School of Computing

BSc (Hons) Computing and Informatics

PRCI303/4 Final Stage Project

Scott Blunsden

Chess Recognition

2002/03

Bsc (Hons)
Computing and Informatics

SCOTT BLUNSDEN

CHESS RECOGNITION

Final Year Project
2002/03

This report is submitted in partial fulfilment of the requirements for the Bachelor of Science (Hons) in Computing and Informatics, School of Computing, University of Plymouth

Acknowledgements

The project manager would like to take this opportunity to thank the project supervisor, Dr. Guido Bugmann for his expert guidance and support for the duration of the project. His help has been invaluable during the course of the project and his enthusiasm guidance and expert opinion enabled the successful completion of the project.

Contents

1. Abstract *6*

2. Related work *7*

2.1. Template based Approaches 7

2.2. Histogram Based Approaches 9

2.3. Snake Based Approaches 12

2.4. 3D Approaches 15

2.5. Biological based Approaches 17

2.6. Vision systems specific to chess playing 19

2.7. Evaluation of the systems for purpose of the project 20

3. Board Localization *23*

3.1. Candidate methods 23

3.1.1. Board detection 23

3.1.2. Snake based approaches 26

3.1.3. Feature Detection 36

3.1.4. Self organising map approach 49

3.1.5. Shape Model Details 57

3.1.6. Final Algorithm Details 61

4. Movement detection *66*

4.1. Board Orientation 66

4.2. Occupied square 71

4.2.1. Colour Difference 71

4.2.2. SUSAN Activation 71

4.2.3. Results 73

4.3. Movement 75

4.3.1. Results 76

5. Critical Evaluation *77*

5.1. Project Management 77

5.2. Investigation and Design 78

5.3. Development 79

5.3.1. Development environment 79

5.3.2. Image processing 80

5.4. Implementation of final system 81

5.4.1. Design 81

5.4.2. Functionality 81

5.4.3. Reliability 82

5.4.3.1. Recognition 82

5.4.3.2. Movement Detection 83

6. Future work *84*

6.1. Recognising individual chess pieces 84

6.2. Robot controller **84**
6.3. Board Tracking **84**
6.4. Other board games **84**
6.5. Extension of the rotation filter Method **85**
6.6. Evolution of the shape model **85**
7. Conclusion ***86***
8. References ***87***
Appendices ***92***
Appendix A - User Guide **92**
Appendix B - Implementation Details **101**
Architecture **102**
Software **104**
System **104**
Appendix C – Sample Game sequence **105**

1. Abstract

This report describes the investigation design and development of a system that allows a computer to convert an image or sequence of images into a form that will allow a computer to play chess. The investigation and design of the system is described and any decisions that affect the systems design are justified. The final system is then described in detail and the issues associated with the design are discussed. The results of each component of the system are presented and then discussed.

An evaluation of the system is then provided along with suggestions for future work. The implementation details of this system are provided in the appendices.

2. Related work

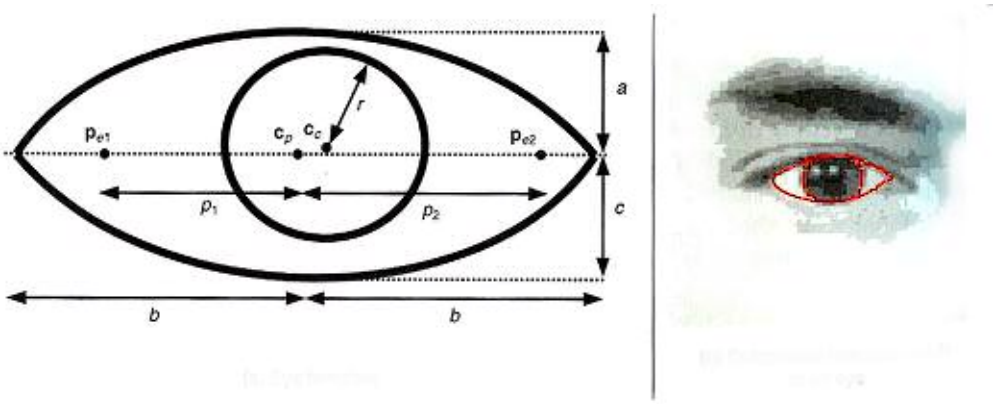
2.1. Template based Approaches

Template base approaches to visual recognition are probably the simplest approach, consisting of comparing one image to another. The approach has major disadvantages mainly relating to the number of templates required to cope with slight variations in the original objects orientation or shape changes, such examples may be the different handwriting styles that can be used to represent a character. Another limitation is the number of comparisons that are required to cover all possible scales and orientations that an object could take. For this reason template methods are only used in situations where deviations from a standard set of templates can be easily compensated for or are not required. An example where template matching was successfully implemented within a visual system is detailed in "Vision-Based Urban Navigation Procedures for Verbally Instructed Robots", [T. Kyriacou, G. Bugmann and S. Lauria, 2002] where by all features that are to be detected can be represented in a by an acceptable number of templates. The system uses templates of road junctions to recognize features within images acquired by a robot. As it is designed to work with only a limited set of situations the method provides a simple and quick solution to the feature detection problem.

In general when it is known in advance the features that are to be detected and how they will be presented template based matching provides a simple way to identify features and objects within the image. The simple template matching systems have been extended to help with the problems of scale and rotation. Such an example would be the use of generalized Hough transforms described in [M. Nixon, A. Aguado a, 2002] whereby a geometric object can be represented by a set of lines radiating from a central point within the object. Further extensions have led to the creation

of an invariant technique that are a more optimised version of the standard model detailed in [M. Nixon, A. Aguado b, 2002].

Other approaches have also followed the deformable approach where an object can be represented by a more flexible method that can define how elements of the object should be placed within the object. An example of this is a deformable model of an eye where the eye object could be represented by a set of parameters defining the radius of the lens of the eye and the width of the complete eye. A further rule would state that the lens has to be within the complete eye. This would allow individual elements of the eye to change size and position while still keeping the overall eye shape and hierarchical elements that make up the shape. One possible deformable template to represent the eye is presented below in Figure 1.



Deformable template match to eye

Figure 1 - Finding an eye using a deformable template, Adapted from [M. Nixon, A Aguado, 2002]

2.2. Histogram Based Approaches

A well used approach to object recognition is based upon the use of histograms to represent objects. Many of these systems have been based upon the work of Swain and Ballard [M. Swain, D. Ballard, 1991] who proposed a method to represent objects by its colour histogram. Objects were then subsequently identified by the use of histogram matching techniques where an image region was matched with the histogram of a previously learnt sample object. This method has proven to be robust to the object orientation, scale and also some robustness was observed to partial occlusion. The systems simplicity of using only colour histogramming as a means for object identification meant that certain objects could not be described using only colour. The reliance on colour also meant that the system was sensitive to changes in light intensity. The limitations concerning light intensity variations have been partially solved Healey and Slater [Healy and Slater, 1994] and Finlayson et al [Finlayson et al., 1998] have all proposed improvements to this basic colour histogramming method by trying to overcome the intensity problems in a variety of ways.

Recognising the limitations of using only colour as a means to describe an object several methods have used the histogramming approach but used other measures of an object to build the histograms. One such method is that employed by the SEEMORE system as described in [B. Mel, 1997], the system still makes use of the histogramming approach for describing an object but many more features are utilized. The SEEMORE system's world consists of encoding an object by using 102 different feature channels. These channels consist of five main groups:

- Circular hue/saturation channels,
- Course-scale intensity corner channels
- Circular and orientated intensity blobs
- Contour-shape features

- Orientated energy and relative-orientation features using Gabor functions.

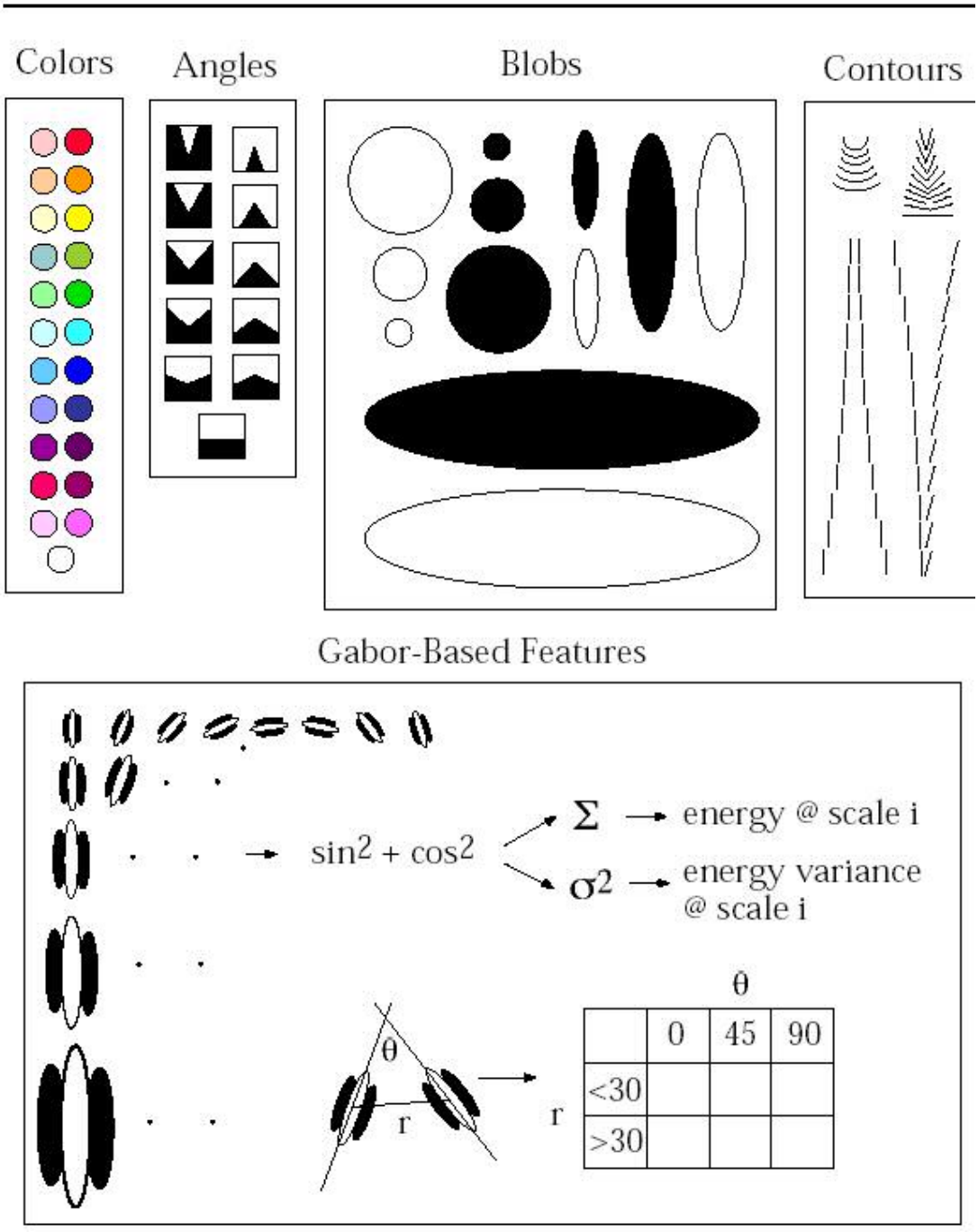


Figure 2 - SEEMORE's feature channels Adapted from [B. Mel, 1997]

For each feature channel the image produces a histogram based upon its response from the filter. Most filters required some pre-processing of the image; details of the pre-processing differ depending upon the filter being used. As an example before applying the colour feature channels images were re-scaled. Impressive results have been obtained by the SEEMORE system; even some of the mistakes made by the system seem understandable with many errors being the result of misclassification of similar looking objects.

The Histogram approach has also been used by Scheilde in “Object Representation using Multidimensional Receptive Field Histograms” [B. Scheilde, 1997] and again in
“Recognition without Correspondence using Multidimensional Receptive Field Histograms” [B. Scheilde, J. Crowley, 2000], where the histograms have been extended into two dimensions. Again filtering is employed to build the histograms using receptive fields over several different scales and rotations of the original image to describe the object. Histogram matching techniques are then used to classify the object. This method shows advantages over the SEEMORE system when attempting to classify partially occluded objects as it provides a means to match several classes of object to one image, so catering for the case where many objects are present in one image.

The use of historamming as a means to represent and recognise images has been well tried and tested by several groups and systems with some good results obtained. The techniques show good robustness to partial occlusion and changes in orientation, mainly due to the histogram removing this information from the image and representing it as a frequency measure for certain classes and its ability to work upon parts of an image.

2.3. Snake Based Approaches

The idea of a snake resulted from the work of Kass,Witkin, and Terzopoulos [Kass et al. 87a, Witkin et al. 1987, Terzopoulos et al 1987]. At its most basic implementation the snake is an energy minimization process where a set of points are continually changed based upon the points distance from each other, the curvature between two points and the energy of the underlying image at a certain point. In the case of a basic snake the image energy refers to contours between image sections, such as those that result from edge detection operations. Each of the components are weighted with the detection of a contour typically having the highest weight, constant movement is ensured by having the snake contract to a centre point of defined by all the current snake points. Snakes were and still are used to extract features within an image where they have been placed outside the feature to be extracted. This has severe limitations for object recognition tasks as initial placement of the snake is a manual operation and they have no direction as to what features they are looking for within an image.

These limitations have led to the development of active shape models where the flexible and dynamic image search features of the snake are combined with previously input rules that guide the snake towards the classification of a single object or feature within an image. In their paper entitled “Active Appearance Models”, C.J. Taylor, G.J. Edwards, T.F. Cootes present active appearance models and describe how they can be used to contain a statistical model of a shape and search an image for these features. The examples presented in the paper uses faces where the face of a person is extracted from an image. Only one AAM, trained upon a variety of face images, is used to successfully extract the face portion of each image. The models use the concept of landmark points where a shape is made up of the connection of points to each other and each point is allowed to change its position within a previously learnt area

to ensure that the model can change its shape to fit the image whilst still retaining the original object features.

This idea has been applied to other objects other than faces. In “A Brief Introduction to Statistical Shape Analysis” [D. D. Gomez, M. B. Stegmann, 2002] D. D. Gomez and M. B. Stegmann show the method successfully applied to hand recognition where a hand is automatically recognized by an AAM trained upon hands in various positions (illustrated below in figure 3). Further applications have seen AAM’s applied to detecting metacarpal bones in an X-Ray image and finding brain stems within a MRI image.

Figure 3 - Set of 52 Landmark points for describing a hand, (Adapted from [D. D. Gomez, M. B. Stegmann, 2002])

The advantage with these techniques are that they work well, and provide a way to model objects that is not as abstract as the histogram based approaches described in the previous sections. The models also prove to be robust to rotation and small variations within an image due to the AAM's ability to rotate and model variations within the image. The training of the AAM is more problematic as it uses landmark points that may have to be assigned by a human in the original images so that the AAM can be trained. When attempting to find out what object the computer is looking at the AAM method would require testing of many AAM's to determine which AAM's converge to identify objects present within an image. AAM's do identify the position of the object in the image that it had recognized where as histograms do not due to the removal of positional information when represented as a histogram (it may be possible to find out positions using the histogram matching techniques but this would be an extension of the ideas presented in the above systems).

2.4. 3D Approaches

3d approaches to computer vision build a three dimensional representation of the images or parts of the image presented to it. Normally the camera has to be calibrated in someway so that depth and/or distance are known allowing the formation of projection matrices to handle the effects of perspective projection. See [M.Sonka, V. Hlavac, R.Boyle a, 1999] for a more detailed description of the geometry involved in 3d vision. Various methods have been described in order to derive this information from an image including shape from optical flow, motion, texture and contour [M.Sonka, V. Hlavac, R.Boyle b, 1999]. Most of these methods require that that the camera is in a known position or have methods that allow camera position to be automatically configured. Specialized hardware is sometimes used such as the camera system described in [A.Johnson et al. 1995] allowing the camera to be moved within a known or predictable range. Other specialized equipment such as stereo heads with two cameras attached to a fixed point as in the BICLOPS system by TRACLabs [TRACLabs, 2003]. Other examples include detecting objects in space by use of light reflecting or laser reflection measurement as in the case of the VI-900 3d laser scanner by MINOLTA [MINOLTA, 2003]. These methods can produce very good results but they usually require specialist and dedicated equipment to acquire the objects in a way that is suitable for 3d Matching.

Figure 4 - BICLOPS PVT system hardware from TRACLabs. Note the two cameras mounted upon a rigid base.

Once the models have been acquired then an attempt at matching can begin. One such system that searches 3d models based upon a sketched 3d representation of the required model is described in [P. Min et all, 2003] along with an interactive demonstration of the system available at [Princeton Model Search Engine, 2003]. Again the matching works once a 3d representation has been obtained of the original object.

2.5. Biological based Approaches

Seeking inspiration from the brain to provide a way to build a visual system G. Wallis and E. Rolls described such a system in their paper “A model of invariant object recognition in the visual system” [G. Wallis, E. Rolls, 1996]. The system uses competitive learning methods and four ‘layers’ arranged in a hierarchical fashion. The network is organized so that many cells can be fired in the first layer but only results in firing of one neuron within the top layer. The model has been further developed from its original Vis-Net model to a more advanced version that responds to simple features presented to it in an image, the features include lines at many orientations which the system was able to distinguish. This development is detailed in “Invariant recognition of feature combinations in the visual system” [E. Rolls, M. Elliffe, S. Stringer, 2002].

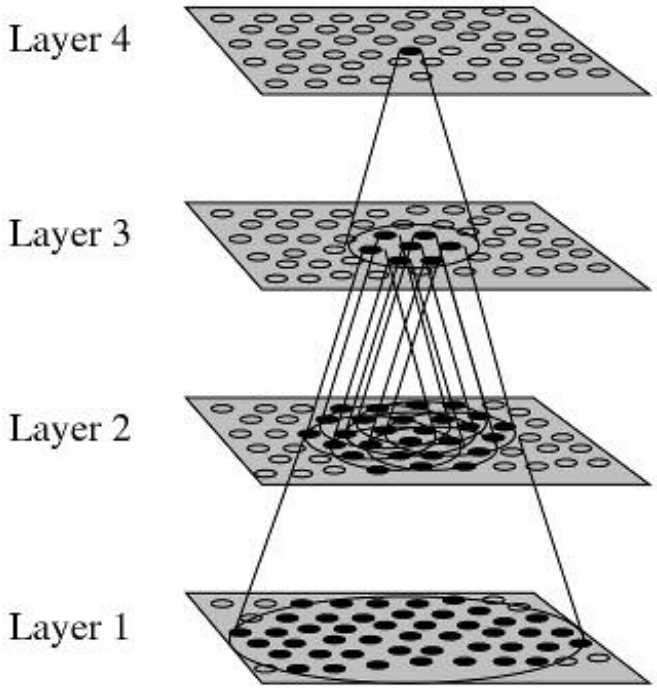


Figure 5 - Stylised image of the Vis-Net four layer network. Convergence through the network is designed to provide fourth layer neurons with information from across the entire input retina.(Original Caption) (Adapted from [G. Wallis, E. Rolls, 1996])

The system is quite complex and at present has only been thoroughly tested with simple combinations of lines at many orientations and is still some way off of becoming a complete vision system. The researchers original goal was to explore the hypothesis presented in the 1992 paper Neuropsychological mechanisms underlying face processing within and beyond the temporal cortical areas by E.T. Rolls[E.T. Rolls, 92]. The system or at least the ideas presented in it have potential to develop into a powerful and biologically plausible visual system but the work is more focused towards exploring ideas and hypothesis than to produce a robust vision system at this stage.

2.6. Vision systems specific to chess playing

There have been few applications specific to chess playing which may be surprising considering that developing strategies for playing chess is very well researched within the artificial intelligence community. One approach presented in [S. Brunner, E. Pohn, 2003] ASP5-Teilprojekte zum - “Turkish Chess-Player”, is that the chess board and the ‘robot’ that a human would play against are both computer generated. This solves the problem of where the chessboard and the pieces by allowing the computer to control their position within the space. The problem then becomes one of detecting where the human player’s arm is and which piece it would be moving. This is almost the reverse of the problem that is attempted to be solved within this project.

An approach where the chess board is real and the computer is presented with an image of the board and the individual pieces is described in a project at Stamford University [M.J. Kochenderfer, J.G. Nichol, J.L. Jacobs, 2002]. The Visual Chess Project by M.J. Kochenderfer, J.G. Nichol, J.L. Jacobs, detects the most likely position of the corners of the chess board using a specialized corner detector and maps it to a plan view of the board. Moves are detected by comparing current and previous images to one another to determine that changes in positions between the two boards.

2.7. Evaluation of the systems for purpose of the project

This evaluation is to attempt to identify those techniques that are applicable to the project and which are not.

Many of the systems have been developed over a number of years and there are examples of successful implementations using practically all methods described above. Within the scope of this project there were constraints imposed that meant some approaches or at least parts of approaches were not possible. As this project is an undergraduate project it would be unreasonable and impractical to be able to obtain some of the specialist equipment that is used within 3D- Vision applications. Although some of the principles of 3d vision are applied with regard to finding the chess board within an image, such as the provision to move a robot arm to control the movement of pieces would require knowledge of the chess board's position in space, 3d vision is not a good solution to problem within the scope of this project.

Template matching methods remain attractive due to their relative simplicity to implement within the time constraints. However simple template matching may require a prohibitive number of example images to cope with all possible rotations and orientations of the chess board that the system would be required to cope with. Still template methods can provide a simple and quick filter to identify areas that should be investigated further, such an example could be to match the black and white areas that characterise a chess board.

Biological systems like those presented in [E. Rolls, M. Elliffe, S. Stringer, 2002] are perhaps not well developed enough to give the confidence to use such an approach within this project. The system is also highly complex and perhaps beyond the abilities of the project manager. The Vis-

Net system also works with simple examples and perhaps is more of a research tool for those investigating biological vision systems rather than an approach to recognizing and identifying objects within an image. The SEEMORE system also claims to be “Neurally inspired” and combines filtering with templates. This approach is appropriate for the chess system for identifying whether certain objects are present within an image. In [B. Scheilde, J. Crowley, 2000] the system also includes a provision for detection of objects within cluttered scenes. Both of these systems use histograms in one form or another as a means to identify objects, one of the main reasons for this is that position information is removed allowing the systems to recognise objects independently of orientation. This approach poses a problem for the chess playing system as such a system must be capable of accurately identifying the object’s location within the image, to identify board positions and movement of a robot arm.

Snakes and AAM allow a object to be identified along with its position within a image and so would provide a good approach to solving the problem. The negative side of AAM based approaches is that a set of landmark points must for each training image in order to work out the allowed variations within the model. This training is not such a problem as it can be automated once there exists a set of models, but the main problem is that the landmark points have to be set by hand or at least carefully checked to make sure the points are set correctly. This is not such a problem as this project deals with only the recognition of chessboards and pieces but care should be taken to ensure the system is not too specialized so it can cope with the wide range of boards and pieces that could be presented to the system.

