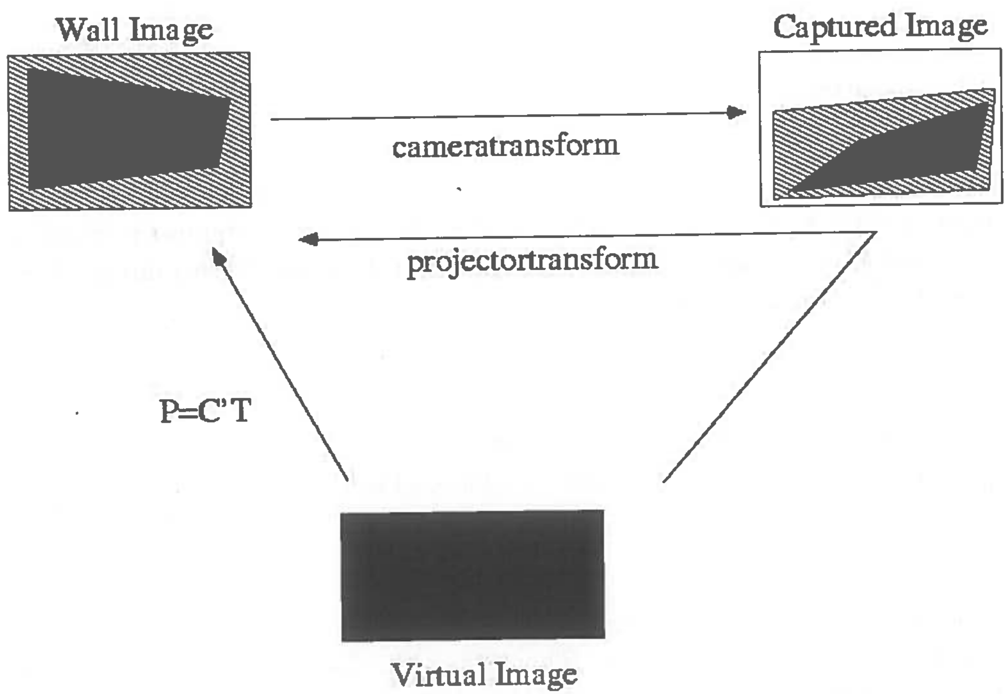
All of these changes created a somewhat complex, although very effective, isolation approach. The final method preformed as below.


BEGIN
Take Background Photo
NO
HAPPY?
YES
Take Camera Calibration Photo
NO
HAPPY?
ATTEMPT TO ISOLATE
lower the threshold
<4
>4
FOUR OBJECTS?
YES
raise the threshold
REMOVE SMALL OBJECTS
>4
FOUR OBJECTS?
<4
Lower Threshold
FAR APART?
NO
NO (USER I
YES=
NO (GIVE

By designing the system to work in flexible surroundings it has also been made very resilient and reliable. In almost all tests the calibration points were accurately detected. However when the points are not accurately detected (or not detected at all) the fallback system which allows the user to specify the points themselves means that there should never be a failure.

The isolation function runs fairly fast and there is a good balance between user control and system automation. A possible improvement would be to set the minimum size of detected objects dynamically through a comparison of the current objects.

3.1.3 Transformation



By calculating the appropriate transformations the system should be able to compensate correctly for the distortion of the image, both captured and projected.

Such a transformation T can be expressed: $$\begin{pmatrix} x\omega \\ y\omega \\ \omega \end{pmatrix} = \begin{pmatrix} t1 & t2 & t3 \\ t4 & t5 & t6 \\ t7 & t8 & t9 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Within this system there are three transformations that are required to properly correct the projected image and interpret the captured image;

$$\begin{pmatrix} X_2 & Y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -Y_2x_2 & -x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -X_2y_2 & -Y_2y_2 & -y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -X_3x_3 & -Y_3x_3 & -x_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -X_3y_3 & -Y_3y_3 & -y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -X_4x_4 & -Y_4x_4 & -x_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -X_4y_4 & -Y_4y_4 & -y_4 \end{pmatrix}$$

- Wall to Camera.
- Projector to Wall to Camera.
- Projector to Wall.

3.1.3.1 Approach

The first two transformations can be calculated via the points gained in the isolation step described previously. Given four identified points, hereafter referred to as (x, y), and four corresponding known points on the virtual (ideal) image,(X,Y) let A be the following matrix:

T then is simply the null value of A which can be found easily in Matlab via the 'null' command. With more than 4 identifiable points this matrix can be extended to provide greater accuracy, however it was thought that four points would be sufficient.