In practice certain elements have been taken from many of the separate areas. The AAM model has been used as a basis for the board detection component of the system and methods from 3d systems have been used

to allow a ‘plan view’ of the board. Filtering techniques that are common throughout all of the approaches have also been used to quickly help determine the existence of features within an image.

3. Board Localization

3.1. Candidate methods

Before any decisions can be made as to the specifics of the algorithm to detect the board a number of methods will be tested and evaluated. This section attempts to identify the main feature detection methods that will be used to identify the board.

3.1.1. Board detection

The problem of board localization is to find out the possible locations of the board. In this particular problem the objective is to find the outline of the board within an image to allow the detection of the current state of the game. The reason for performing this operation is to identify the board outline so that further processing can be performed to identify the individual positions of the squares. One of the most widely used algorithms for detecting lines is that of the Hough algorithm. As this was the most well known methods for line detection this was the first method implemented and tested and the method and results are detailed below.

3.1.1.1. Hough

The Hough operator is essentially an evidence gathering method that was first described in [Hough 1962]. The method requires that the image is first run through an edge detector (such as sobel or canny) before the Hough algorithm is applied. Some of the results are shown below.

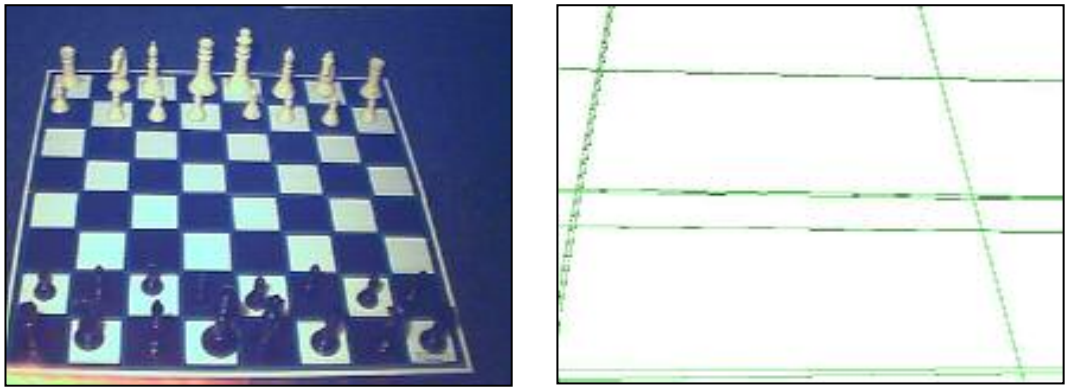
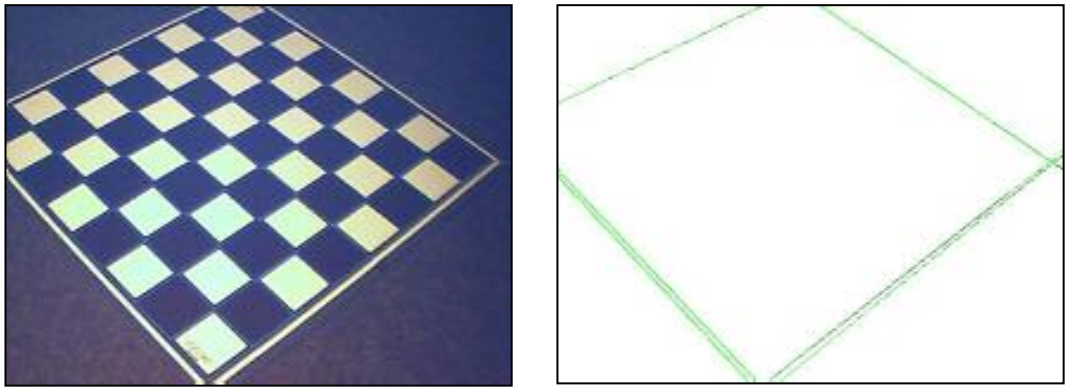


Figure 6 - Original image and hough processed image



T

Figure 7- original image (Rotated) and the hough processed image

The Hough operator performs very well in practically all rotations of the chess board, as would be expected. The presence of more lines than just simply the edge of the board is shown in Figure 6 and 7, this effect should be expected due to the image having lines made up of the chequered pattern. It is possible to compensate for this problem relatively easily, if you know the edge of the board then these additional lines can be discounted. Another possibility is to set a threshold to determine when the line should be drawn and when they should not. This method although effective at determining lines from all rotations, is not sufficient to identify the position of the chess board.

Many lines may be detected but the problem is now to identify which lines are those of the border of the chess board and which are those of internal lines or other objects in the image. It should also be noted that the strongest lines are those of the edge of the physical board that includes a border, although in this example the border is relatively small and would only produce small errors when approximating the individual squares this error would increase substantially when dealing with larger borders around the board.

3.1.2. Snake based approaches

3.1.2.1. Snake

The idea of a snake resulted from the work of Kass,Witkin, and Terzopoulos [Kass et al. 87a, Witkin et al. 1987, Terzopoulos et al 1987]. The applicability of the snake in this situation is to identify regions for further exploration. As the snake contracts it also forms a model of the outline of the feature it has detected. The final convergence of the snake will provide a connected graph representation of the image feature detected. This could then be used as a way to determine what shape has been found and if it's a match to the one that we seek, in this case the chess board.

The snake is used to enclose a target feature in an image. It is made up of a set of individual points that are joined together in a sequential manner. The snake starts outside the feature to be detected and then iteratively evolves to enclose a region. The general snake algorithm can be shown as:

$$E_{\text{snake}} = \int_{s=0}^{1} E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s))\,ds$$

Equation 1 : Basic Snake Equation

Internal energy (E_{int}) controls the behaviour of the snake and also the arrangement of the snake points. E_{image} are the image features to which the snake is attracted to (e.g. edge contours), the constraint energy (E_{con}) is allows other higher level information to guide the snakes evolution. There may be many additional constraints that operate upon the snake. Two of the most commonly used constraints are those of contour energy and curve energy. The contour energy constraint attempts to ensure that all points have a similar distance between them to cover the image as completely as possible. Curve energy controls the shape of the snake and states whether the snake can represent right angles or whether more gentle curves are preferred. Both of these constraints were used within this implementation of the snake algorithm, a detailed description and analysis of why these constraints are used will not be given here but one is given in [M. Nixon, A. Aguuado d, 2002].

These constraints are weighted so it is possible to get different behaviours of the snake, although the image constraint (E_{image}) should be given a high weighting to make the snake responsive to the image data that it is searching. The snake method is also robust to rotation and scale as the same algorithm can operate on any scale and rotation of an image feature. Due to the guided search nature of the snake larger image sizes do not effect the time required to process that image in the same way as if every pixel was to be processed.

The snake's initial position is set either by hand or by starting at a specified region of the image. The snake does successfully enclose the board in these images. The evolution of the snake can be viewed in the subsequent images.

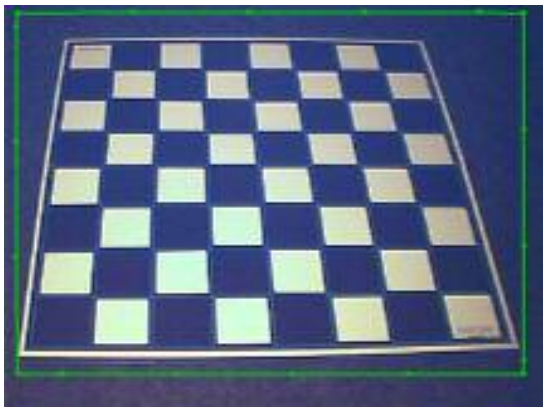


Figure 8 - Initial Position of Snake

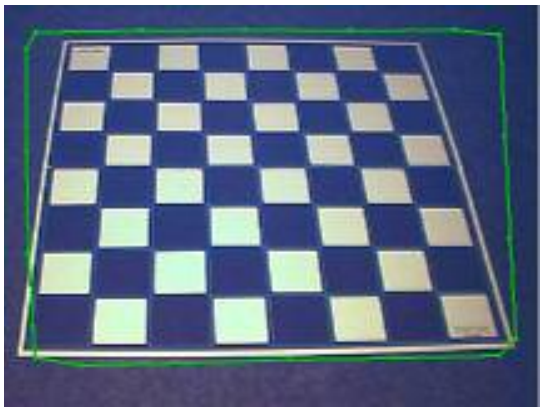


Figure 9 - After 25 Iterations

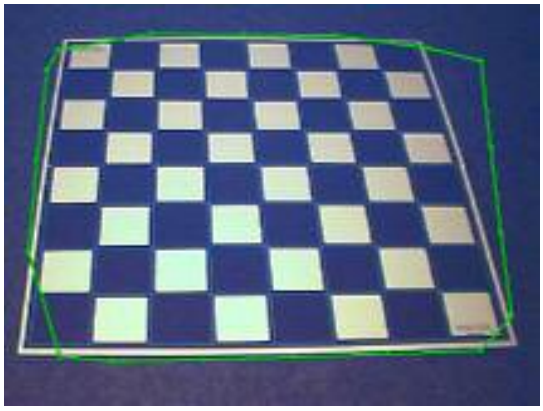


Figure 10 - After 50 Iterations

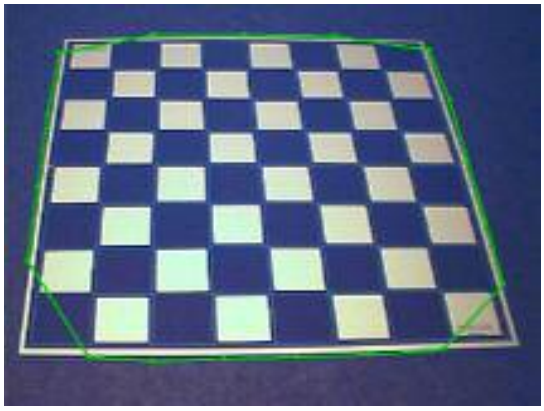


Figure 11 - After 100 iterations, no more movement

Within this relatively uncluttered scene the snake performs very well but this example has been deliberately chosen to illustrate the snake's evolution. Within a more realistic scene the effectiveness of the snake is reduced. There is also the problem of the initial positioning of the snake. If one were to simply start the snake on the edges of the image it would stop on the first image points as illustrated below. The snake converges on the table boundary that due to the high contour energy there (given by the edge filter). This could be overcome by giving the other constraints a higher weighting but then the snake would converge to a more rounded shape as well as the snake not stopping on the edges of the chess board itself, which is the very thing we are searching for (as is shown in figure 13).

Figure 12 - Snake caught on incorrect image features

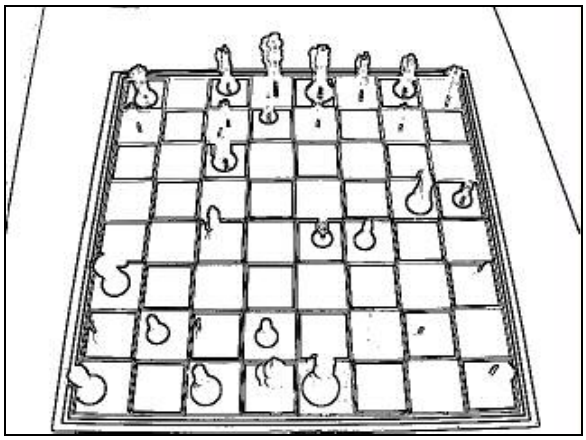


Figure 13 - Sobel edge detected image

Figure 14 - Higher weighting given to curvature and elasticity constraints – the snake doesn't stop at the original boundary of the chess board.

In its present form a snake would be unable to correctly identify the presence of the board within an image region the properties of scale invariance and smart searching make it highly desirable. The original algorithm also allows the specification of any features to be searched upon and any constraints to be used with the basic energy minimisation process. It would be useful to combine the snake model with a desired shape that it is to find to avoid this problem and still retain the benefits of the snake approach.

3.1.2.2. Active Shape Model

The previous section was concerned with a basic snake following simple rules to allow the enclosing of regions within an image. The work of C.J. Taylor, G.J. Edwards, T.F. Cootes in their paper Active Appearance Models describe one such approach to incorporating additional constraints upon the standard snake model and is mentioned in section 3.3 of this report. These models consist of individual nodes that are allowed to reside within a certain elliptical distance from the original model.

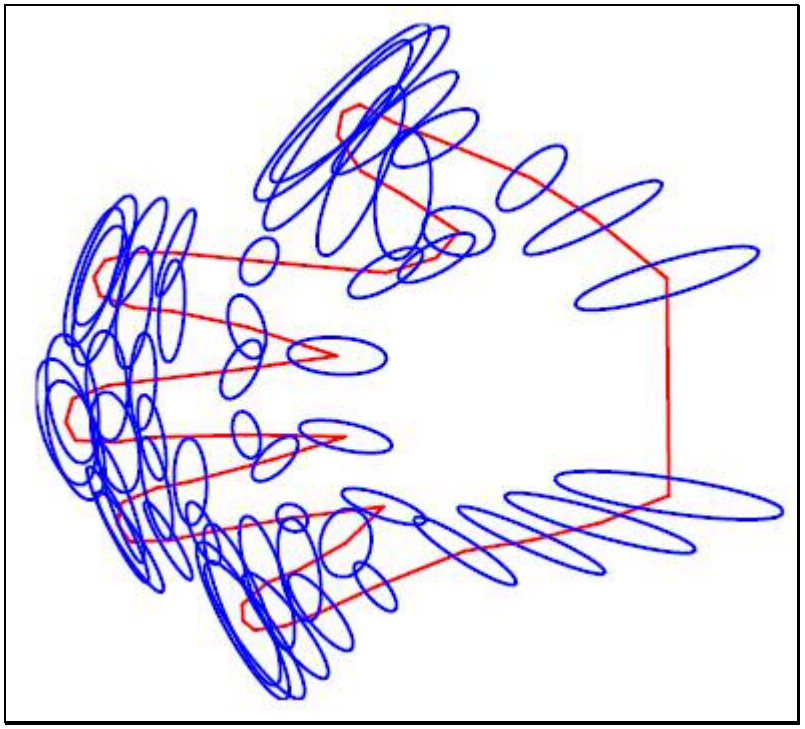


Figure 15 - Allowable point variation of a hand model

The actual implementation of the shape model is rather complex and is still under research especially in the area of model learning. For the purposes of the project this idea was changed and simplified, the reasons for this were:

- Methods were still in research stages and the results were not widely tested to give the required confidence to use the method.
- The board detection problem is simpler and more constrained than that of finding the 'one size fits all' method being proposed by these models. Although the board finder solution should be applicable to other situations the additional complexity of these models was felt to be too high.

As such an adapted method was developed.

Image Energy

The original snake algorithm operated upon an edge filtered image and used this to determine the energy of the image at a particular point (E_{image}). The shaped snake uses a specific filter upon the image to reveal the presence of features that are being searched for. The node types and the filters used are explained below.

The snake is primarily concerned with the detection of the chequered pattern of the chess board. The filters used are a simple convolution operator that is centred on the individual point's position in the image. The orientation of the filter is also determined by the snake.

1	1	0	-1	-1

1	1	0	-1	-1
0	0	0	0	0
-1	-1	0	1	1
-1	-1	0	1	1

Table 1 – Filter at zero degrees

0	-1	-1	-1	0
1	0	-1	0	1
1	1	0	1	1
1	0	-1	0	1
0	-1	-1	-1	0

Table 2 - Filter rotated 45 degrees

Orientation of filters is determined by the average normal of the line passing through a node, this ensures that the snake work at any orientation.

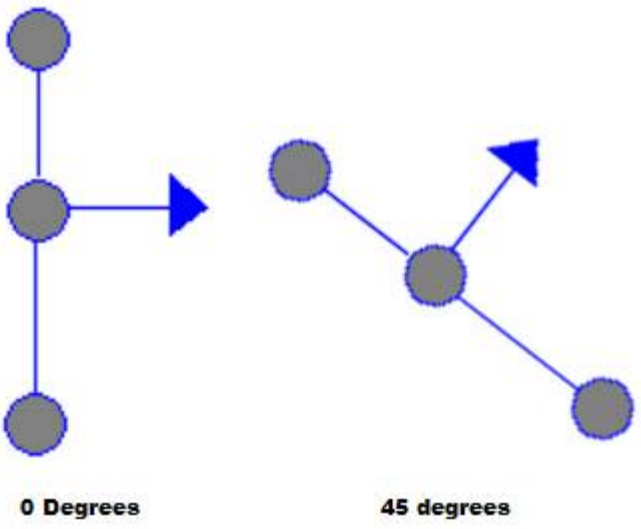


Figure 16 - Filter orientations

Evolution constraints

In order to successfully represent the shape of a chess board the chess snakes constraints consist of.

Line Energy Constraint

Nodes that make up one side of the snake are to be kept in a strait line with each other. The line is the line of best fit of all the points along a side.

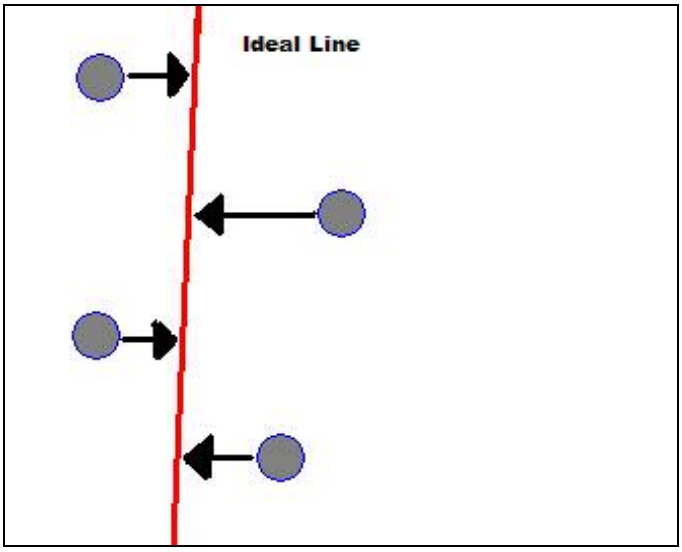


Figure 17 - Point movements resulting from the line constraint

Contour energy

Nodes should not get too near each other. When a node gets within a certain distance of another the energy starts to increase so this position is less desirable and the node should move away again. This rule is the same as the standard snake with just the implementation details differing.

Evolution of the Snake

The snake is started outside of the feature and slowly evolved to enclose the image region.

Results

The shape snake has had mixed results. The rules do indeed keep the strait line features that are characterised by the chess board but the complexity of the evolution of the snake means that matches cannot easily be found. The snake consistently ‘misses’ the edges of the board due to the edges of the board not being strait. This can be catered for but the problem of how to evolve the snake remains. If the snake simply shrinks it will miss edges that are slightly out of the image. The method works well for simple test cases but is not generally applicable as a way of finding image features.

3.1.3. Feature Detection

All of the methods described work with a pre-filtered image that emphasises the features that are of interest. This section describes several feature detectors that are used to help find the chess board within an image.

3.1.3.1. Corner detection

The inclusion of corner detection was included in the project as a way to identify areas where the corners of the chess pattern were located. By detecting the corners that are present in the image it is a first stage in locating the.

Method

The corner detection method used is that of the SUSAN corner detection filter. This filter was chosen for its speed, simplicity and reliability. The method is described in [S.M. Smith, 1995]. The method involves placing a circular mask of a over an image region. The area covered by the mask is known as the Univalue Segment Assimilating Nucleus (USAN) with the centre pixel in this mask known as the nucleus. Throughout the mask each pixel's brightness is compared to that of the nucleus so to define an area which has a similar brightness as the nucleus (SUSAN). This region finding within the mask area both one dimentional image features (edges) and two dimentional features (corners). A USAN is at maximum when all the pixels in the mask are the same as that of the nucleus (e. in Figure 18). When an edge is present this falls to half (b in Figure 18) and even further for corner features (a in Figure 18).

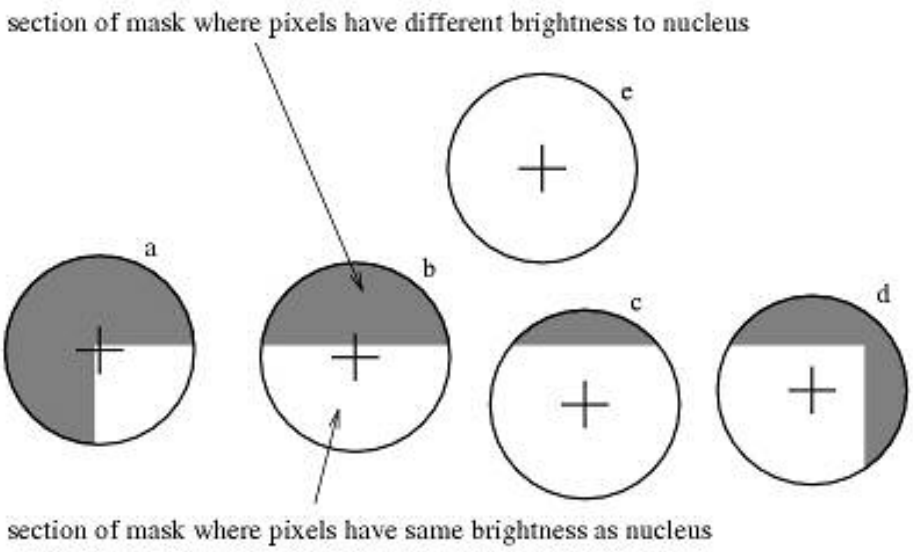


Figure 18 - Four circular masks with similarity colouring; USANs are shown as the white parts of the masks. [Adapted from S.M. Smith, 1995]

The equations used to perform this process are:

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}).$$

Equation 2 Comparison equation, all pixels are compared to the nucleus and that comparison result is summed.

Where:

n is the area of the USAN (number of pixels).

$\vec{r_0}$ is the nucleus and $\vec{r}$ is any other point within the mask.

$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}) & \text{if } n(\vec{r_0}) < g \\ 0 & \text{otherwise,} \end{cases}$$

Equation 3 – Subtracting the USAN size from the geometric threshold

Where:

g is the geometric threshold

$$c(\vec{r}, \vec{r_0}) = e^{-(\frac{I(\vec{r}) - I(\vec{r_0})}{t})^6}.$$

Equation 4 - Comparison function

I(r) is the image brightness at the particular point.

t is the threshold that determines the minimum contrast of edges that the filter will respond to.

The corner detection algorithm used consists of five steps:

1. Place a circular mask around the pixel in question (nucleus)
2. Calculate the number of pixels in the mask that have a similar brightness to the nucleus (using Equation 3).
3. Subtract the USAN size from the geometric threshold to produce a corner strength image (using Equation 4)
4. Test for false positives by finding the USAN's centroid and its contiguity.
5. Use non maximal suppression to find corners.