Using the four identified wall calibration points the above gives the Wall to Camera transform ('cameratransform'). Then using the four projection calibration points the Projector to Wall to Camera transform ('projectortransform') can be found. From these two transformations the more practical transformations can be derived. The mapping from Projector to Wall is given by

$$cameratransform^{-1} \text{ x } projectortransform$$

This transformation allows us to find the necessary 'prewarp' required to make the image appear correctly after projection. This prewarp is simply the inverse of Projector to Wall mapping.

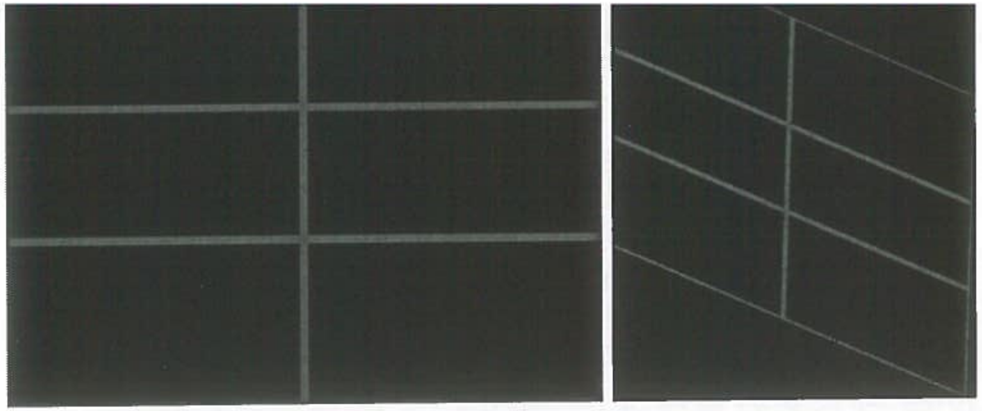


Figure 3.1: This illustrates the original (right) and corrected (left) menu boundaries.

3.1.3.2 Problems Encountered

Although straightforward in theory the implementation of this section was troublesome. The difficulty is that the only real opportunity to judge the accuracy of the prewarp is at the final stage, which led to time consuming tests. Also, due to the interaction of the matrices, tracing any faults was difficult, and often required exhaustive searching.

The accuracy was further hampered by the sensitivity of the prewarp matrix. When dealing with heavy distortion, small changes in the isolated points causes a large impact on the final transformation.

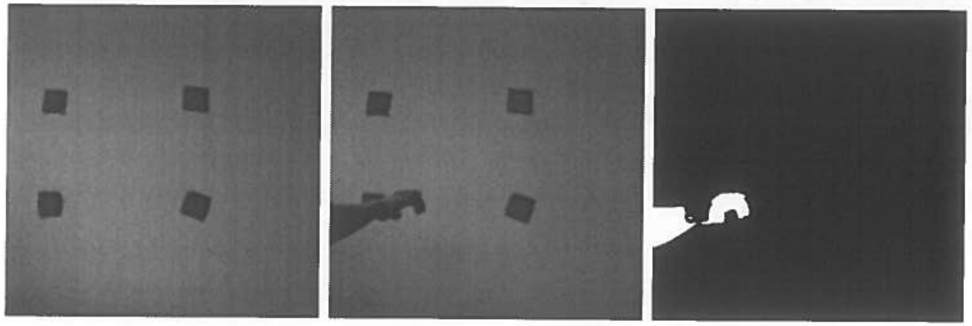
Possibly the largest problem however was finding the inverse mapping. Often finding the mapping from the real world image to the virtual, or vice versa, was relatively simple, however finding the inverse was not so easy. This may well have been due to inexperience with Matlab.

3.1.3.3 Review of Implementation

Although the transformations established here are not new work, they have caused a considerable amount of frustration, due to their sensitivity. Identifying the four corner points on the wall (projected or otherwise) typically creates a margin of error up to three pixels which is then propagated through to the matrices.

point. This would allow a more accurate calculation of the first two matrices and thus the accuracy of the prewarp matrix would be far improved.

3.2 Pointing



3.2.1 Description

This section describes how the user's interactions are handled by the system. The 'Pointing' section illustrates the success (or failure) of the calibration, as it is only here that the results will be evident. Within the Pointing stage there are two main sections; Isolation and Projection.

3.2.2 Isolation

3.2.2.1 Approach

Although similar to the above method of isolation, here things are somewhat different. Whereas before, the system was dealing with a static image, here frames from a dynamic scene are used. This means that the isolation must be fast and subsequently adaptive thresholding is less suited. Fortunately, here accuracy is not as large an issue, and thus the threshold can be quite high. Further, the matte properties of skin allow for higher contrast.

The translation of the target in the captured image to the virtual target should be simple to achieve via the previously calculated matrices.

As mentioned above, the streamer command line causes a large delay in capturing images. Considerable time was spent attempting to fix this, however no better solution could be achieved.

The pointing was at first somewhat unreliable, however this was due to a small bug in the coding. When that was fixed the only real issue was ensuring that the threshold was set to a reasonable level. Often better results were gained by wearing a glove. However, once the threshold was set appropriately the system preformed very well, identifying the user's intentions from a variety of angles.

3.2.2.3 Review of Implementation

The thresholding applied to the captured image is of concern as it is hardcoded. Given more time a dynamic thresholding technique would be implemented, which would allow for a greater variety of skintypes and pointing techniques. That is not to say that the current system is inflexible, just that there is scope for greater flexibility.

Also in tests it was highlighted that the restriction of enforcing the user's position in respect to the screen (left of the screen), should be changed. Allowing the user to stand on either side would be a fairly simple update.

3.2.3 Projection

3.2.3.1 Approach

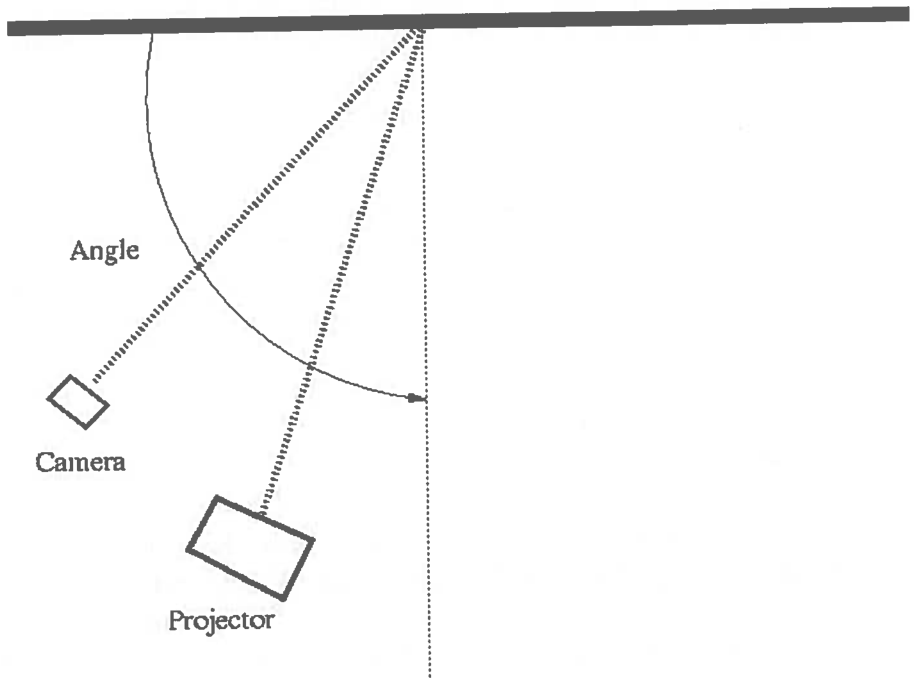
Altering the projection via the previously calculated transformations should be straight forward. The prewarp has been calculated and by applying the transformation to each pixel a new image is calculated that should appear orthogonal to the ceiling and floor when projected.

3.2.3.2 Review of Implementation

Unfortunately a large section of implementation was neglected. The rescaling and repositioning of the menu within the projection area was not fully accounted for. Although this translation does take place to a degree, a fully defined transformation of Virtual Image to Corrected Image would have been preferable and allowed for greater accuracy.

the space, and the angular distortion was completely accounted for. Strangely the base of the prewarped image was occasionally over-warped. This is most likely due to the radial distortion of the webcam.

4.1 System Results



Appendix 2 shows the results of a set of experiments in which the camera and projector were positioned at a series of different angles to the wall. First the camera and projector were positioned, and then several measurements were made. The angle of distortion along the top of the image was measured before and after correction. The isolation method was measured by determining whether the calibration points were properly isolated, for both the wall and the projector based points. Finally the user was asked to point to each of the target regions 5 times, in a random order, and the calculated target region was noted.