As this method is well researched and has been extensively tested there was a high degree of confidence that the method would work as expected. However parameters may need to be tuned and the speed of the implementation would need to be tested for detecting corners on the chess board.

Results

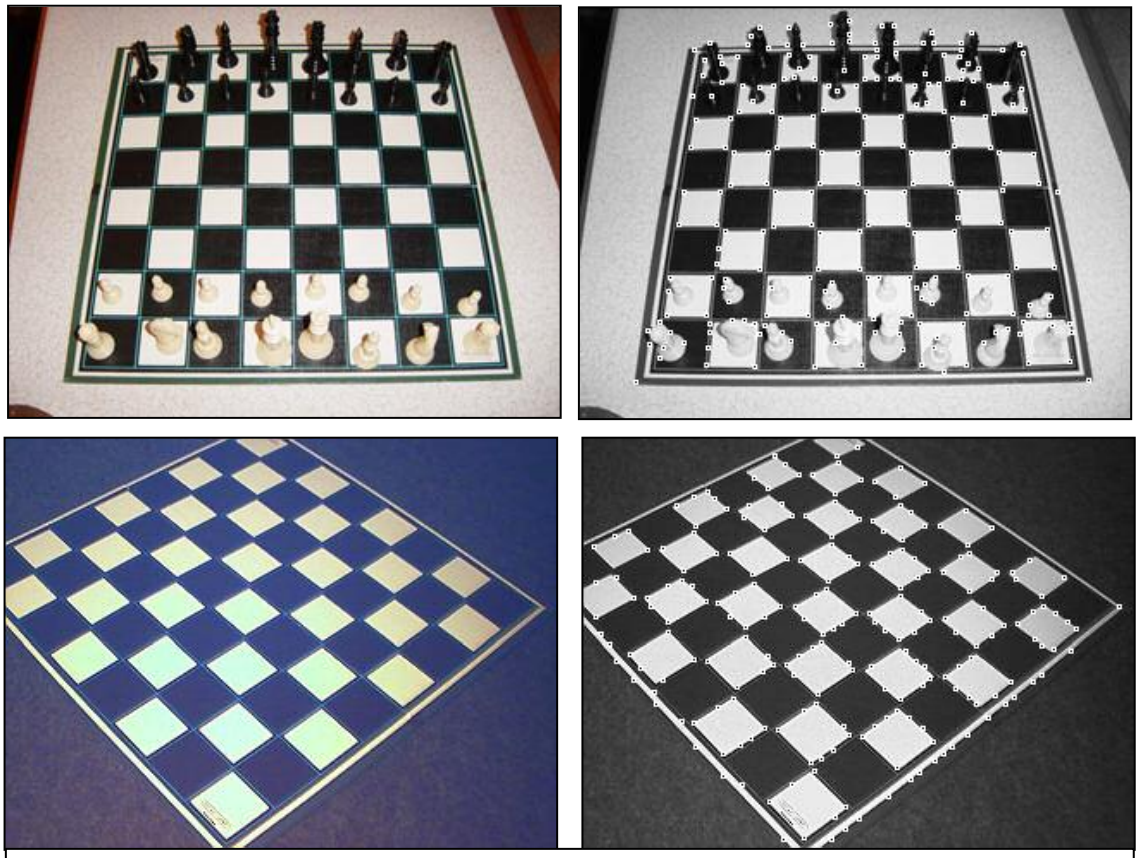


Figure 19 - Before and after images of corner detection using the SUSAN method.

The results show that further processing is needed to ensure that only the chequered pattern of the board is registered. Also there are responses that are not corners but caused by the jagged lines that are a result of the digital approximation of a line in the image.

3.1.3.2. Specific Feature Detecting Filters

This section details the filters used to detect features that are of interest when attempting to localise the chess board. The aim of this is to find out small and localised areas that form part of the larger shape that is to be recognised.

3.1.3.2.1. Convolution

Although the corner detection will identify areas where checked patterns are present further information is needed to identify areas where the patterns are located. The purpose of this stage is to produce high responses to the features searched for. In this case we are interested in the areas where the chequered patterns meet. The filters response should be greatest at this point in the image.

Method

In this particular project the image features that are to be found are those of the check pattern present on the chess board. For this purpose a matrix that responds favourably to these type of patterns has been used. To enhance the check patterns that are present on a chess board a matrix of the form -

1	1	1	1	1	0	-1	-1	-1	-1	-1
1	1	1	1	1	0	-1	-1	-1	-1	-1
1	1	1	1	1	0	-1	-1	-1	-1	-1
1	1	1	1	1	0	-1	-1	-1	-1	-1
1	1	1	1	1	0	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0
-1	-1	-1	-1	-1	0	1	1	1	1	1
-1	-1	-1	-1	-1	0	1	1	1	1	1
-1	-1	-1	-1	-1	0	1	1	1	1	1
-1	-1	-1	-1	-1	0	1	1	1	1	1
-1	-1	-1	-1	-1	0	1	1	1	1	1

Table 3 - 11 by 11 matrix used to find chequered patterns

The pixel currently being considered is in the centre of the matrix – highlighted in red. Each pixel is multiplied by the value of the matrix for that point and the result of this multiplication is added to the result. The produces a result or a score of how likely it is that the feature is present for that particular pixel. As every pixel will produce a result of some type only the top percentages of the scores should be considered.

This method also has an additional advantage for detecting the orientation of a chess board as patterns certain patterns produce a high positive response whilst others produce a low negative response. This absolute value of the response can then be taken so that both cases respond highly.

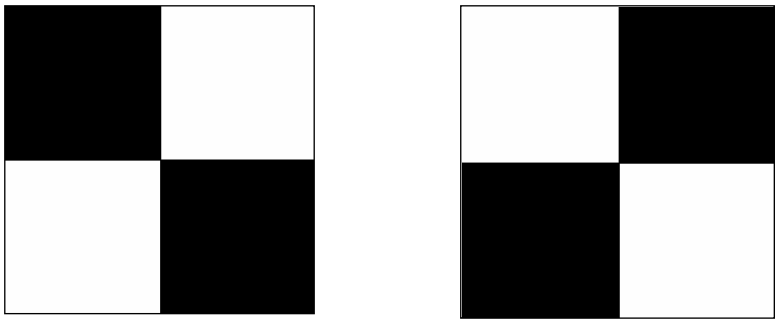


Figure 20 - The two combinations of the chequered pattern
High positive response = 1275 Low negative response = -1275

This method is not without problems. The main one is that it is very sensitive to rotation. If the picture is rotated a sufficient amount the response that is produced by this method falls dramatically. There are several approaches to solve this problem. One is to rotate the filter at various degrees and try and match it to the best response. This would only need to be done between 0 and 45 degrees before the filter mask would start to repeat itself as it responds to the differences rather than the colours themselves this is shown below.

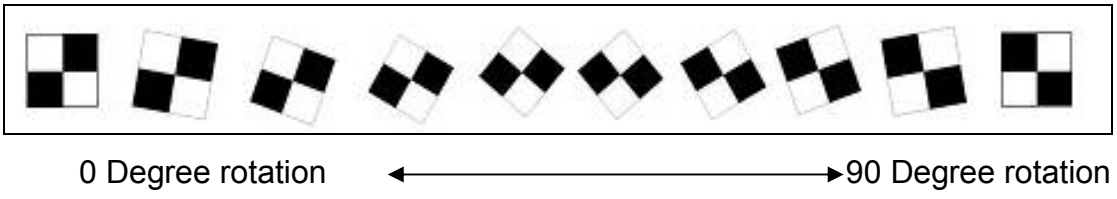


Figure 21 - Rotated image sections that should be found

The amount of computational power and therefore time to compute responses at the various rotations is quite significant. Due to the time that can be taken to apply these filters methods of speeding up the process or alternatives have been considered.

Results

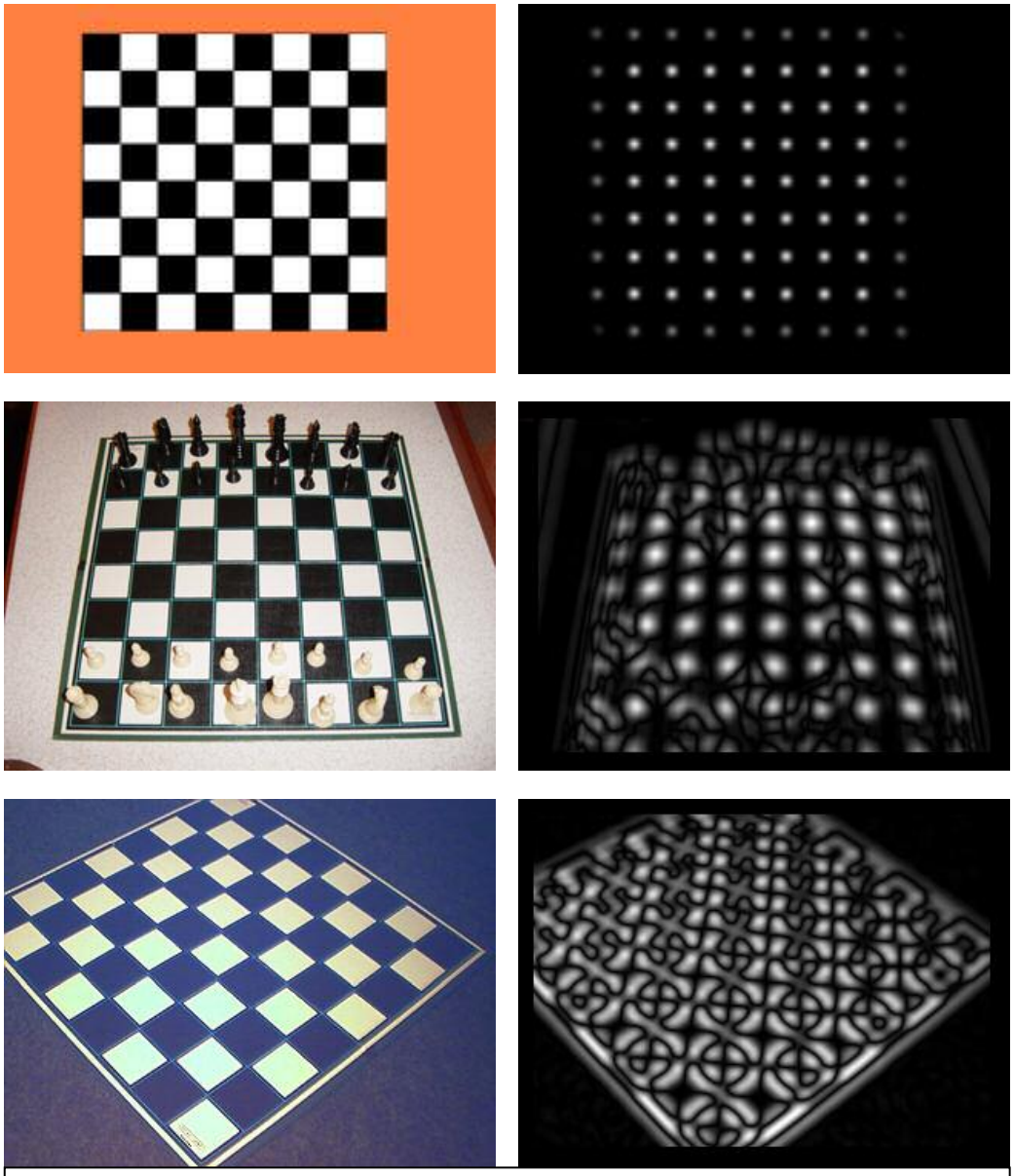


Figure 22 - Original images and results of applying matrix in table 3 to every pixel

One problem that is visible right away is the performance of the matrix filter when the board is rotated. This is rectified by using a rotated version of the filter so that the features are correctly identified.

3.1.3.2.2. Rotational invariant filters

The problems of using static matrix filters are illustrated in the last section and in particular in figure 16. The matrix approach gives very good responses for detecting features when applied to an image. The problem with this approach is that many filters have to be run to determine the best response, although in practice this can be reduced to two or three rotations the problem remains. If the detection of features could be completed in one time step this would significantly improve performance.

Method

The filter is made up of both changes in colour/intensity and the length of the sections. The current feature we are searching for is that of the check pattern as shown below.

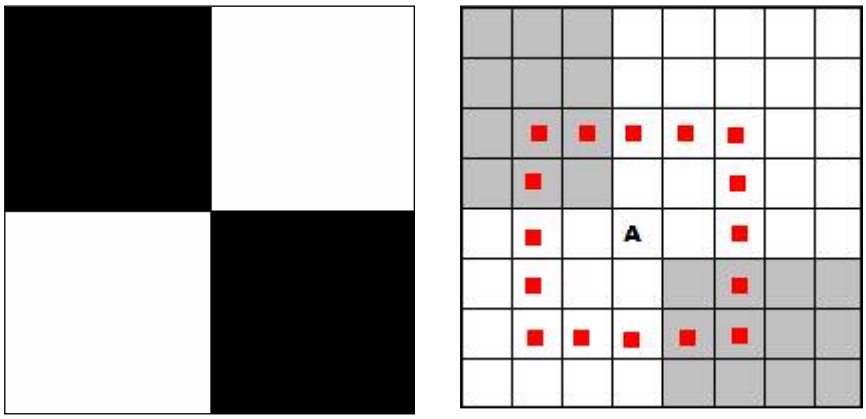
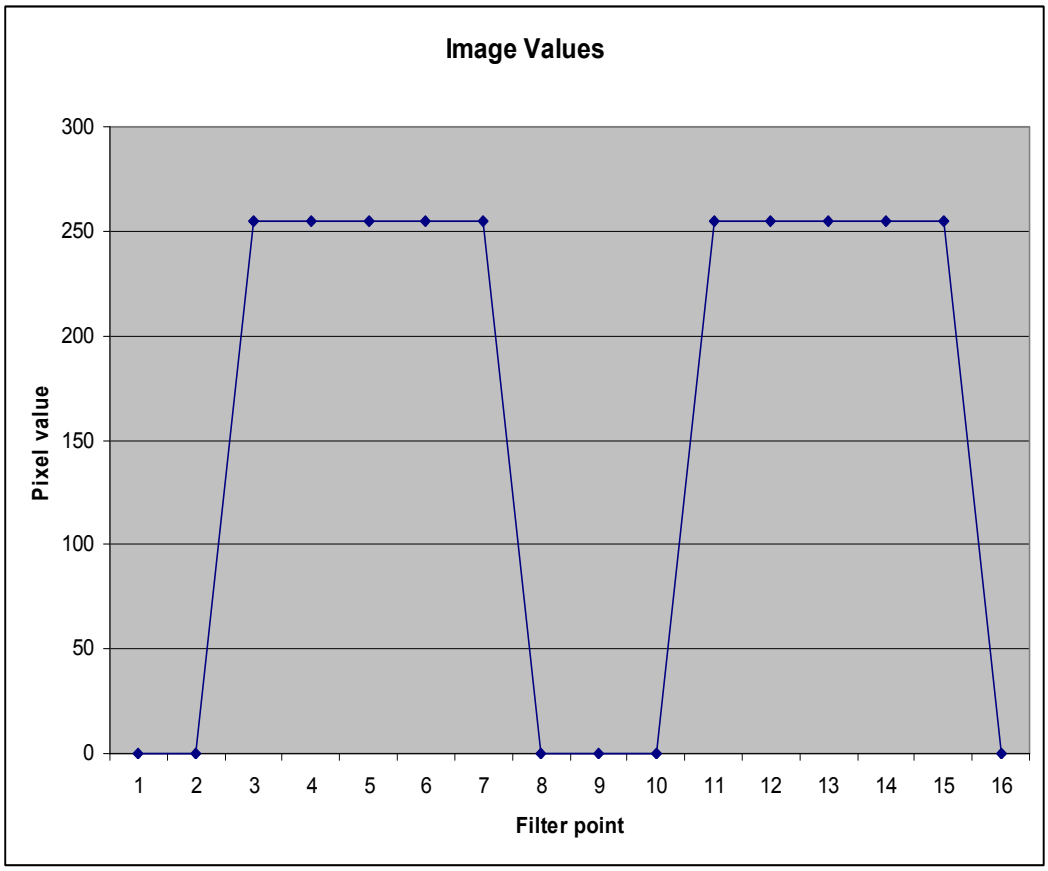


Figure 23 - Original image segment and filter operating on it

Assume that A is the image point currently being considered by the system the filter points are shown by the red points, in total there are 16 points, the filter can be described as having a radius of two as it extends 2 pixels in each direction. The filter points are assigned the value of the pixel at that point. The points of

the filter are ordered in a clockwise direction, this filter would work equally well if anticlockwise direction is chosen.

This information can be represented on a graph. The last point is 'wrapped around' so that the point after the last point is the first.



Graph 1 – ordered pixel values of the filter.

To use this information in the filter we will search for areas where there are 4 changes and where the peaks or troughs are of similar size. This method will work for the various rotations in the image that will be encountered.

Results

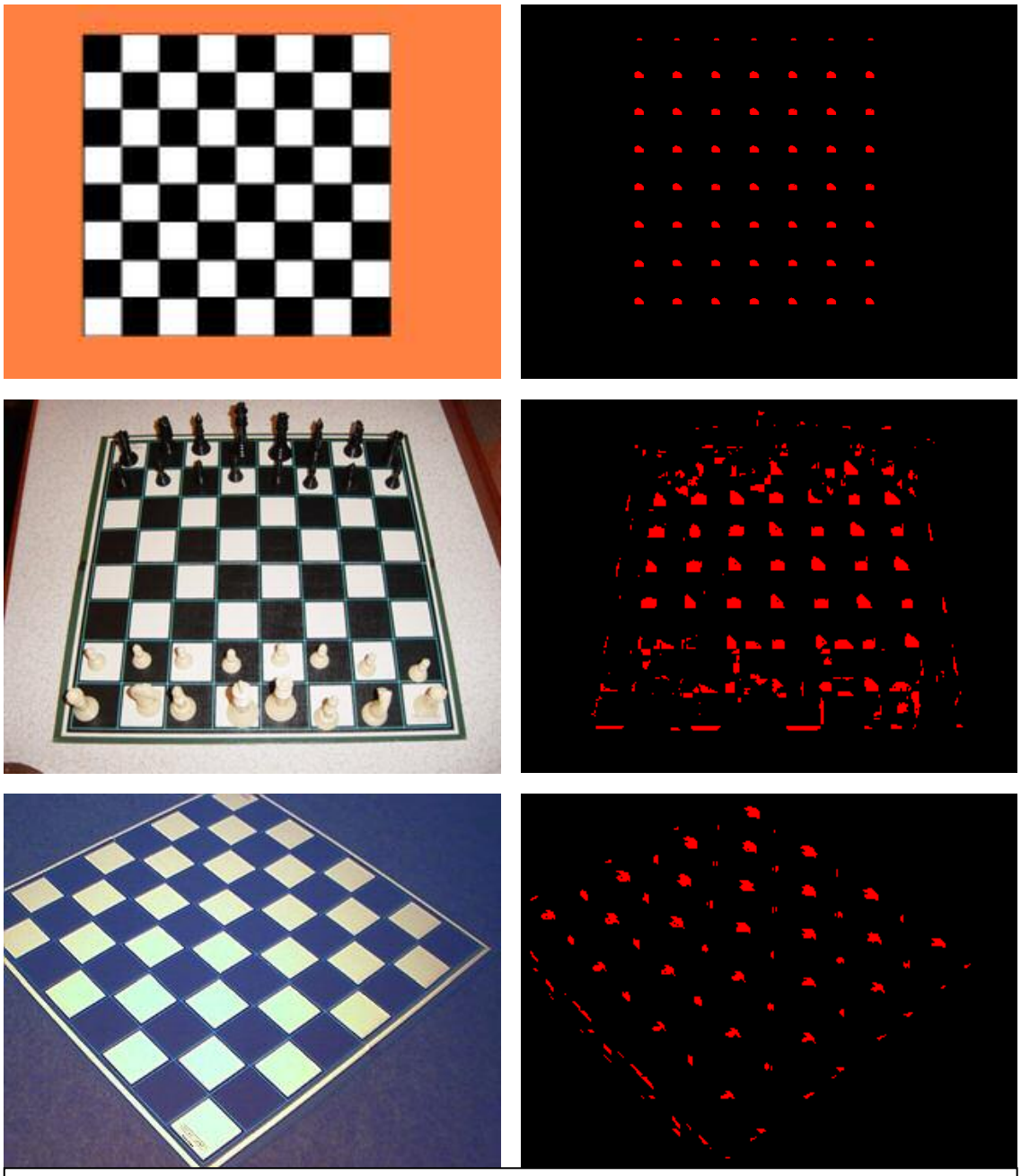


Figure 24- Original images and results of rotation filtering

The filter performs well under the various rotations of the features. The problem that can be seen in the images is that of responses around the edge of the board. This is due in this case to the border providing a similar graphed response as that of the check pattern. The borders have a similar number of changes and a the size of the peaks and troughs are similar to that of the chequered pattern.

3.1.4. Self organising map approach

Responses produced by both sets of filters need to be utilised in some way so as to fit a model of the board to the data. The Kohonen self organising network is used within artificial intelligence to represent the input data, in this case output from a filter. More frequent inputs are better represented, as well as this the network is topologically organised. This fits the problem well as we need to represent the output data but also keep the distance. The problem also 'looks like' a self organising network as it is a grid pattern. This idea was developed as an evolution of the snake approach where the image is filtered and then a model attempts to match the responses.

Self-organizing maps (SOMs) are a data visualization technique invented by Professor Teuvo Kohonen which reduce the dimensions of data through the use of self-organizing neural networks the method is described in [T, Kohonen, T, 1997]. Kohonen SOM networks can have many dimention, in this case only the two dimentional SOM is used.

Architecture:

Method (Algorithm)

Initialise Map
Repeat
    Select Sample
    Get best Matching Neuron
    Update Neuron Weights in Neighbourhood
    Reduce Learning Parameters
Until Finished

Training Data

The data that the network is trained upon is made up of responses from one of the two filters described in 4.1.1.3. The filter is used on the original image and the response image is used as input. Each response produces one training data sample, the data describes the position in the x and y direction. Thresholding of the filters responses is used so as to remove very weak responses that are not part of the feature searched for, this is illustrated in Figure 25.

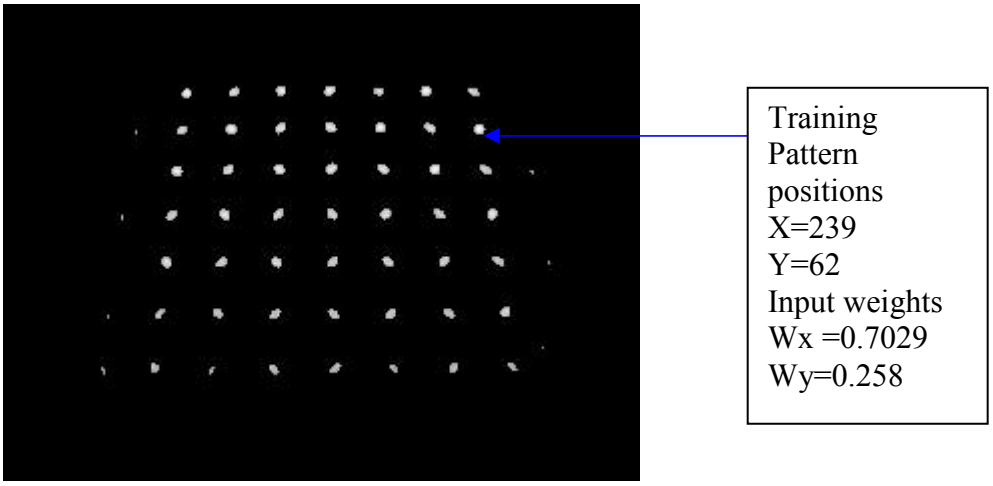


Figure 25 - Threshold Matrix filter response and training data extracted from it, threshold only copies the responses that are in top 40% into the output image. (Any non black pixel RGB = 0, is considered as a training pattern)

Initialising the weights

Within this setting there are two main ways to set the starting positions of the weights. The first involves randomly setting the positions of the neurons throughout the output space. This should mean that the space is covered well by the neurons and that any position they occupy is not forced upon them. However in this case we have a model that we want to fit to the data so it is acceptable to initialise the neurons so that they are already in a grid position. In practice both of the initialisation methods do not affect the evolution of the network greatly due to the large changes in position of neurons during the early stages of training. Differences are observed when the initial neighbourhood, and weight update parameters are very low so neurons can not change their weights significantly. However this makes for a very poor representation of the data and so this situation doesn't occur in practice.

The input data is normalised in both the x and y direction before it is presented to the SOM network. The normalisation of the input data is carried out so that the data can be scaled easily throughout many resolutions of images. The use of normalised data is now not widely used by SOM networks who prefer to use radial basis nodes that respond to an area of the input space without normalising. The use of radial basis nodes would be preferable, but by using normalised input it is easier to use multiple scales of images not just for the SOM network but also for further processing that is to be carried out upon the model.

Update weights

The updating of neuron weights is completed in two parts. The first part identifies the neuron with the weights that are closest to the current input weight. The node (*k*) with the closest weight vector (***w***) to ***x*** is determined by the distance between the two, with the smallest distance identifying the closest weight (euation shown below).

$$||\boldsymbol{w_k} - \boldsymbol{x}|| = \min_j ||\boldsymbol{w_j} - \boldsymbol{x}|| \qquad \textit{Note}: \|x\| = \sqrt{\left(\sum_{i=1}^{n} x_i^2\right)}$$

Equation 5 - Determining the neuron closest to the input weight

Once the closest weight vector has been found the amount that it can be moved towards the input vector is used to alter the weights. This parameter (phi) determines the maximum that a neurons weight vector can move towards a input vector. This sum is weighted by the distance from the winning node so that neighbouring neurons weight vectors in the neighbourhood are also updated towards the input vectors weights. The neighbourhood range is controlled by the parameter theta, determining the maximum size of the neighbourhood. Neurons in the neighbourhood are updated based upon a gaussian function centred upon the winning neuron so that nearer neurons are updated more than further away ones.

$$\boldsymbol{W_j}(t+1) = (phi * f(i,j)) * (||\boldsymbol{w_j}(t) - \boldsymbol{X}||)$$

$$f(i,j) = \boldsymbol{exp}(-1*dist(\boldsymbol{w_j}, \boldsymbol{w_i})^2) / (2 * theta^2))$$

$$dist(\boldsymbol{w_j}, \boldsymbol{w_i}) = sqrt(||\boldsymbol{w_j} - \boldsymbol{w_i}||)$$

Equation 6 - Weight update rules for the SOM network

where Wj is the current neuron vector and X is the current input vector. F(ij) is the neighbourhood response at a certain point in this case the guassian function. Phi is the amount to update the weights. Dist is the distance between the two weights. Theta is the neighbourhood range.

The function f(I,j) produces a graph like figure that show in Figure 26, the nearest nodes are changed more than those at the boundry which may not be changed at all. The function is centred upon the winning node.

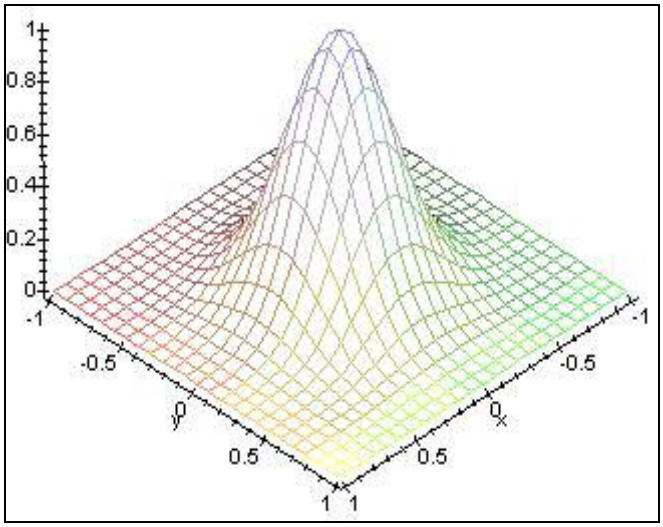
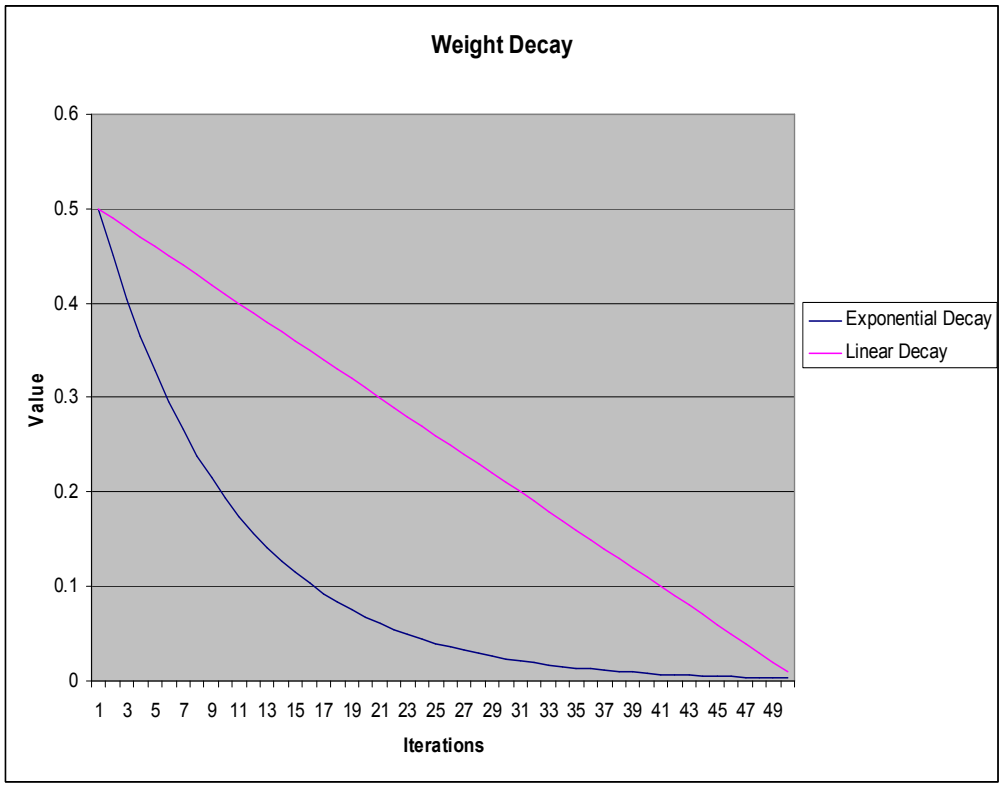


Figure 26 - Guassian function

Reduction of Learning rates

The reduction of learning rates is necessary in this case to enable the network to stabilise. High initial neighbourhood ranges and weight movement amounts are needed to ensure that the SOM can move sufficiently to represent the input data well, but these parameters need to be reduced so that the network can stabilize.

Originally in the implementation the weight decay was exponential (as shown on Graph 2). This led to the networks neighbourhood and update parameters decreasing very quickly and most of the time was spent in the lower values. This decay type meant that momentum values near 1 had to be used to keep the net moving for long enough to represent the training data. A linear decay was then introduced in order to spread out the activation throughout the evolution of the network. This was tested as the middle section of the evolution on the network was the most interesting part of the evolution where the balance between stability and plasticity.



Graph 2 - Weight Decay types Exponential vs Linear

Once the network evolution has been completed it is the final position of the weights that is the most interesting feature of the model. The network will be used as a model of the chess board with each node becoming centred where there is a cross section of the chequered squares. A network is usually interpreted by supplying input data and taking the node with the highest activation to be the correct 'output'.

The Self organising map was not considered to be the final solution to the board localisation problem as it was anticipated that other constraints would be needed to help guide the network to the correct positions. The results of just the Self organising map approach are presented to show how the approach works and to show the difference when additional constraints are added.

Results

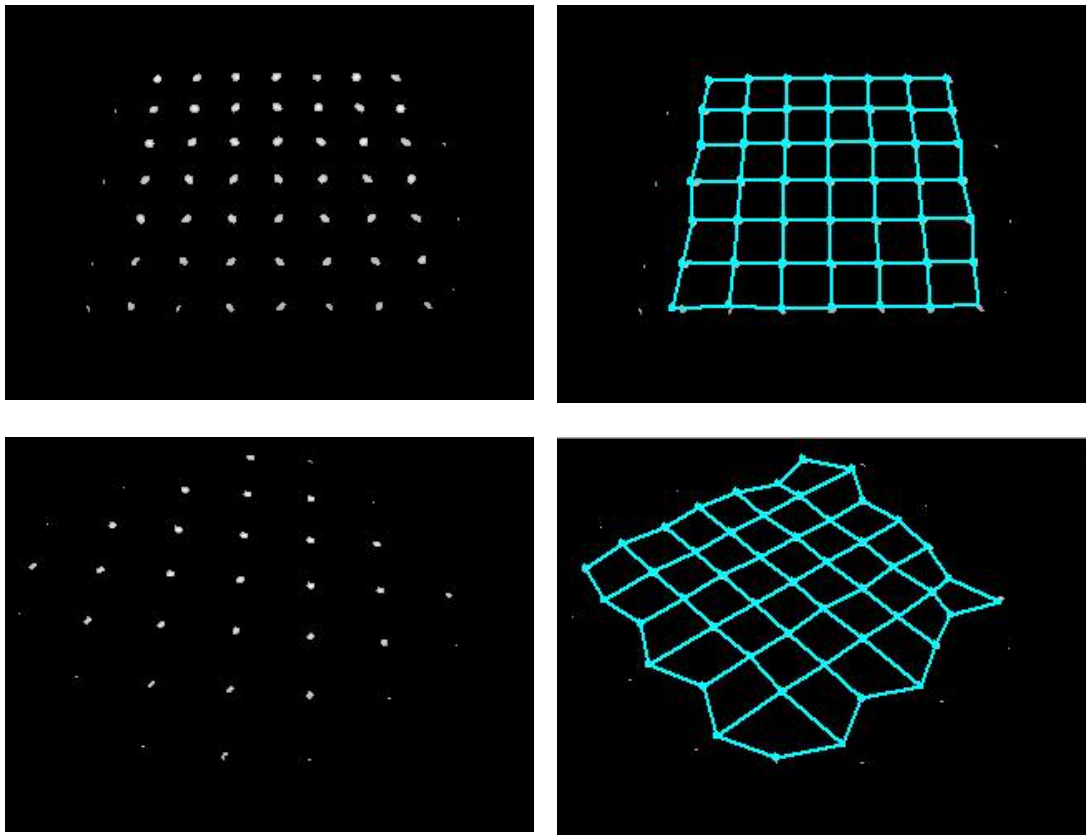


Figure 27- Input data and final SOM network position using a matrix filtered image as input, neighbourhood function = 0.5, weight update amount = 0.5, momentum = 0.9996 iterations = 2500.

The SOM network is a very good fit in the first instance but incorrect input data outside drags the neurons towards them. This is shown more on the second set of input data as the grid shape is lost around the edges of the SOM due to outlying data.

3.1.5. Shape Model Details

The self organising network will attempt to find the best representation of the input data based upon the frequency of inputs. The details of how the network is trained are contained in the previous section the results also contained in that section illustrated that further constraints would be needed so that the network could represent a pre-determined model, in this case the chess board.

The self organising network by itself contains no shape information as to how it should organise itself or where a node can and cannot be positioned if it is to accurately represent the object it is trying to identify. To help guide the evolution of the network separate rules are applied that enforce the shape restrictions. These rules are described below.

Each node is connected to at least two other nodes, with most having multiple connections to other nodes. The angles between the lines should be within a range for the shape to be acceptable. The overall model is made up of 49 nodes arranged in a 7 by 7 grid if only the inner dimensions of the board are to be matched or a 9 by 9 grid if the entire board is to found. It is usually easier to match the inner dimensions of the board due to the various borders that are used on different chess boards. Also if the border was to be found another type of filter would have to be used to identify the border areas and this would take a significant amount of time.

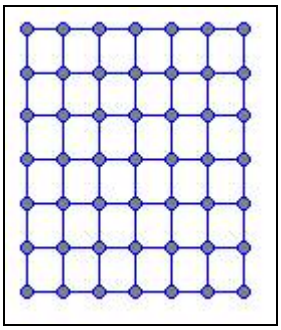


Figure 28 - 7 by 7 model of the feature to match

Corner Nodes

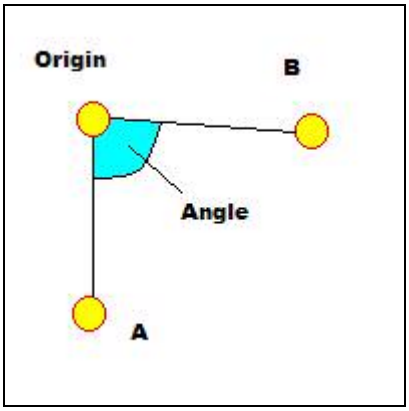


Figure 29 - corner node in the model

These nodes have two connections and represent the corner of the chess board. The ideal angle is 90 degrees between the lines defined by Origin to B and Origin to A.

Side Nodes

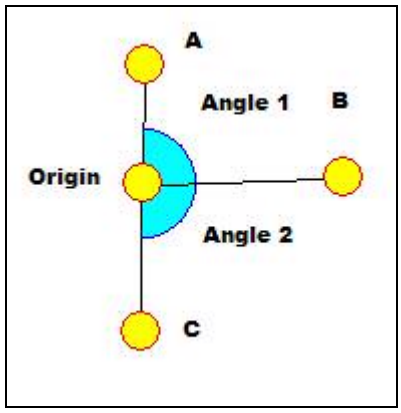


Figure 30 - Side Node in the model

Again the same principle is applied to the origin node except that there are two angles to measure.

Centre Nodes

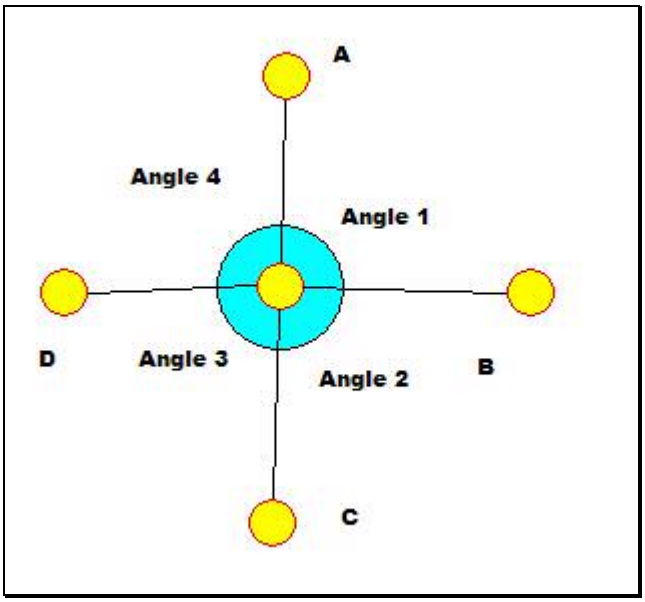


Figure 31 - middle nodes in the model

This case has the most connections and four angles to measure. The best angle between the lines is again around 90 degrees.

Movement of nodes

The node currently being considered is the only node moved. This is to avoid nodes with many connections being moved more than those with fewer connections. An example would be where if the origin node stays stationary whilst moving those nodes it is connected to towards the ideal angles those nodes in the centre would be moved twice as much as corner nodes. This would lead to instability in the model where each movement fights against the others.

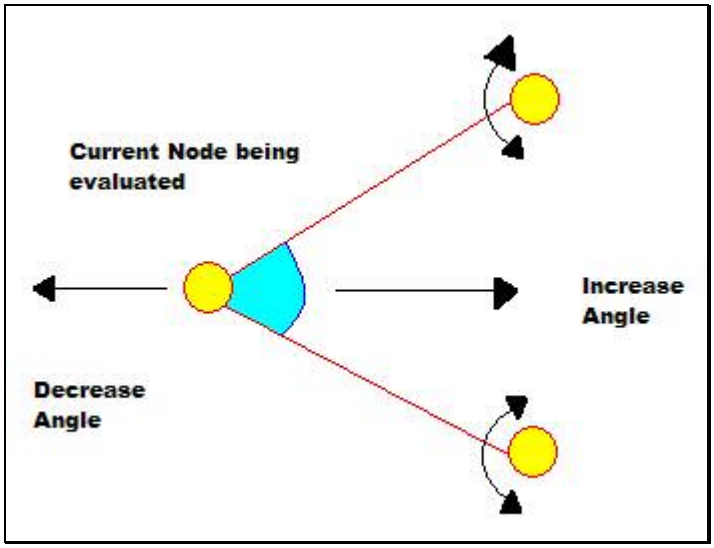


Figure 32 - Movement of nodes

The original node is moved in relation to the node it is connected to. The connected node either rotates it clockwise or anti clockwise to either increase the angle (shown in the above diagrams) or to decrease it depending upon the angle. The nodes are only moved if the difference between the ideal angle and the present angle is too large. This difference usually has some tolerance, for example if the ideal angle between two nodes is 90 degrees then the model should not aim for exactly 90 degrees. This is because small computational errors when working out the angles may mean that the node is constantly being corrected and would never come to rest at a

position. Another major reason is that the image being processed may have lens warping or large perspective effects and so the model will have to fit image features that do not adhere to the strict angle measurement.

3.1.6. Final Algorithm Details

The shape model itself needs to be combined with the self organising map to keep in shape whilst the data is fitted by the self organising map. To combine the two methods it is possible to either run one of them for a specified length of time and then run the other upon the result, or to run them both at the same time. Taking the approach of running one method until completion followed by the other method it seems that the SOM network would have to be run first and then the shape model applied to the results. This is due to the SOM network moving nodes significantly during evolution that would completely change the snake model.

The result of running the SOM network until completion and then applying the shape model does give a better match then running only the SOM.
The problem with this approach is that the shape model's evolution is independent of the data, it doesn't use it to guide it, it just adapts the shape of the SOM network to match the criteria.

To combine the shape model and the SOM more closely an approach of 'sharing evolution' has been adopted. This means that the SOM method will be run for a small number of iterations followed by the shape model running for a small number of iterations. This swapping of model evolution can be controlled so that a SOM runs for 2 iterations and then the snake model runs for 2 iteration and evolution

continues like this. This method gives a better fit to the data than the approach of applying the shape model after the SOM has finished as it uses the data as well as the angle rules.