For wall calibration points pink square pieces of paper were used and placed in an adequately sized square on the wall. The camera and projector were kept at least 200 cms from the center of the projection at all times to enable a usable projected menu.

4.2.1 Isolation

The isolation of the wall calibration squares has been done well, failing only once when the camera was mounted at 40 degrees. This may well be because of a slight slip in the camera's position during the trials, as the webcam used had no stand. Any significant changes in position or orientation of the camera would alter the image sufficiently that the image subtraction would recognize the parts of the background as new objects. The simplest way to solve this problem, besides an adequate camera mount, would be to threshold and then dilate the original images (background and projector/camera) before performing the image subtraction.

Also, a more precise method of calibration in regards to the projector alignment would be a large improvement.The current system rarely recognises the projected light patches. This is due to the poor cameras, however the images are still of a high enough quality that a more developed system would be able to isolate the patches.

4.2.2 Calibration

Projector calibration is definitely the system's strong point. From these results you can see that the system preforms well, aligning the display grid almost perfectly in all trials. Although not shown formally, trials were done where the camera and projector were positioned at mirrored angles (the camera at 60, the projector at 150) and the alignment worked well here also.

Although the top angle was well calibrated, the warp applied to the shape also caused deviations along the bottom edge. This gave mixed results, with some projections being wildly distorted along the bottom edge. After investigation and further testing it was shown that this was due to errors with the camera lens which suffers from heavy radial distortion. The best method for bypassing this is to ensure that the camera is vertically aligned with the center of the projection.

4.2.3 Pointing Analysis

The system preformed very well identifying the user's input in most cases. Although a few cases gave unexpected results, these may be down to the sensitivity of the transform matrices. However, judging from the thresholded images it is more likely that they are due to the value of the threshold in the locator method.

make a large difference. Although changing it is not difficult ... correct value can be tedious. Adding a calibration step to this would not be overly difficult, and the benefits would be worthwhile.

Another reason for the few failures, is the projection. As mentioned at the beginning (1.4 Potential Issues) using a projected menu causes difficulties. Even with this basic border menu system when beams overlay the user's arm that area is seen as part of the background and removed. The only potential option then, would be to use a light based pointer of some sort. This would enable the user to affect the menu easily and intuitively. However, the locator method would have to be considerably more accurate for this to be viable.

4.2.4 Miscellaneous

It is important to note here that due to the space available, tests were done at a fairly close distance to the wall. Depending on where the system was installed this close proximity may or may not be a factor and the system would likely benefit by furthering this distance.

Although noted earlier it is worth stating once again that the hardware and software here has been a major drawback. By using a better quality camera the calibration points could be more easily identified, allowing for more accurate transformations. A faster method for capturing images would have greatly accelerated the process of locating the user's target, and would be absolutely necessary for any real-world installation.

4.3 Conclusion

In order to decide whether or not the system has been a success the individual components will be reviewed, before concluding on the system as a whole.

4.3.1 Calibration

This component would appear to be a success. The wall calibration points were automatically identified almost every time, even with complex backgrounds and noise. Within the isolation method, small noise is removed and the correct threshold is calculated by the system. Although the automated isolation of projected

[1] for an experienced programmer. It would not be expected of a technician

... the opportunity to correct the (wrongly) identified points in all cases. If the success of the wall calibration could be mirrored in regards to the projector calibration then this component would be greatly improved.

The importance of a well calibrated menu can not be over-stressed. The entire system is dependant on proper calibration, in order to facilitate the pointing interface. The combination of accurate (and partially automated) isolation and strong corrective prewarp ensure more accurate results.

4.3.2 Pointing

The pointing analysis works incredibly well from a large angle of incidence. Even when the angle is more than 45 degrees from a perpendicular position to the projected image, it can accurately detect the user's intentions. This accuracy is somewhat limited by the distance from the wall, in that the camera's poor resolution causes pixel confusion at long distances. This could be easily rectified through the use of a more modern camera.

As stated earlier the main problem with the pointing method is that the projected menu interferes with the user as they point at their target. Whilst the use of a laser pointer would help to eliminate the problem, such tools detract from the purpose of this project. Ideally the system should be open to all users and require nothing from the user (except themselves). Introducing laser pointers and other such interface devices introduces the possibility of theft and loss, and thus would not be suitable here.