Results

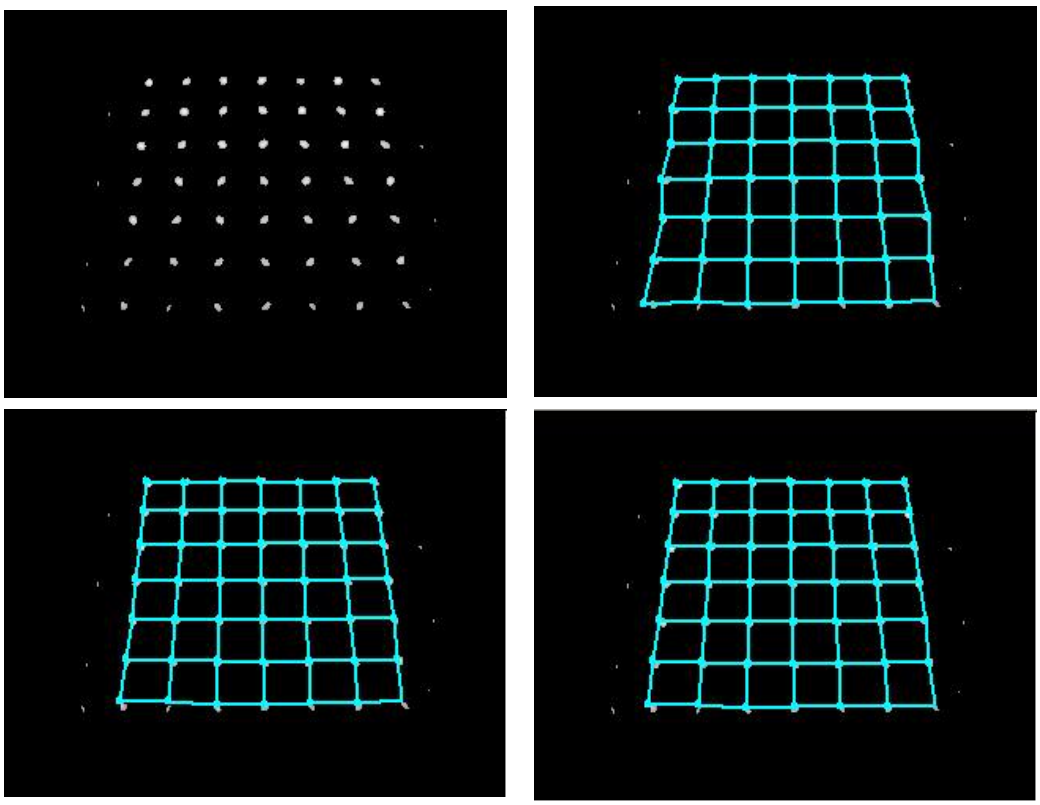


Figure 33 - Fitting of image data by shape models, neighbourhood function = 0.5, weight update amount = 0.5, momentum = 0.9996 iterations = 2500. top left: Input data, top right: SOM only, bottom left: Som run for 2500 iterations then shape model run for 2500 iterations. bottom right : Som and Shape model evolution shared 1 iteration of the SOM then 1 iteration of the shape model for 2500 iterations,

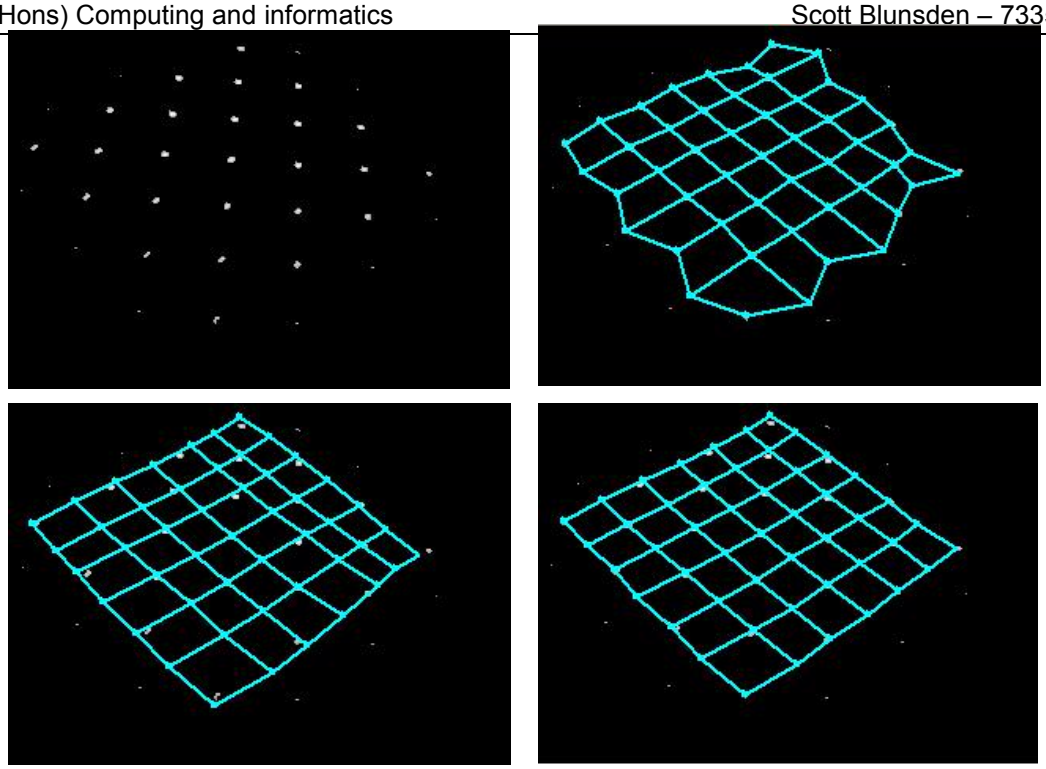


Figure 34 - Fitting of image data by shape models, neighbourhood function = 0.5, weight update amount = 0.5, momentum = 0.9996 iterations = 2500. top left: Input data, top right: SOM only, bottom left: Som run for 2500 iterations then shape model run for 2500 iterations. bottom right : Som and Shape model evolution shared 1 iteration of the SOM then 1 iteration of the shape model for 2500 iterations,

The problem of running the SOM and then applying the shape model principles to the evolved SOM is illustrated more in figure 36 than in figure 37. Both cases do provide a better fit than just using the SOM in isolation, but the interlaced approach provides a slightly better match than that of the just applying the model to the result of SOM evolution.

If the approach of running the SOM for a small number of iterations before running the shape model is to be adopted the issue of how much each model should move a node needs to be considered. The SOM network approach gradually reduces the amount that a node can be moved an how neighbouring nodes are moved, whilst the

shape model also moves nodes. One approach to solving this problem would be to reduce the amount that the shape model moves nodes over time. This is however not a good idea as during the first stages of evolution the SOM is still learning the data and trying to fit it and so letting the model control this too much may mean that the data is not learnt correctly.

The approach taken is that the snake model always moves a small constant amount and as the amount that a node is moved during SOM evolution is reduced the shape model's influence is increased. This gives the SOM enough time to learn the data and when it is adjusting itself to gain a better fit the areas where it can move to are guided by the shape model.

Growth of the net

As was mentioned in a previous section the method searches for the intersections of opposite colour squares within the chess board. Although there is nothing restricting the shape model from fitting the whole area of the board when using a filter that responds to only the inner areas of the board the model must be able to expand so that it can make a 'best guess' as to the position of the outer edge of the board. To do this the model is simply expanded by taking an edge point and moving it the same distance and direction as the square it was attached to. Although simple this method produces good results very quickly. A better method would be to work out the 3d projection of the shape model and then expand it and the use it as the final model, but this would take longer and is considerably more complex.

.

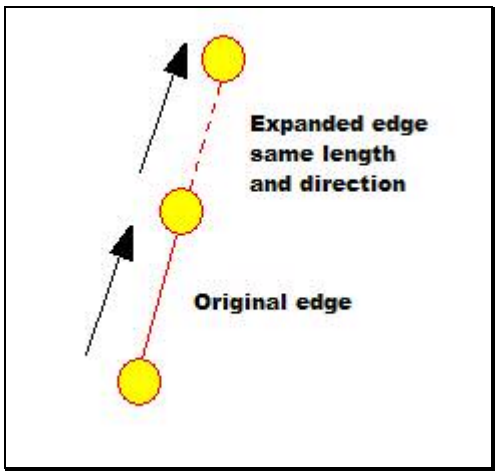


Figure 35 - Expansion of the shape model

4. Movement detection

Once the board has been located in the image it is necessary to find those board locations that are occupied by a chess piece. This will enable the system to establish the current state of play within the game.

4.1. Board Orientation

Once the location of the board has been found within the image further processing is still required to obtain its orientation so that the system can identify which square is which and what side it is playing on.

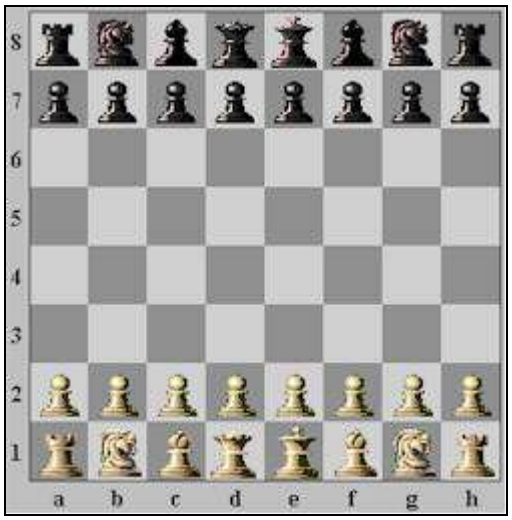


Figure 36 - Initial chess positions and identification of board positions

The aim of this stage is to match the shape models positions to those of the chess board so that the model has the same index as that shown above in Figure 36. This step is necessary to enable the computer to identify the correct positions so that it knows which squares are which. At present the shape model will match the shape from the filtered representation but will not know which square is which.

The first stage in this process is to work out which square is black and which is white. This is achieved by trying to match those squares that are nearer an idea white colour or an ideal black colour.

The shape model is converted into a set of squares that cover an area of the image. The area covered by each square is then compared to the ideal black and white pixel. The squares colour is then initially set to be the colour with the smallest difference to the ideal pixel colour.

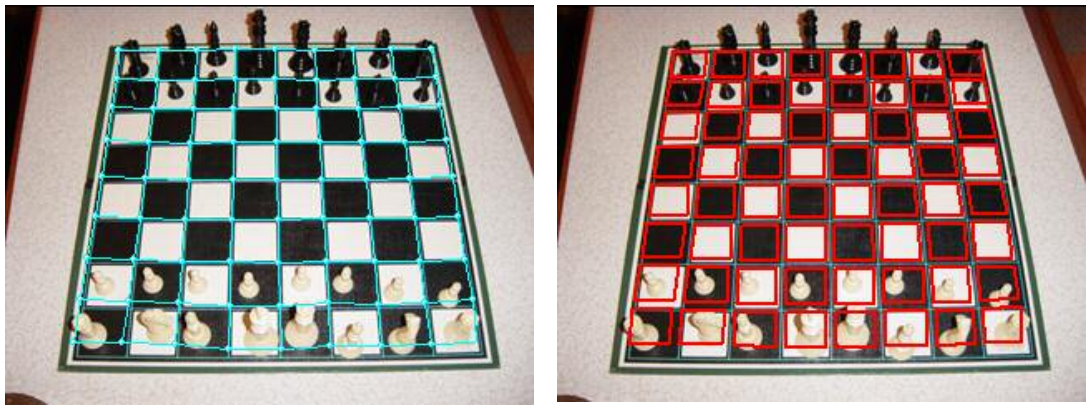


Figure 37 – Left: shape model match to the image, Right: model converted to individual squares

O

nce the initial colour has been set there may be errors where a black square with a large white piece on it would have been counted as a white square. This problem has to be rectified for the system to determine the orientation of the board in the image. A probabilistic approach is taken where it the two possibilities are taken into account. We either assume that the first square is white or it is black. The algorithm then increments itself for those which it was expecting and decrements itself for those which are incorrect. The situation with the highest score is deemed to be the starting square colour of the board.

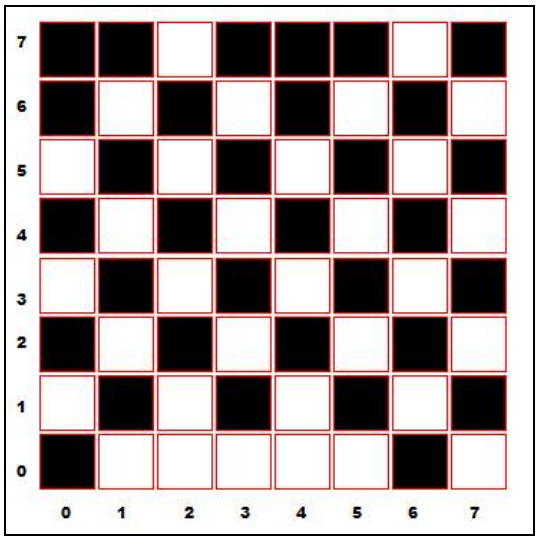


Figure 38 - Results of initial square matching - note the incorrect assignments on areas with large pieces of the opposite colour, Result of assuming first square (0,0) is white = -60; assuming first square (0,0) is black = 55. So the board starts with a black square in this case

Once the squares colour is known it the next stage is to try and find out the correct indexing of the board to the image points. Whilst at the moment we know which squares are black and white the system doesn't know about which square is a,1 b1, etc. This can only be completed if there are pieces present on the chess board as the indexing works from the white positions towards the black positions (as shown in Figure 36).

The aim of this processing is to identify both the square a1 and the square h1. Both squares have to be identified in order to establish the direction of the letters go in.

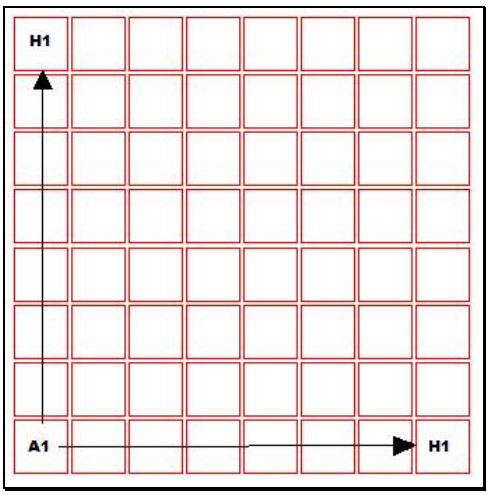


Figure 39 - possible orientations of the board for the point under consideration

Again a probabilistic approach was taken. There are two possible candidates that could be chosen as the square a1 they are the black squares in each corner of the board model. Each square is compared with the average pixel colour for black. The square with the highest difference is then chosen as square a1 as an opposite colour will respond higher than the same colour. The square h1 is then selected from the two possible candidates (as shown on Figure 39) (assuming a1 has been found) and the one with the lowest response is chosen as there should be a similar colour on similar colour. Once the squares have been found the squares that make up this model are re-ordered so that they match the indexing values of the chess board shown in Figure 36.

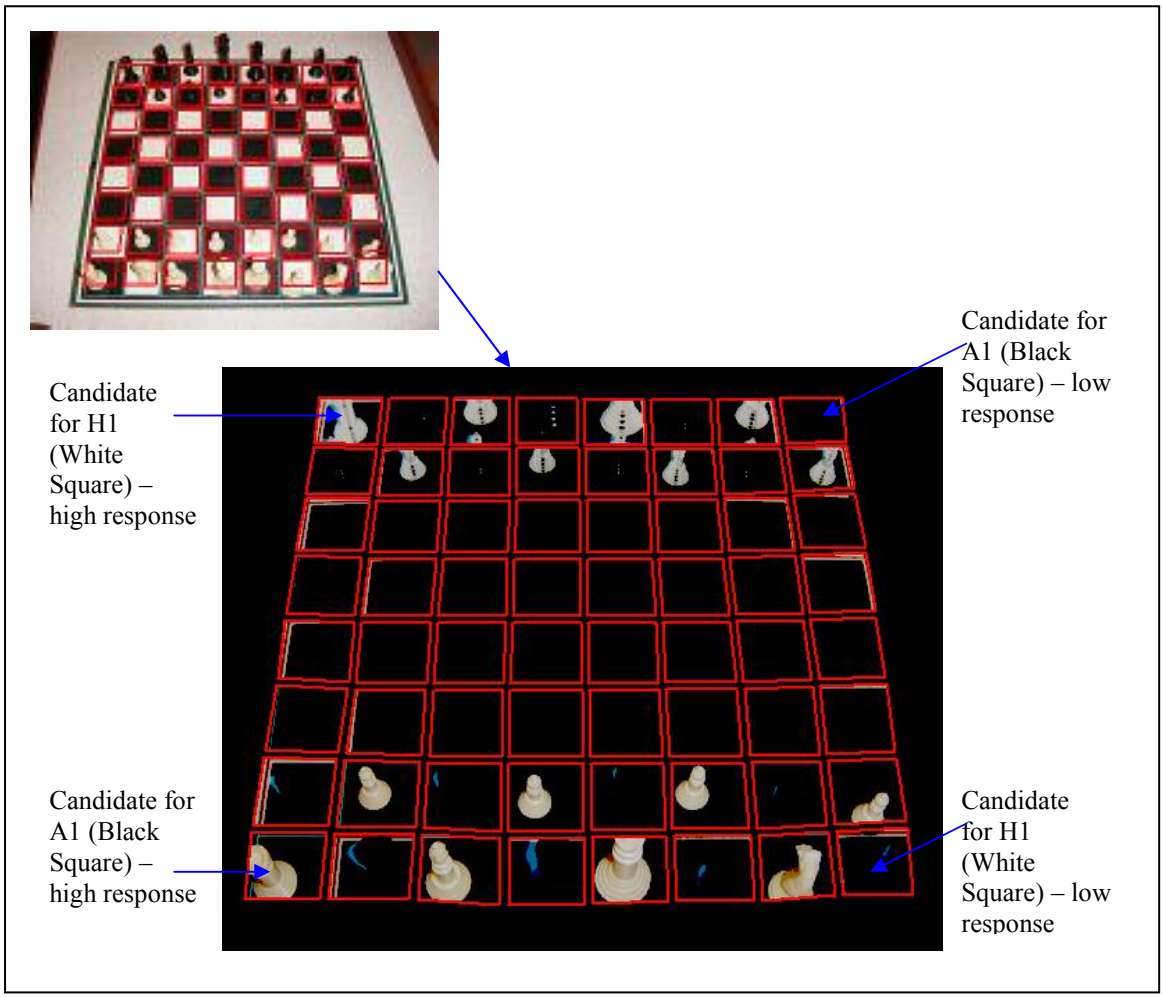


Figure 40 - Original matching areas and the results of the difference between the average pixel colours for both black and white square colours, High responses are shown brighter. Note how some occupied squares produce virtually no response, although this is advantageous in this situation it is problematic later – see next section for details.

The method although simple works well, It has been tested upon a number of different boards and works well. This is to the use of the colours that make up the image rather than comparing it to either an completely black or completely white pixel. This method does require a reasonable fit to the chess board by the model.

4.2. Occupied square

Once the squares have been identified it the system will attempt to find out which of those are occupied by the system. This is so that the system can build up a representation of the current state of the game.

Each square has an activation that is created from either the colour difference or from the SUSAN activation. Other methods could possibly be used such as comparing the textures within the square or template matching within the region. Which ever method of establishing the level of activation within a square is chosen the idea is the same, with those squares having an activation larger than the threshold are considered occupied by the system.

4.2.1. Colour Difference

First the average colour of the white square and the black square are calculated. This value is then taken from the sum of the difference between each pixel in the square defined by the model and the average pixel. This difference is the absolute value so that all values are positive. If the activation of a square is larger than the threshold value the square is considered to be occupied. An example of this process is shown in Figure 48 above.

4.2.2. SUSAN Activation

This method uses the SUSAN principle described in section 4.1.3.2 where differences between regions respond more than areas where there are small differences. This method used in preference to a simple colour difference as the average pixel colour may produce high activations when the squares are textured or some are under direct light. The SUSAN activation will perform region finding on a local scale. The colour method could be modified to cope with

these conditions but the SUSAN method already does this and is well tried and tested.

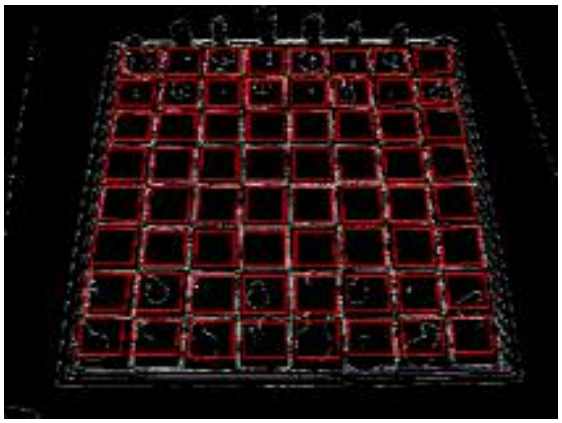


Figure 41 - SUSAN filtered image with the board model overlaid

The same principle applies however where each response (white pixel) increases the activation within the square by one. Only those squares with an activation over a certain amount are classified as occupied.

4.2.3. Results

The SUSAN method and the colour method produce similar results. They both do suffer from the same problems. The first is due to the model overlaps on the borders between the squares. This produces a large activation for that square. The solution to this problem is to not consider all of the square when determining its activation. This effectively shrinks the square so that the area around the edge of the square is not considered.

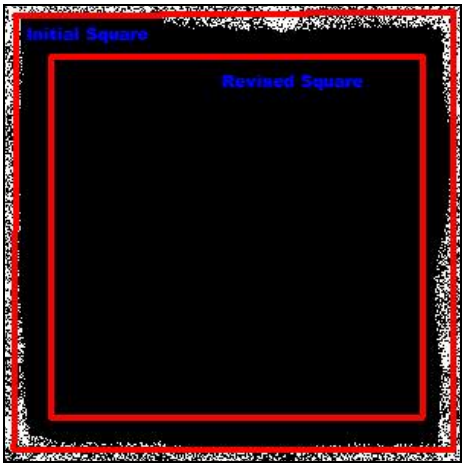


Figure 42 - Initial square position - High activation due to the high responses around the edges of the square shown in white, Revised smaller square removes the activation caused by responses between squares – the square is empty and that is now reflected by the very low activation.

The second problem is much more severe, the squares where a white square is occupied by a white piece or where a black piece occupies a black square produce a very low activation. This means that the system will think that an occupied square is not occupied. This is understandable as the black pieces on black are hard for human vision to detect. The problem for the system is that to detect the presence of a piece in this extreme case would require the threshold being set very low and so increasing the chances of the an unoccupied square being shown as occupied.

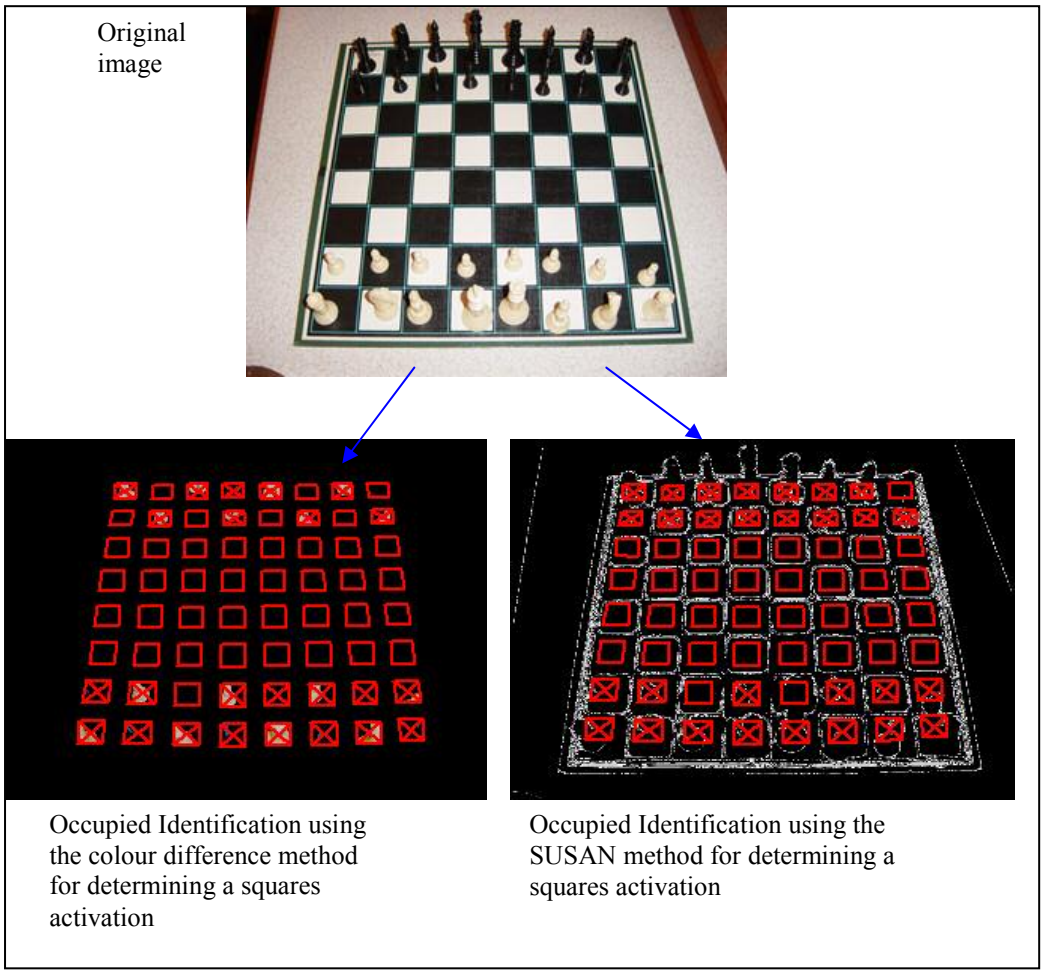


Figure 43 - Original image and matched board showing tose squares that the system has determined is occupied, where there is a very low activation the system incorrectly determines that the square is empty, The SUSAN method performs slightly better perticularrly when identifying black pieces on black squares.

4.3. Movement

The detection of a movement of a piece allows to system to keep track of the current state of the game. The system should be capable of re-finding the board if it moves within the image (or the camera moves in relation to it).

Originally it was planned to detect which square was occupied and then use this information to determine where a piece has moved to and from. This idea was then revised when the results of the board occupation method were tested. The problem of not detecting an occupied square would mean that the system would have trouble deciding upon moves. By comparing the changes in activation between subsequent images the problem of changing threshold values and dealing with only one change in occupation (eg where only one square has changed from the previous board position) are reduced.

To determine whether the board needs to be re-located the previous image is compared to the current image. If the difference between the images is significantly different the board is re-found using the shape method described in previous sections.

The system needs to determine the where the piece has moved from and to. To accomplish this, the information from the previous game state is utilised. The differences between the current activation of the square and the previous activation are calculated. This information is then sorted into a list from the square with the largest increase in activation to the square with the largest decrease in activation. A large drop in activation would be consistent with the idea that something has moved from this square and the square with the largest activation is it most likely to have had something move onto it.

As well as the simple comparison of changes in activation the rules of the chess game were used. For instance the square moved from has to have been occupied by the side who's go it is, so for example on the first go white must move. The rules are also used to determine whether a move could have been made. For example a pawn can only move in a strait line unless it is taking a piece of the opposite side. These additional constraints attempt to maximise the chance of finding the correct position moved from and to by rejecting the invalid moves. However this idea was discarded as the system should be able to cope with recognising all movement even invalid movement as it is focused upon the recognition rather than enforcing the rules of chess.

4.3.1. Results

Even when comparing subsequent board activation levels the system doesn't always produce the correct result. The situation is improved over that of working only on the occupied 'flag' that was previously suggested. The reason is similar to that stated in the previous section in that when the piece is too similar to the squares colour the activation for the square is zero or very near zero whether it is occupied or not. The benefit of checking for changes in activation rather than checking for changes in when the squares are occupied is that there is a greater chance of finding a piece if its colour is very similar to the back ground. The correct recognition rate is around 50% overall but it depends on how well the pieces stand out from the background, where they move to etc. This low rate necessitates the need for a method to correct the board positions. A sample game sequence is shown in the appendices.

5. Critical Evaluation

5.1. Project Management

A project consisting of a large amount of investigation was hard to plan for using traditional methods. Originally the amount of time that was allocated to each stage of the project was based upon a fairly even split of the time available for the project with the most time being allocated to design and development. Originally these sections ran consecutively one after the other. Within the first few weeks of the project it was obvious that a more iterative approach had to be used. This was primarily due to the project managers inexperience of how certain image operations behave when they are implemented.

The investigating of image processing methods and then the implementation of them for evaluation meant that the system was being built as the project progressed. This made the development of the system difficult and it took longer than expected mainly because of the project managers inexperience with computer vision applications and the amount of investigation and research necessary before design or development could begin. This meant that many contingencies had to be considered such as not meeting the original objectives or narrowing the scope of the project.

5.2. Investigation and Design

The investigation and designing of the system took up most of the time that was allocated to the project. The project had a high research content involved where there was no pre-determined way to design the system. The results of the investigation were fed back into the design of the system in an iterative way.

The approach involved researching a way to accomplishing a specific stage of the system. As an example the Hough transform was deemed to be a good way to locate the edges of the board so that the board can be found within an image. The specific details and issues of this approach are detailed in section 3.1.1.1 but it was deemed necessary to implement this image processor so that a conclusion can be reached about whether to use it in the final system. In this case it was not used within the final system.

During the early stages of the project many different methods were tried and implemented and then subsequently dropped from the design of the system this took up lots of the development time of the project. This investigation sometimes went into too much detail, an example would be that three different methods of edge detection were implemented and tested when only one was really needed. This situation occurred less frequently as the project progressed and the project manager gathered more experience in image processing features.

The design of the system also evolved over the lifetime of the project. This was intentional but did mean that lots of work was completed and not used the ideas behind it were then adapted so that they could be used. An example of this is the snake where the original snake implementation is not used and a significant amount of time was spent on it. The snake idea of fitting a flexible model to processed image was adapted to the final

shape model. This investigation and adaptation of the design meant that project management aspects were hard to manage. For instance at one stage in the possibility of not finding a solution to the problem was a major concern and the contingency was planned for.

One of the major problems of the investigation and design stages was how far to go. Although the project aims were clearly defined it was still possible to know when a objective had been met. As an example consider the objective of *'Recognize the boundaries of the chess board and the individual positions on the board'*. This could be extended to allow the computer to recognise the objects presented in an image before deciding if a chess board was present. This would then require an object recognition component to the system which is another project in itself.

5.3. Development

5.3.1. Development environment

Visual C++ was used as the development environment, the reasons for this are described in the Appendix B for the image processing aspects of the project the system proved very easy to use and interfaced with DirectX extremely well. The Visual SDK provided easy to use and powerful classes to interface with both static and live images.

The project manager had never developed anything significant in Visual C++ and this meant that there was a steep learning curve whilst developing the project. Although it is easy to blame a third party for problems the system did hinder the development of the project at certain stages. The user interface features supplied with it are very basic and hard to use, especially compared with something like Borland's Delphi and meant that user interface design had to be revised several times. The system also

occasionally deleted files from the computer system and meant that the program had to be restored from a backup. The system occasionally crashed when the program was compiling and although annoying did not damage the program.

If it was possible and there was time another development environment would have been used in preference to this one although one suspects that the integration with the Vision SDK would have required more work.

5.3.2. Image processing

The development side of the project was relatively easy as once the investigation and design of a feature had been completed it was usually a matter of converting the design to code. Most design was based upon tried and tested methods in image processing and where it was not it was usually a combination of methods.

The project was developed using classes where each set of operations were usually groped together so for instance an edge detection class exists and deals with all the various methods of edge detection (sobel, canny). The image processing aspects were usually strait forward in development. However many of the methods were described in research papers and many were described in purely mathematical terms. This meant that a lot of time was spent upon implementing the methods in code so that they could be evaluated. This situation was not ideal and instead the project manager should have attempted to find out results and pre-written implementations in order to evaluate methods before deciding whether to include them in the system.

The final system and the rotational filter were unique to the project (to the best of my knowledge) and so these were more difficult to develop and there was a great degree of uncertainty when designing and developing them.

5.4. Implementation of final system

5.4.1. Design

The overall design of the recognition aspect of the program was constantly evolving. The testing of each feature such as the snake or the shape model changed the design again or at least modified it. This continual changing has led to certain inconsistencies in design (such as differing style of classes).

The system grew up around the research and development of various image processing techniques rather than being designed in a traditional way. This initially led to problems such as the values being displayed by the user interface program being out of step with those values used in the program. A more formal design would have eliminated this problem and saved time in fixing it. The prototyping tools that are supplied by visual C++ were

5.4.2. Functionality

The functionality of the project fulfilled all the objectives of the project. However it is perhaps lacking a portion in dealing with live images in that the system should be able to tell when a move has been made rather than being told by the pressing of a button. This feature is suggested as a future enhancement but it could have perhaps been included into this system.

5.4.3. Reliability

When testing an image system it is very hard to say that the system works under all conditions that it could be presented with. There are lots of situations that the human visual system copes with effortlessly that are extremely hard to cope with in a computer program. The system has been tested with various boards, rotations of boards, lighting conditions and different cameras. However even extensive testing could not guarantee that that system could cope with any situation.

5.4.3.1. Recognition

It has been shown in section 4.1.3.3.1 that the size and orientation of the feature filters have a significant impact upon the quality of the output. Unfortunately only certain situations are accurately coped with and although the system will try to work with what it is presented its performance in board localisation is very dependent upon such things as size of filters as they provide the input to the shape model. It would be much better if the system could automatically work out the best feature detectors to use.

The method that finds the orientation of the chess board is reliable when presented with the starting position of the board and then updated with only small changes to the image. For instance if the board is rotated the system needs to keep track of this. If for example the board is presented to the system half way through a game it would not be possible to accurately index the squares. This is a problem in general that doesn't have a solution. If one was presented with a game half way through it would not be possible to tell which square was a1, a2 etc unless the observer was told it would not be possible to tell which square was which. This obviously creates a problem for the system in that it cannot represent the game state.

5.4.3.2. Movement Detection

Without doubt the most unreliable aspect of the whole system is the movement detection with rates as low as 50% although partially correct answers are more frequent. The reasons for this are detailed in the relevant sections but these issues still impair the systems performance.

It is very hard to find out how to correct this problem as if two colours are very similar then it is hard to differentiate between an empty square and a similar colour on the same square.

This problem is further emphasised by the fact that if the system is not corrected then the game state that is shown by the system can become wildly different to what is happening.

6. Future work

Rather than answering all the questions surrounding this project more have been raised throughout its duration. There is a wide range of possibilities for future work within this area.

6.1. Recognising individual chess pieces

In its current state the system is essentially 'blind' to the individual pieces that occupy the chess board. A better system would recognise what was occupying a square rather than just identifying that it was occupied or had changed.

6.2. Robot controller

The addition of a robot controller that would allow the computer to physically move the pieces and so play against a human opponent. The system currently provides a matrix that has been used to project the image into a plan view. This could be extended to provide coordinates that a robot arm could use to guide itself to the correct position to move a piece. The computer would also have to know how to play chess is it was to play against a human opponent.

6.3. Board Tracking

Once the model has been matched to the board it should keep evolving so that it can keep track of where it is moved to. This could be accomplished by keeping some movement in the shape model after it has been initially matched. The system would constantly update the system so that it can track of the board if it was moved within the image.

6.4. Other board games

The system could be extended so that it can deal with for example draughts by swapping the game representation aspect of the system.

6.5. Extension of the rotation filter Method

The ideas proposed by the rotation filter are very promising and would solve some of the rotation problems that are illustrated when using the matrix approach. Although the rotation filter did not perform as well as hoped further development of the method could provide a version that works more reliably.

6.6. Evolution of the shape model

The idea of the shape model itself could be extended to match other objects. The angles between nodes can be changed within the current system but the idea could be further extended by allowing nodes to be drawn towards different image features (eg lines, corners or different colours). A distance measure could also be used to keep the nodes closer together or further apart.

7. Conclusion

The original goal of the project was to

> 'To create a system that is capable of turning a digital video representation of a game of chess into an internal representation that the computer can use to play chess. The system will thereby acquire and process the digital images.'

To a large degree this goal has been met, the system is capable of acquiring images, either live or static images and processing them to identify the position of the chess board within the image. The System does work and the project manager is pleased that it was able to reach a stage where it could be used in this way.

The system can be used to play chess but with the limitations of the system not always producing displaying the correct move, although there is a provision to correct the system.

During the life of this project one gained an appreciation of the human vision system and its workings, how ever much you think you know about it you really have no idea.

8. References

[A. Lanitis, C.J. Taylor, T.F.Cootes, T.Ahmed, 1995] A. Lanitis, C.J. Taylor, T.F.Cootes, T.Ahmed, 1995, Automatic Interpretation of Human Faces and Hand Gestures Using Flexible Models. Proceedings of the International Workshop on Automatic Face- and Gesture recognition. Zurich

[A.Johnson et al. 1995] A. Johnson, P. Leger, R. Hoffman, M. Hebert, and J. Osborn, 1995, 3-D object modeling and recognition for tele-robotic manipulation
Proc. IEEE Intelligent Robots and Systems, Vol. 1, pp. 103 - 110.

[A. Witkin et al, 1987] A. Witkin, D. Terzopoulos. M. Kass., 1987, Signal Matching through scale space, International Journal of Computer Vision, 1(2):133-144

[B. Mel, 1997] B. Mel, 1997, Seemore: Combining color, shape, and texture histogramming in a neurally inspired approach to visual object recognition. Neural Computation 9:777-804

[B. Scheilde, 1997] B. Scheilde, 1997, Object Representation using Multidimensional Receptive Field Histograms. PhD thesis I.N.P Grenoble. English Translation

[B. Scheilde, J. Crowley, 2000] B. Scheilde, J. Crowley, 2000, Recognition without Correspondence using Multidimensional Receptive Field Histograms. International Journal on Computer Vision, 36(1):31--50

[B .Caputo. et al, 2001] B. Caputo, D. Paulus, S. Bouattor, 2001, A novel probabilistic Model for 3D Object recognition: Spin-Glass Markov Random Fields. VMV

[C.J. Taylor, G.J. Edwards, T.F. Cootes, 1998] C.J. Taylor, G.J. Edwards, T.F. Cootes, 1998, Active Appearance Models, Proc. European Conference on Computer Vision 1998, Volume 2, pp484-489, Springer

[D. D. Gomez, M. B. Stegmann, 2002] D. D. Gomez, M. B. Stegmann, 2002, A Brief Introduction to Statistical Shape Analysis, pp. 15, Informatics and Mathematical Modeling, Technical University of Denmark, DTU

[D. Terzopoulos et al, 1987] D. Terzopoulos, A. Witkin, M. Kass, 1987, Symmetry seeking models for 3D object reconstruction, In First International Conference on Computer Vision, London, England, pages 269-276, IEEE, Piscataway, NJ

[E. Rolls, M. Elliffe, S. Stringer, 2002] E. Rolls, M. Elliffe, S. Stringer, 2002, Invariant recognition of feature combinations in the visual system, Biological Cybernetics 86, pp59-71

[E.T Rolls, 1992]E.T.Rolls, 1992, Neuropsychological mechanisms underlying face processing within and beyond the temporal cortical areas. Philosophical Tranactions of the Royal society, London [B], 335, 11-21

[G. Wallis, E. Rolls, 1996]. G. Wallis, E. Rolls, 1996, A model of invariant object recognition in the visual system. Progress in Neurobiology, submitted for review

[MINOLTA, 2003] MINOLTA, 2003, 3D Digitiser, (online) http://www.3dscanner.ch/ , date accessed 01/02/2003

[M.J. Kochenderfer, J.G. Nichol, J.L. Jacobs, 2002] M.J. Kochenderfer, J.G. Nichol, J.L. Jacobs, 2002, Visual Chess Project, [online] http://www.stanford.edu/%7Emykel/chess/, date accessed 10th, December, 2003

[M. Kass et al, 1987] M. Kass, A. Witkin, D. Terzopoulos, 1987, Snakes: Active contour models, International Journal of Computer Vision, 1(4):321-331

[M. Nixon, A. Aguuado a, 2002] M. Nixon,2002, A. Aguuado, Feature Extraction and Image Processing, Great Britain, Newness, 1st edition , pp199-206

[M. Nixon, A. Aguuado b, 2002] M. Nixon, A. Aguuado, 2002, Feature Extraction and Image Processing, Great Britain, Newness, 1st edition, pp206-213

[M. Nixon, A. Aguuado c, 2002] M. Nixon, A. Aguuado, 2002, Feature Extraction and Image Processing, Great Britain, Newness, 1st edition, pp 218

[M. Nixon, A. Aguuado d, 2002] M. Nixon, A. Aguuado, 2002, Feature Extraction and Image Processing, Great Britain, Newness, 1st edition, pp 227

[M.Sonka, V. Hlavac, R.Boyle a, 1999] M.Sonka, V. Hlavac, R.Boyle, 1999, Image Processing, Analysis and Machine Vision, 2nd Edition, Brookes Cole Publishing Company, USA, pp 448-483

[M.Sonka, V. Hlavac, R.Boyle b, 1999] M.Sonka, V. Hlavac, R.Boyle, 1999, Image Processing, Analysis and Machine Vision, 2nd Edition, Brookes Cole Publishing Company, USA, pp 511-520

[M. Swain, D. Ballard, 1991] M. Swain, D. Ballard, 1991, Color Indexing, International Journal of Computer Vision, 7(1):11-32

[P. Min et all, 2003] P. Min, J. A. Halderman, M. Kazhdan, T.A. Funkhouser 2003
Early Experiences with a 3D Model Search Engine, to appear in Proc. Web3D Symposium, Saint Malo, France

[Princeton Model Search Engine, 2003] Princeton Model Search Engine, 2003, (online)http://shape.cs.princeton.edu/search.html., date accessed 01/02/2003

[S. Brunner, E. Pohn, 2003] S. Brunner, E. Pohn, 2003, ASP5-Teilprojekte zum
“Turkish Chess-Player”, Imagination Computer Services GmbH, (German Language Version)

[S.M. Smith, 1995] SUSAN -- A New Approach to Low Level Image Processing Technical Report TR95SMS1c,
Oxford Centre for Functional Magnetic Resonance Imaging of the Brain (FMRIB), Department of Clinical Neurology, Oxford University, Oxford, UK

[T. Kyriacou, G. Bugmann and S. Lauria, 2002] Theocharis Kyriacou, Guido Bugmann, Stanislao Lauria , 2002, Vision-Based Urban Navigation Procedures for Verbally Instructed Robots, Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS'02) EPFL, Lausanne, Switzerland, pp.1326-1331

[TRACLabs, 2003,] TRACLABS, 2003, Biclops Brochure, (online) http://www.traclabs.com/BiclopsBrochure.pdf, date accessed 01/02/2003

[T, Kohonen, (1982)] T, Kohonen, Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 1982, 59–69.

[Kohonen, T, 1997] Kohonen, T., Self-Organizating Maps, New York : Springer-Verlag, 1997.

[Microsoft Vision Research Home Page, 2003] Microsoft Corporation, 2003, (online) http://www.research.microsoft.com/research/vision/ date accessed 30/04/03

[Direct X download site, 2003] Direct X Download Site, 2003, (online) http://msdn.microsoft.com/library/default.asp?url=/downloads/list/directx.asp date accessed 30/04/03

Appendices

Appendix A - User Guide

1. Selecting an Image source

If there are multiple image sources available on the computer then the system will ask which one you want to use within the program. This screen is also presented when the program is unable to find a camera input. If no camera input is available the press cancel on the screen to indicate that the program should not attempt to look for one. If there is one image source available the user will not be prompted to select one and the program will automatically use the available image source. The screen is shown in Screen 1.

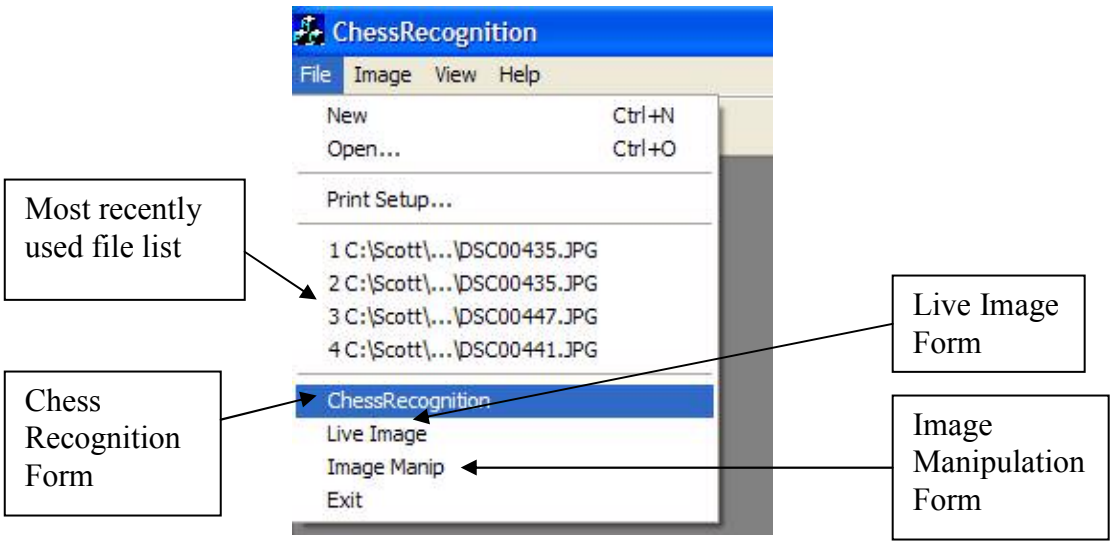
Screen 1 - Selecting an image source provider

2. Selecting the form to use

There are three types of form to select from within the program, they are:

- Chess Recognition Form
- Image Operations Form
- Live Image Form

The type of form required can be selected from the file menu of the main form by selecting then like so:



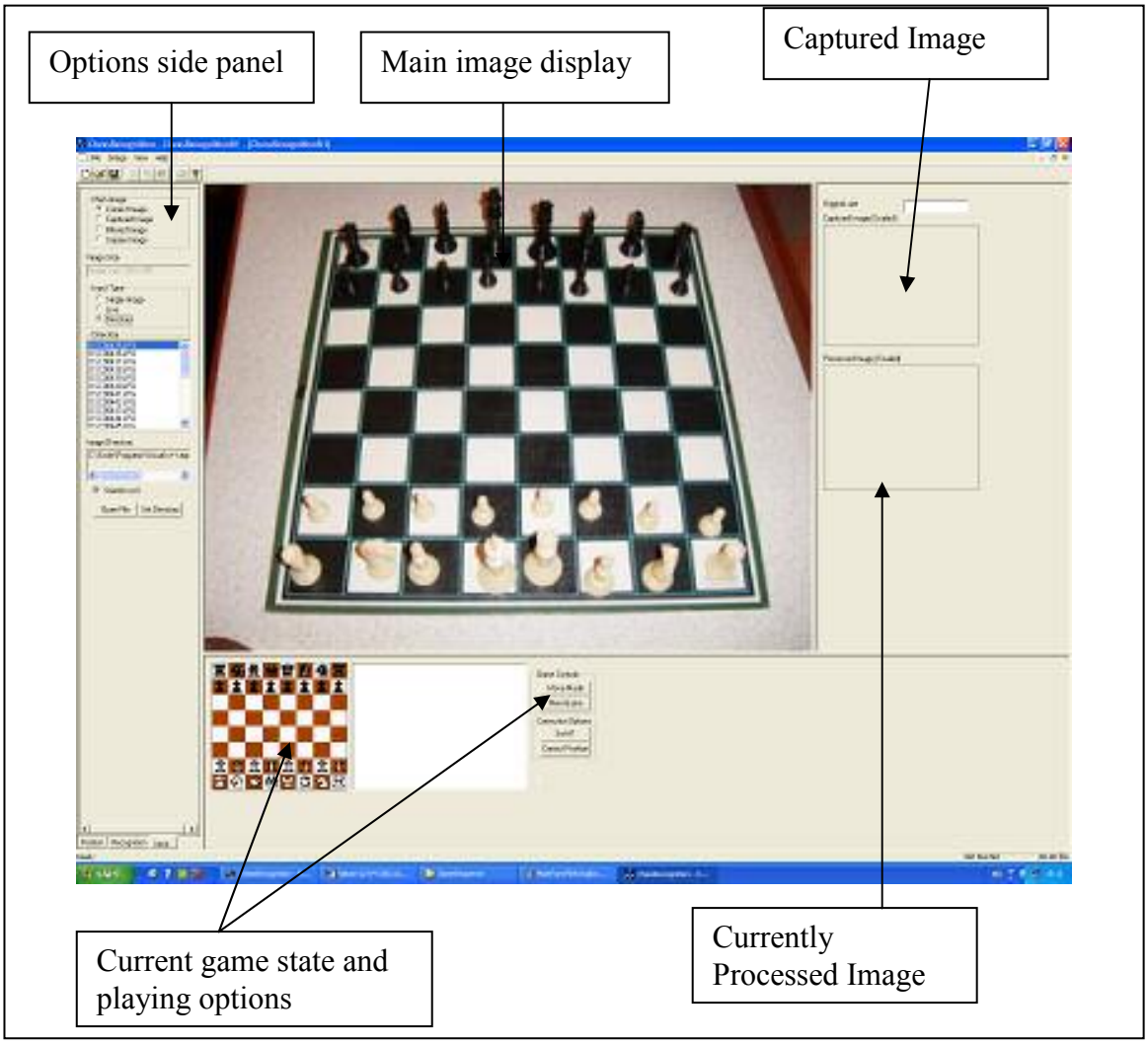
Screen 2 - File menu options

If a file from the most recently used list is selected the system opens up a the image processing options form.