Instead a more precise method of detecting the changes such as those used by Elgammal et al. [5] would perhaps pick up the distortion. If not, the shadow cast by the users arm would provide another potential source for the locator method.

4.3.3 System

Although both Calibration and Pointing are an integral part of the project, they can be used separately. By mounting a projector with a simple webcam and using the techniques described above, the projector could provide automatic keystone correction. The stationary position of the webcam and projector, as well as the increased distance, would provide far better results, eliminating most errors.

The Pointing component could be used in a variety of environments, and would actually work better without the projected menu. One possible use would allow

---

[2] The walls used in the test scenario were glossy and reflected the projected menu.

weatherman.

However, the purpose of this project was not only to create Calibration and Pointing components but to ensure they worked together. Here this system is a great success. The Calibration section will accurately and automatically align a projector and camera (in arbitrary positions) to a screen, with minimal help from a user. Then the user can interact with the menu in an easy and intuitive manner.

The system is not without flaws, most notably the delay caused by streamer in capturing the images. For many this would cause them to be hesitant to use the system, however any real implementation would not have this problem. A real implementation of this system would have serious improvements over this prototype. In fact the only remaining problem would be the projected menu interfering with the user. It is because of this that the system can be claimed successful. Given the hardware available it preforms very well, and the software used could easily be adapted to use better hardware and thus produce far better results.

[1] G.P. Stein. Lens distortion calibration using point correspondences.

[2] Rung-Huei Liang and Ming Ouhyoung. A real-time continuous gesture recognition system for sign language.

[3] J. Davis and M. Shah. Visual gesture recognition.

[4] Paul L. Rosin. Thresholding for change detection.

[5] David Harwood Ahmed Elgammal and Larry Davis. Non-parametric model for background subtraction.

[6] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision.

[7] Jean Ponce David A. Forsyth. Computer vision: A modern approach.

[8] Andrew Raij and Marc Pollefeys. Auto-calibration of multi-projector display-walls.

[9] Robert G. stockton Rahul Sukthankar and Matthew D. Mullin. Smarter presentations: Exploiting homography in camera-projector systems.

4.4.1 Matlab Image Processing Toolbox Commands

'im2bw': This function takes a colour image and threshold as variables, and converts the image to binary based on the threshold value.

'bwlabel': This function takes a binary image and returns a matrix of the same size as the input image matrix, listing all the connected components as different objects. A second output variable gives the number of objects.

'imdilate': This function takes an image matrix and dilates the image with a specified shape.

4.4.2 Implemented Methods

'doall': This is the main function, from which all of the others are called. This is what the technician would need to start. Doall takes no variables, and returns no output, simply looping indefinitely.

'init': This function takes two images and identifies the changes, before attempting to locate the four points required to calculate the transformation matrix. This is one of the more advanced functions.

'locator': This function takes two images, and returns the coordinates of the leftmost part of any new objects. Locator is used to identify where the user is pointing, however these coordinates relate to the captured image and must be translated to find the region.

'myinput': This function takes a variety of inputs and translates them from a captured image to the ideal one.

'myoutput': This function translates menu.jpeg into the prewarped image required to properly align with the walls.

'region': This function simply returns the relevant region of the coordinates passed to it from the output of myinput.

'takephoto': This function interacts with the user to confirm that photos taken are correct. This was done to minimise problems caused by the webcams.

'transformgenerator': This function creates the 3x3 transformation matrices required from the points found via init.

Projector Position			Camera Position			Distortion	Wall Isolation		Camera Isolation	Pointing					
Angle	X	Y	Angle	X	Y	Angle	Angle	Success?	Success?	1	2	3	4	5	6
90	0	270	90	0	270	0	3.11	Yes	No	5	5	5	5	5	5
			65	111	238	0	0	Yes	No	5	4	6	5	5	5
			40	167	140	0	1	Yes	No	5	4	5	6	5	5
65	111	238	90	0	270	6	2	Yes	No	5	5	4	5	5	6
			65	111	238	6	0	Yes	No	5	5	5	5	5	5
			40	167	140	6		Yes	No	5	4	7	5	5	4
40	167	140	90	0	270	9	0	Yes	No	5	5	5	5	5	5
			65	111	238	9	0	Yes	No	5	5	5	5	5	5
			40	167	140	9	0	No	No	6	5	6	4	5	4