The forms and how to use them are described in the following chapters.

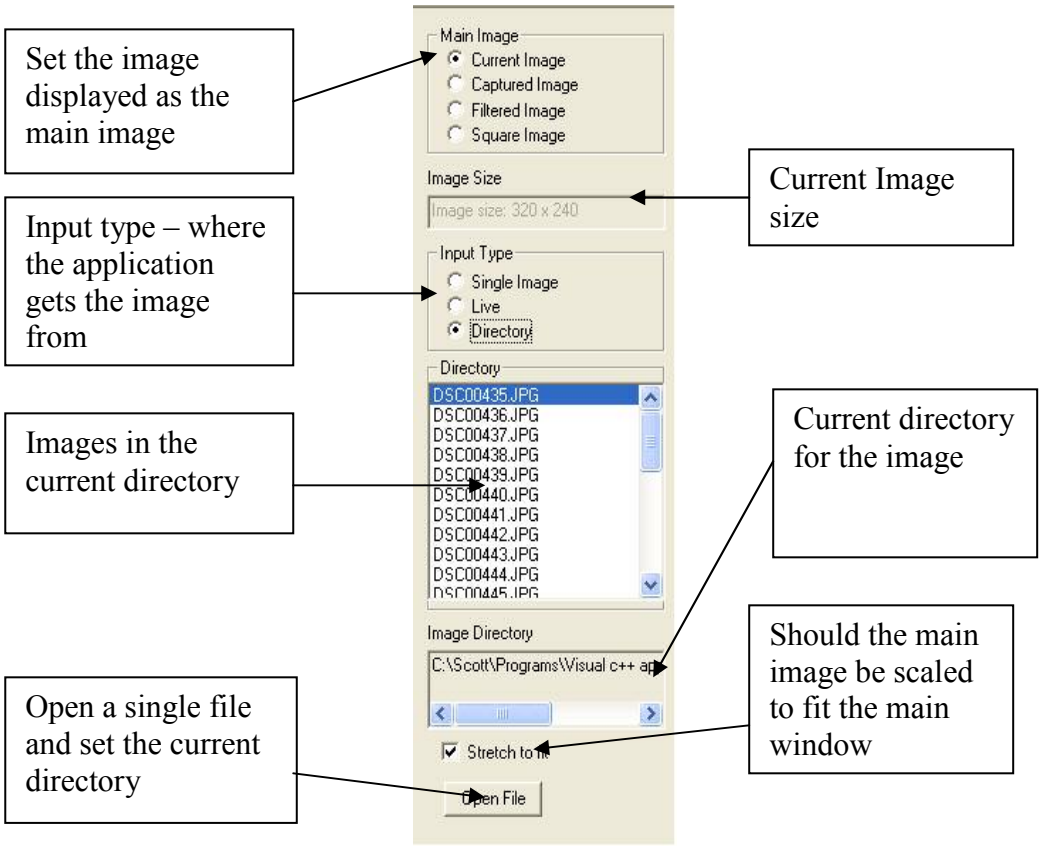
3. Recognition Form

The recognition form is the main part of the system and allows the system to convert images into a computer readable form. The main screen displays the current image in the centre the captured image that the system is working from and processed images. A list of available options are displayed down the side of the form in a tab options form. These role of these parameters are discussed in the main body of the report. The main screen is shown below.

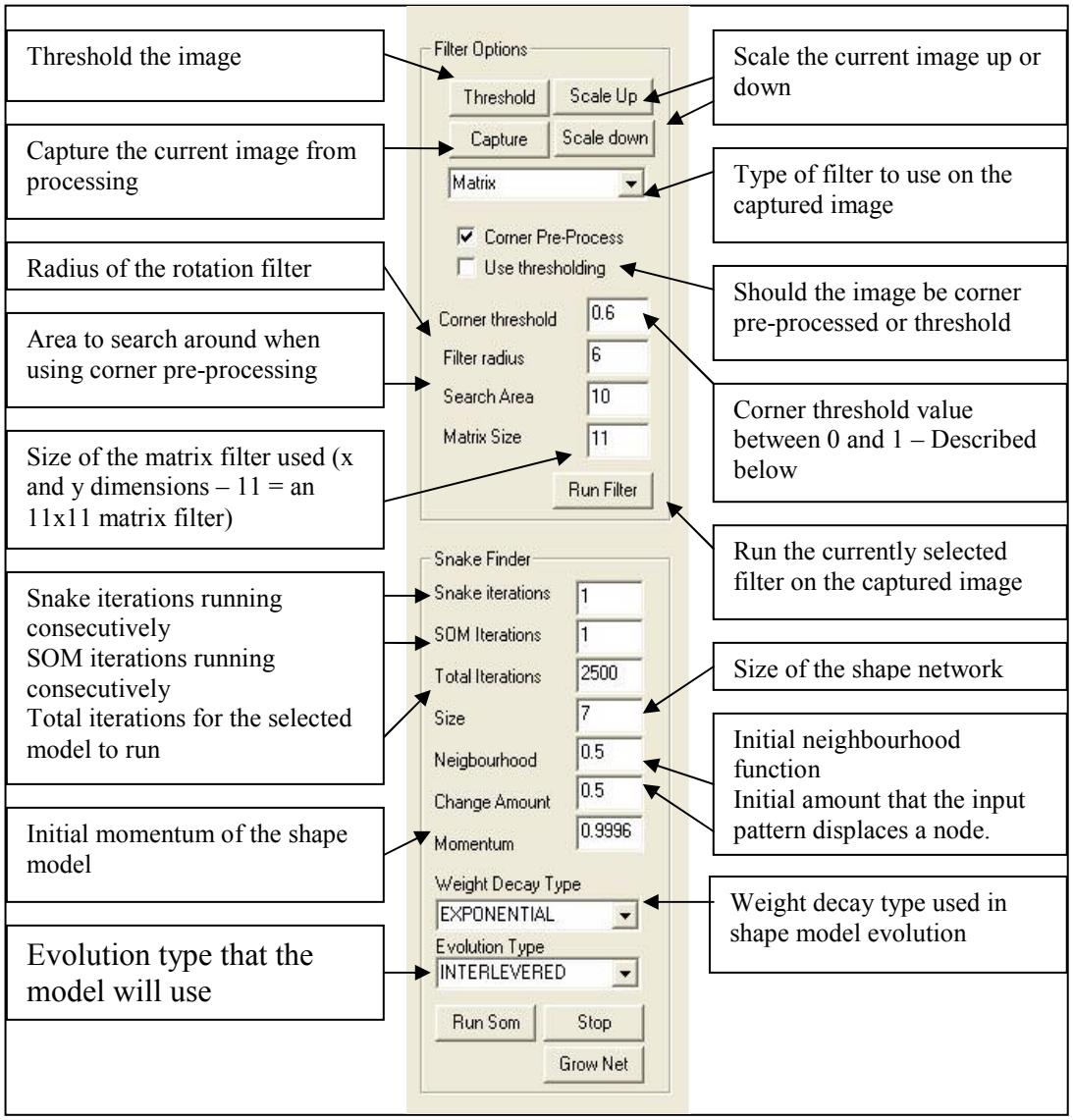


Screen 3 - Chess recognition screen

3.1. Image Input Options



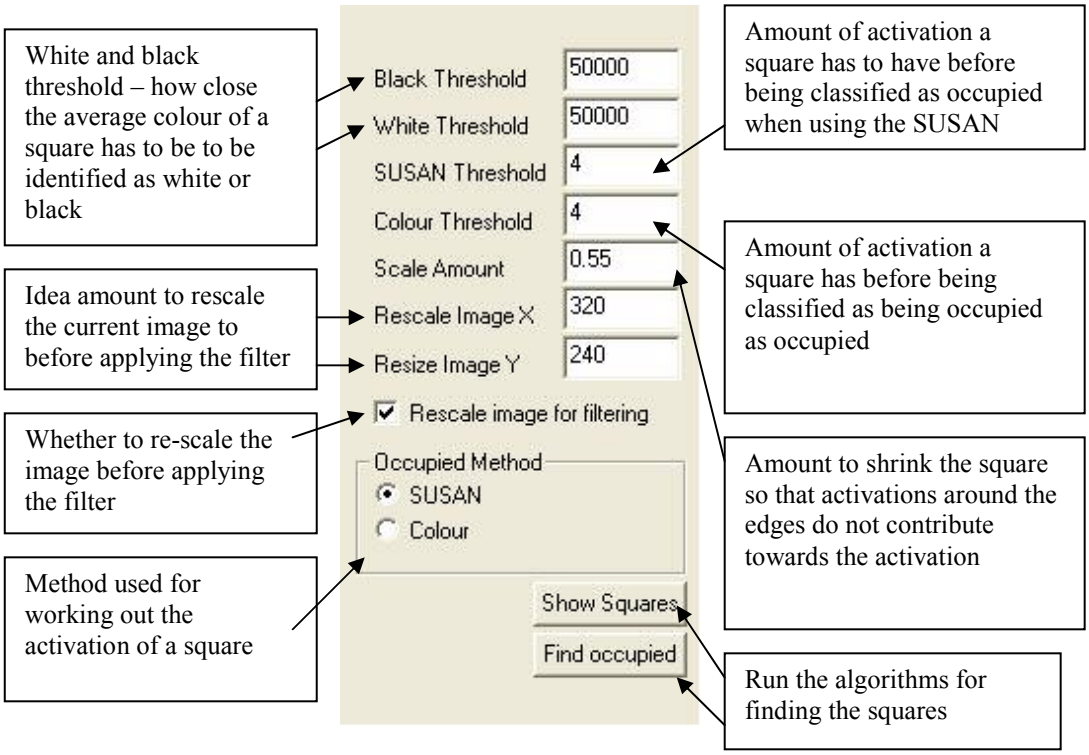
3.2. Recognition Options



Screen 4 - Recognition Options Side Panel

The corner threshold removes the bottom x amount of responses when using the matrix filter. For example 0.6 will only place those responses that are in the top 40% of the response matrix into the filtered image.

3.3. Board positions Options



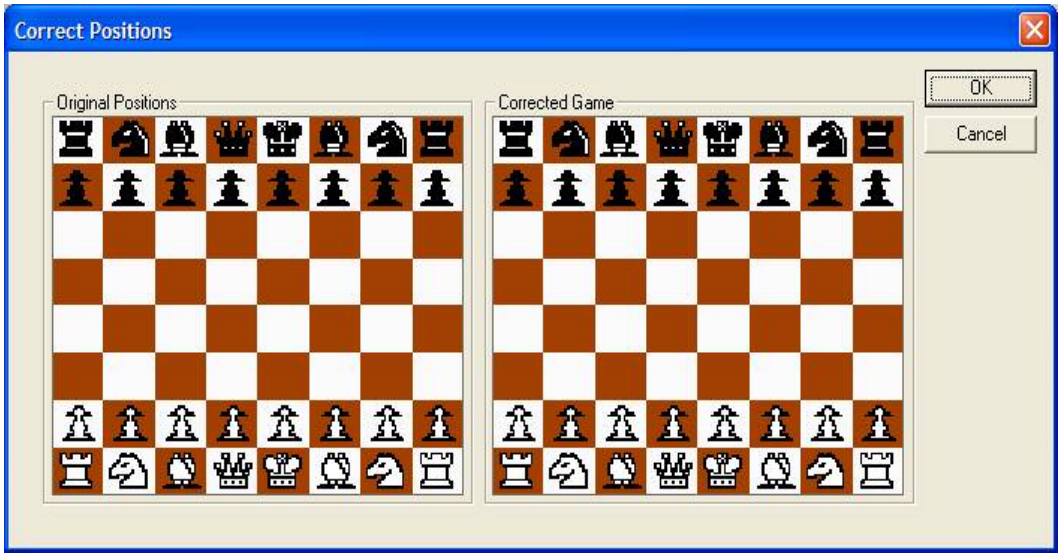
3.4. Playing the game

Once an image has been displayed in the main image screen the new game button should be pressed and the system will automatically process the image and attempt to find the board and index the squares. The system will then display what it is doing on the screen. There are three main stages that are displayed by the user, they are initial filtering of the image, matching of the shape model to the filtered image and matching indexing of the squares.

Matching of the Shape model to the data

Indexing of the chess board

Once a new image has been selected either by the live camera input or by loading a image from file the made move button should be pressed, the system re-finds the board and displays where it thinks a piece has moved. Should the system get it wrong there is an option to correct the system. Pressing the button labelled ‘correct position’ will bring up a screen illustrating where the previous position on the right and the current position on the left. The user can drag a piece into the correct position so that the computer corrects itself to confirm the move the user should click on ‘ok’ or if they wish to cancel the action the cancel button should be pressed.



Screen 5 - Correct game state screen

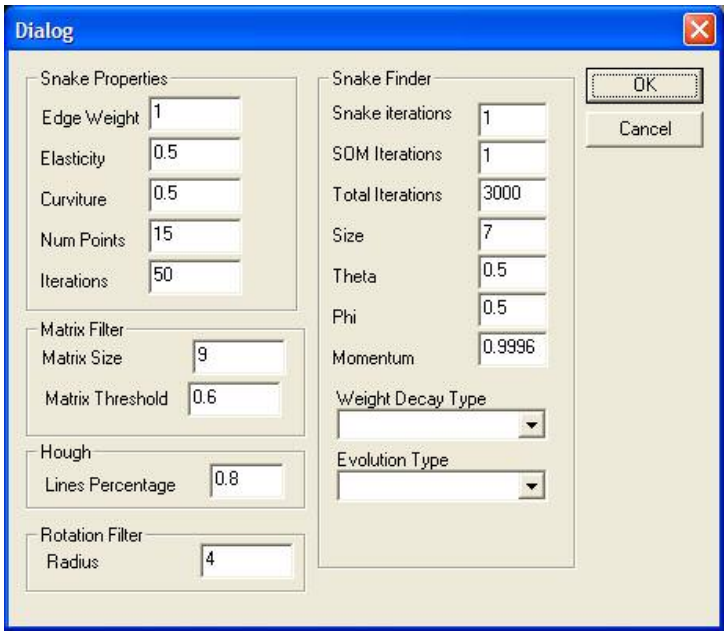
4. Image processing Form

4.1. Menu Options

Once an image has been selected and is displayed within the form a menu option is presented. To perform the option select it from the menu and the image will be processed and the result will be displayed in a separate form. There is a special case for the snake where it is possible to define where the snake should start from. To define this box click and drag within the image and a rectangle will be displayed that represents the starting position of the snake.

4.2. Image Processing Variables

All of the image operations that are available are controlled by a set of options. To set these options select Options from the main menu and the select Set Options from the drop down menu.



Screen 6 - image processing options

Appendix B - Implementation Details

The software was developed using Microsoft Visual C++ version 6. The project uses the Direct X software development kit to provide camera input to for the system. Visual C++ was chosen for its speed and close integration with Direct X. The Visual SDK is also only compatible with the C++ programming language. C++ provides the a high speed implementation of the image system.

Direct X is a set of low level application programming interfaces that provide access to multimedia features within windows. The system provides one way to access these multimedia features, in this particular case the interface with imaging equipment is what is utilised. Most windows machines come Direct X installed and although a download is required to develop in it, no download is required for the program to run. The advantage of this is that many image capture devices can be used by the system with no modification of the program or no re-compilation. The version SDK version used within this project is version 8.1 and is available from [Direct X download site, 2003].

The system uses the vision SDK for loading saving and manipulating images. The reason for this was mainly due to the projects focus on the processing of the images rather than the numerous technical details that would be involved if routines had to be created before using images within the program. The visual SDK is a development kit that provides a C++ classes that enable easy access to images. The visual SDK is downloadable from [Microsoft Vision Research Home Page, 2003] and provides details of how to install and configure the development kit (a copy is provided on the disk).

Virtualy no modification was required to the Vision SDK however it was necessary to add a method within the a base class of so that images could be scaled to fit within a rectangular area. The method display in DisplayInHdcStretch was added in the file visual base.ini. A copy of the modified version is included on the accompanying disk. This should be copied into the file or the default version should be replaced by the modified version of the file.

```
inline bool CVisImageBase::DisplayInHdcStretch(HDC hdc, const RECT&
rectDest) const
{
        return DisplayInHdc(hdc, Rect(),
                        rectDest, SRCCOPY,
                        0, false);
}
```

Architecture

The previous sections described the various operations that are carried out upon an acquired image in order to locate a board within an image scene and determine the state of the game. The steps involved in this process are displayed below. Each processing step provides the input for the next section. The outline of this process is shown in Figure 44 below. In previous sections various methods were explored and compared, within the final system a stage processes either an image or a model of that image and passes it along to the next stage. Within the final system it is possible to compare different combinations at each stage. For example either the matrix filter or the rotation filter can be used as input for the shape model. How to do this in the final application is described in users guide in Appendix A, but this idea allows a user to compare the various methods for themselves. This design also makes it possible to easily insert new filters or shape models within the system.

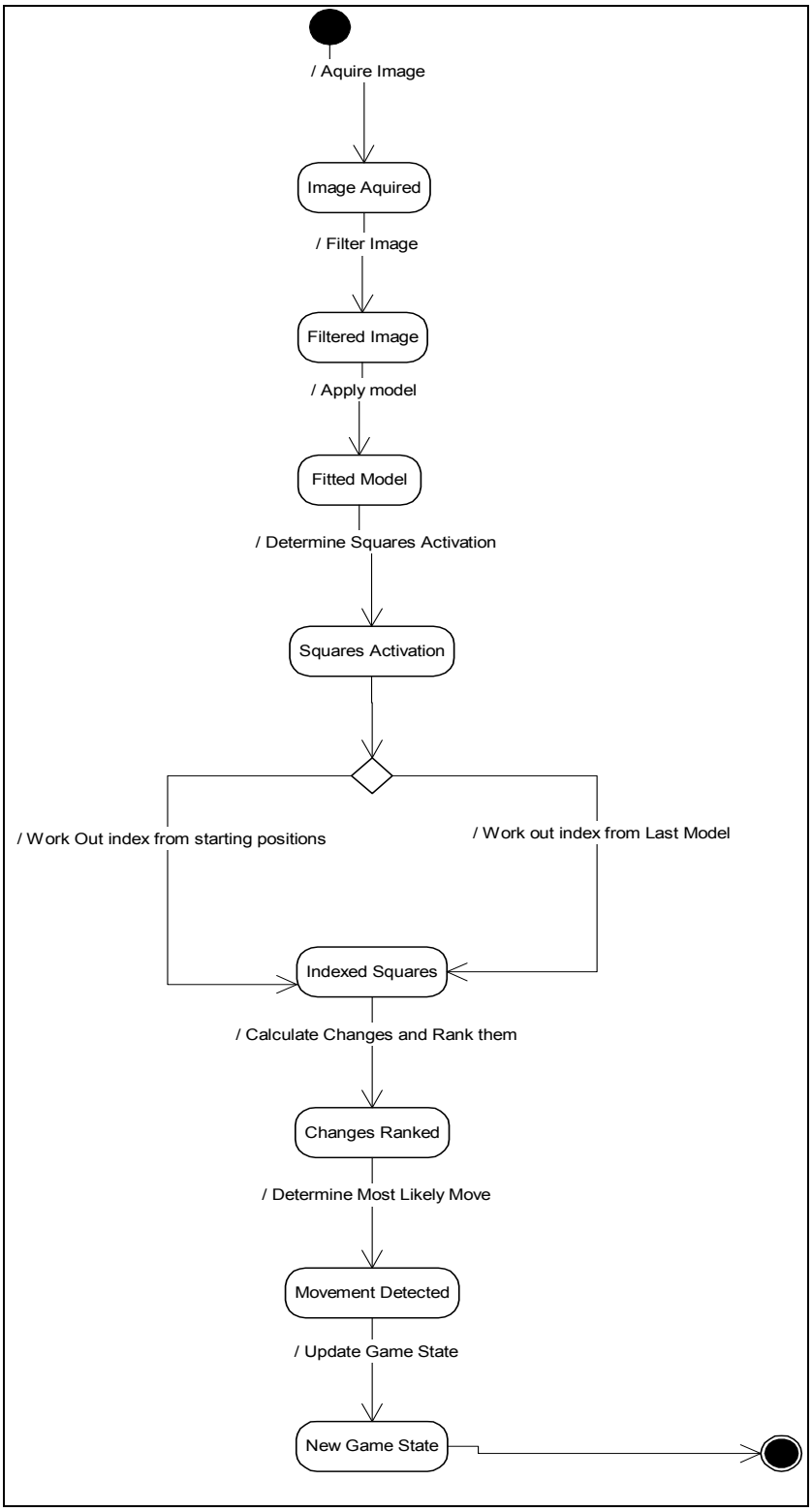


Figure 44 - Processing steps that the system performs

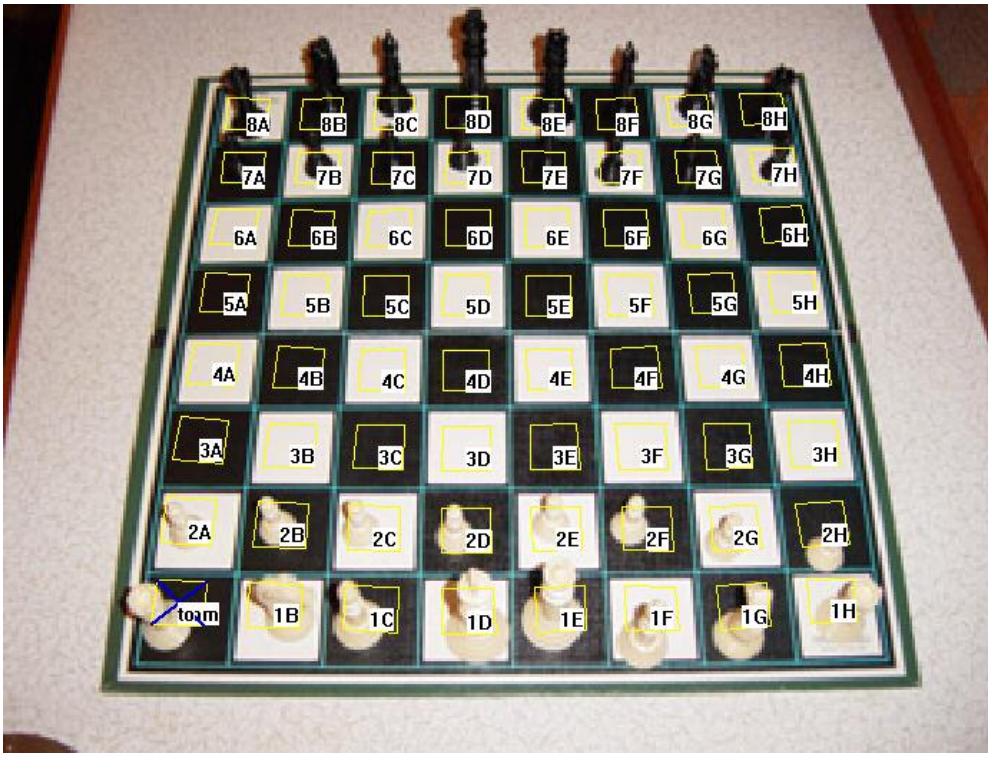
Software

The languages used and the software architecture is described in the appendix. The system runs on a Windows machine with DirectX installed and takes camera input from a DirectX compatible camera. More implementation details are provided in the Appendix B.

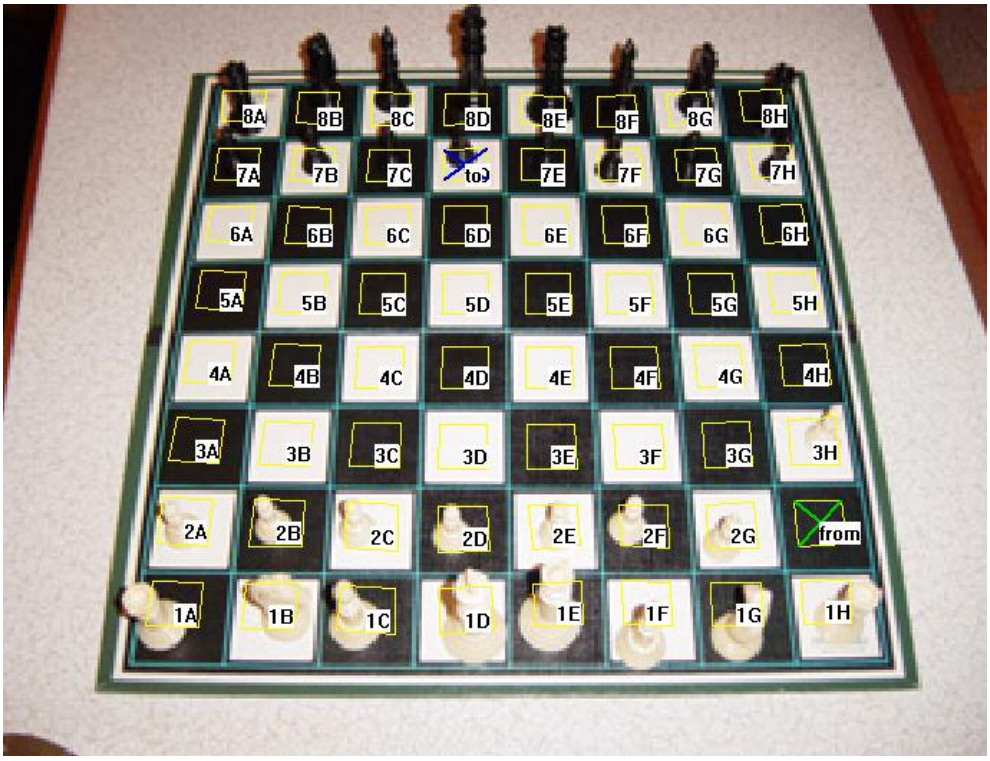
System

The final system is the bringing together of previously independent aspects of the system. The final system performs well when finding the board within an image. The matrix filter performs significantly better than the rotation filter as much stronger responses are given at the intersection of four squares. Also the system provides the facility to re-scale the current image before it is filtered to reduce the time that is taken to perform the recognition.

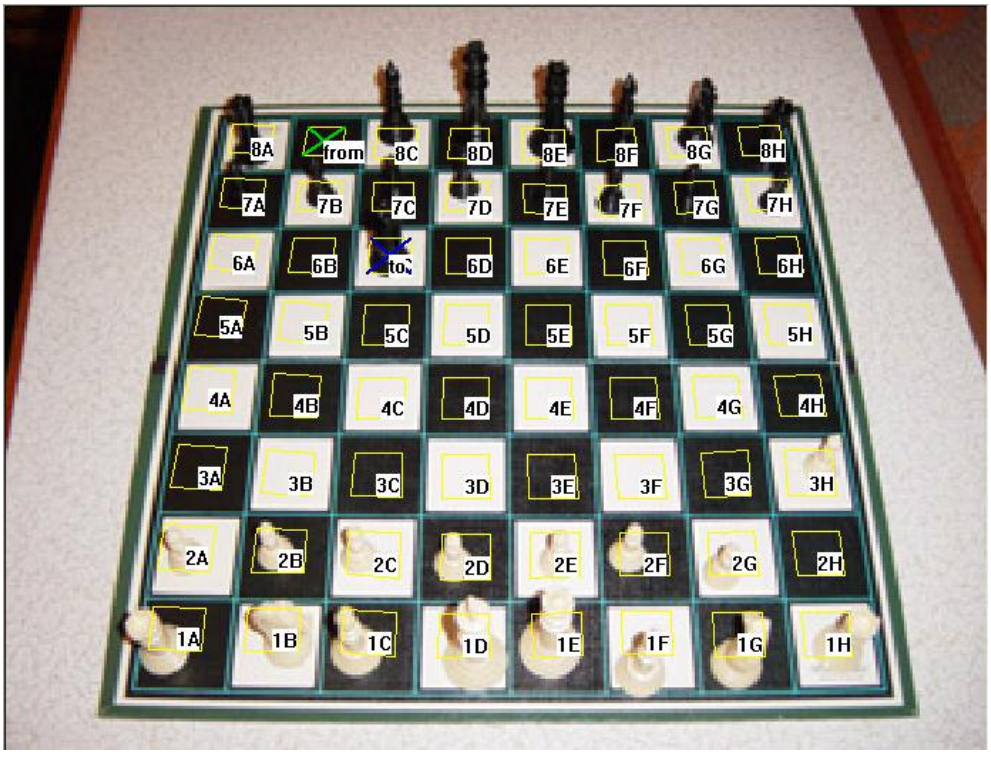
Appendix C – Sample Game sequence



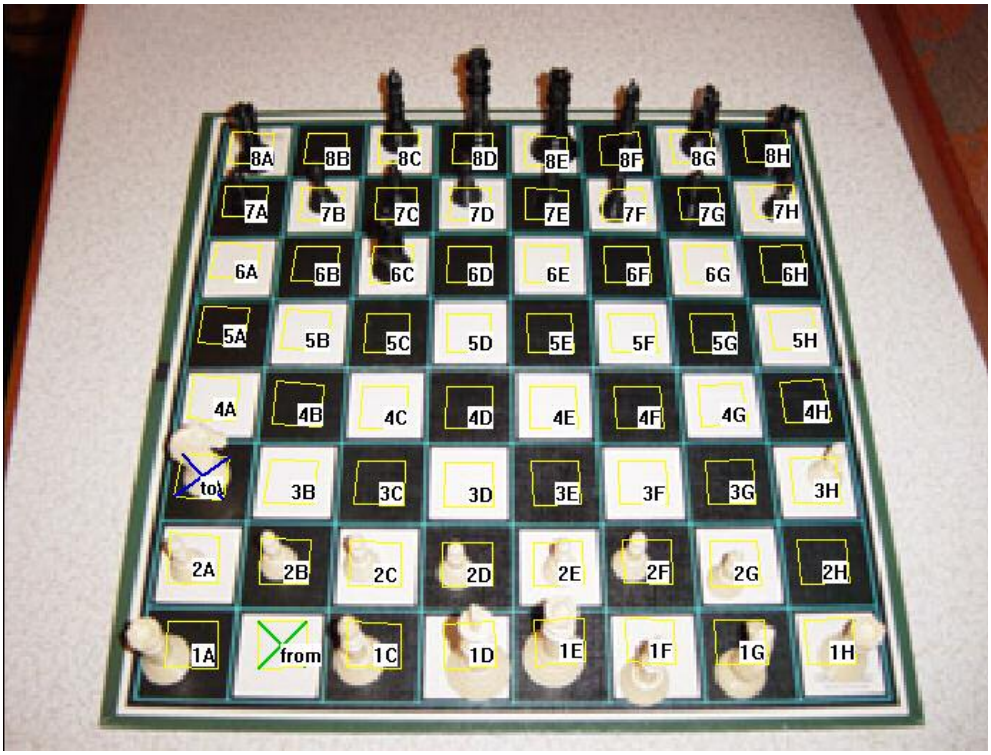
Game Sequence 1 – initial starting positions



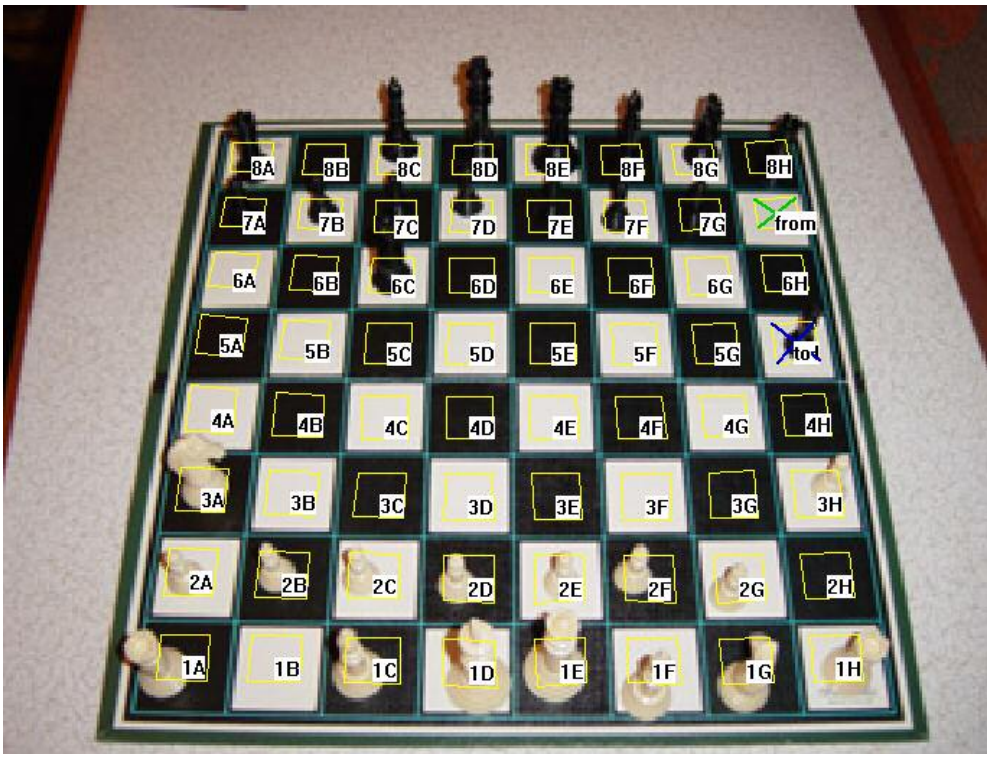
Game Sequence 2 - White Pawn Moves from 1 H to 3H, Incorrectly identified move, the movement from position is correct but where it moves to is incorrect



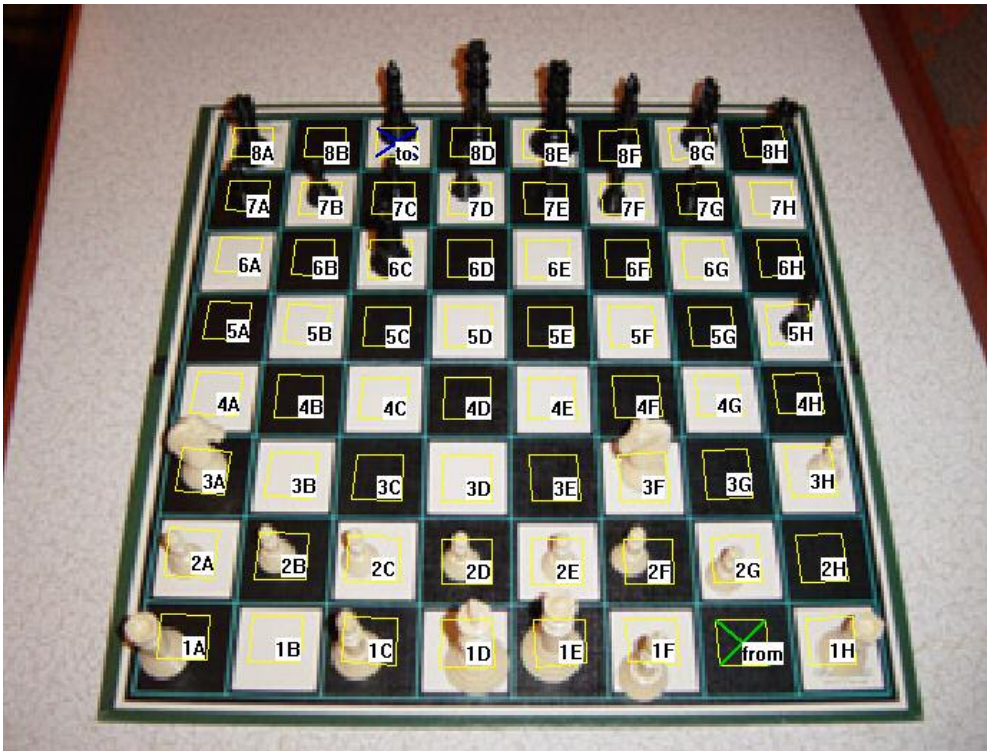
Game Sequence 3 – Black Knight moves from 7B to 6C, Correct identification of the move



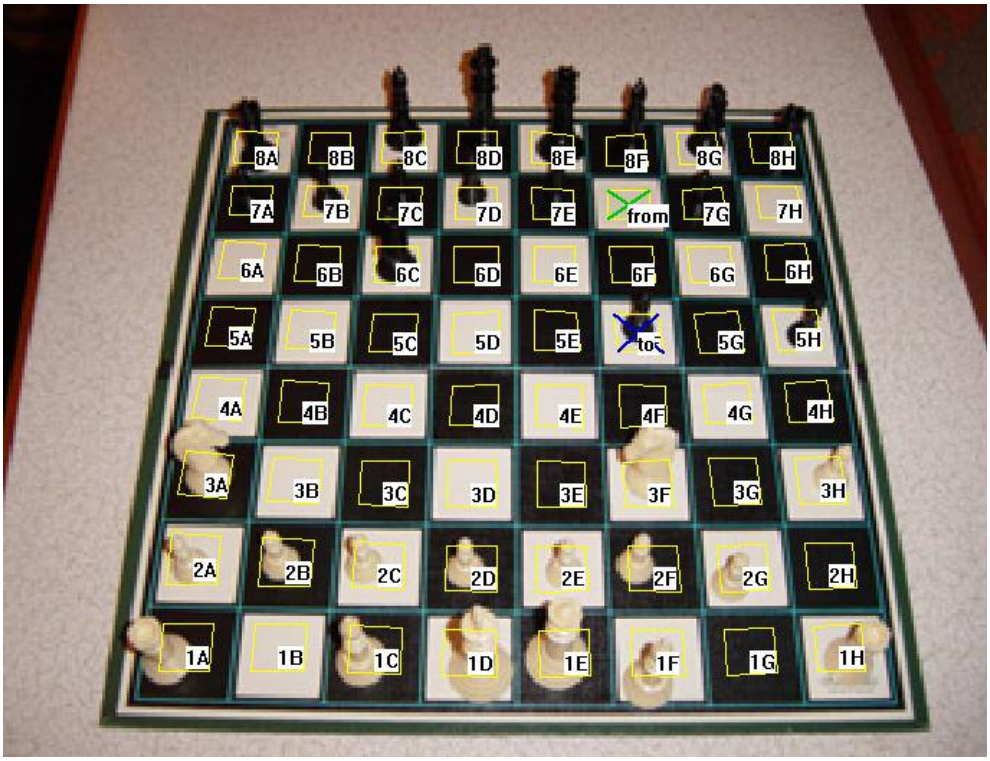
Game Sequence 4– White Knight moves from 1B to 3A, Correct identification of the move



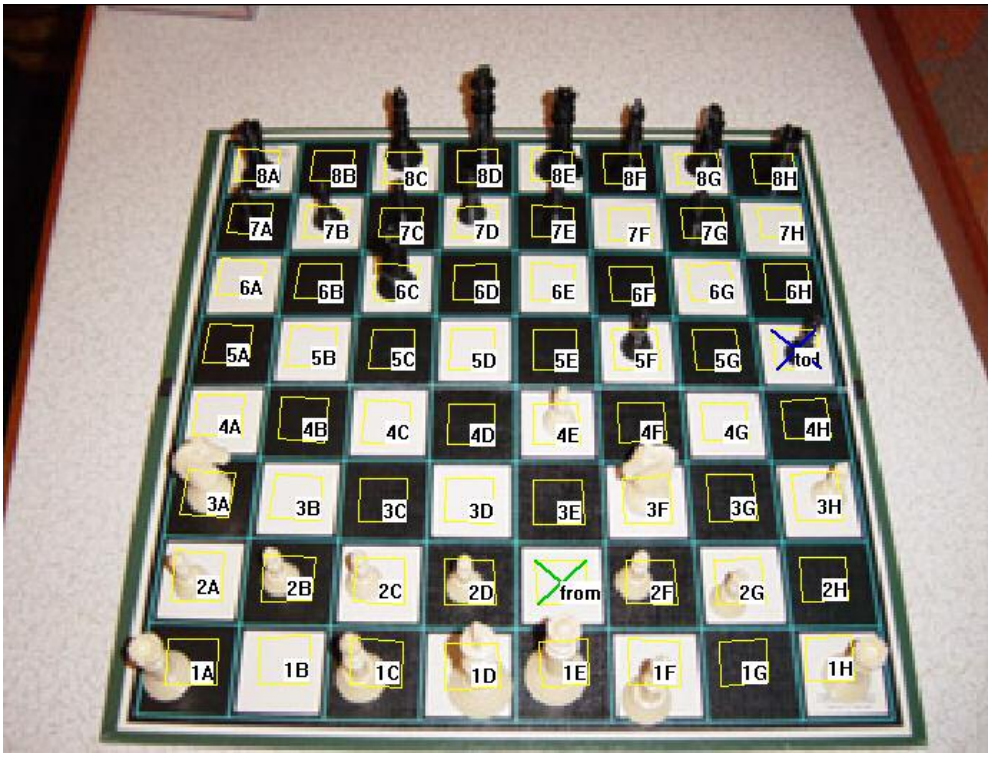
Game Sequence 5- Black Pawn moves from 7H to 5H, correct identification of the move



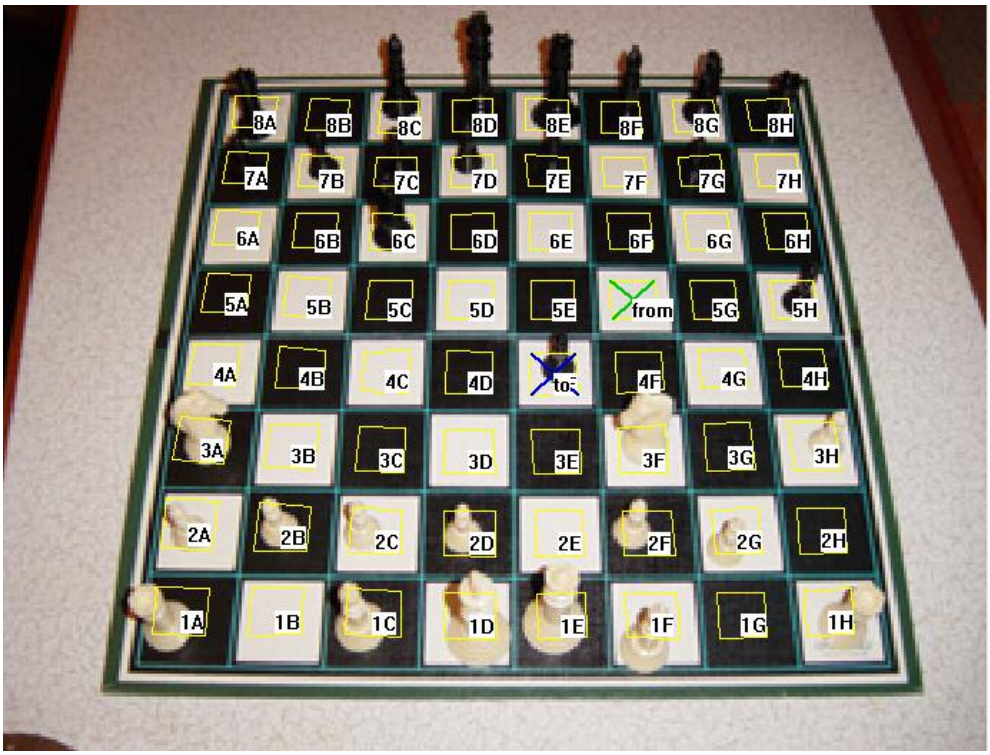
Game Sequence 6 - White Knight moves from 1G to 3F, Incorrect identification of the move from position is correctly identified



Game Sequence 7- Black Pawn Moves from 7F to 5F move correctly identified



Game Sequence 8- White Pawn moves from 1E to 4E, move incorrectly identified, from position is correct but the destination is incorrectly identified



Game Sequence 9 - Black Pawn moves from 5F to 4E move corectly identified

The Time Is Now
-Nasir Jones