To the Graduate Council:

I am submitting herewith a thesis written by Mark William Eklund entitled "Visual Tracking Using Correlation Techniques: Algorithms, Analysis, and Pipeline Architecture Implementation." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Mohan M. Trivedi

Dr. Mohan Trivedi, Major Professor

We have read this thesis
and recommend its acceptance:

Michael G Thomason

Donald W. Bouldin

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

Visual Tracking Using Correlation Techniques: Algorithms, Analysis, and Pipeline Architecture Implementation

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Mark William Eklund

May 1994

Abstract

In this thesis, a correspondence based tracking algorithm was detailed. This tracking algorithm tracked a region of an image located on an object. It used corrospondence between the training region and the current region to locate the object in the current image. Since manipulations of the object could dramatically change the geometry of the training region, the training region was updated while the tracking algorithm ran. Six steps were takin in creating the tracking algorithm. These were to

1. Define the issues associated with tracking.
2. Examine seven different tracking algorithms and choose a correspondence based tracking algorithm.
3. Introduce four correspondence algorithms to be compared.
4. Compare these four algorithms for accuracy and speed.
5. Choose the Minimum Noise and Correlation Energy (MINACE) filter for correspondence.
6. Build and evaluate a system that uses this tracking algorithm.

The tracking algorithm was implemented on a VME rack equipped with a Datacube MV20 pipeline processor and a Motorola 68030 processor. A data update rate of 15hz was achieved.

Contents

1 Introduction To Visual Tracking 1
- 1.1 Definition of Visual Tracking . . . 3
- 1.2 Issues in Visual Tracking . . . 4
- 1.3 Summary . . . 9

2 Visual Tracking Algorithms 11
- 2.1 Tracking Using Deformable Contours . . . 13
- 2.2 Model Based Tracking Algorithms . . . 16
- 2.3 Road Following . . . 19
- 2.4 Tracking Based on Groupings of Several Features . . . 21
- 2.5 Tracking by Verging Two Cameras on the Object . . . 24
- 2.6 Tracking Based on Optical Flow . . . 27
- 2.7 Correlation Techniques in Tracking Objects . . . 29
- 2.8 Summary . . . 30

3 Description of Four Correspondence Algorithms 33
- 3.1 Terminology . . . 34
- 3.2 Minimum Average Correlation Energy Filter . . . 37
- 3.3 Minimum Noise and Correlation Energy Filter . . . 41
- 3.4 Normalized Cross Correlation . . . 44
- 3.5 Sum of Squared Difference Correlation . . . 46
- 3.6 Summary . . . 47

4 Numerical Analysis of Four Correlation Algorithms 50
- 4.1 Experimental Setup . . . 51
- 4.2 Minimum Average Correlation Energy Filter . . . 59
- 4.3 Minimum Noise and Average Correlation Energy Filter . . . 62
- 4.4 Normalized Cross Correlation Filter . . . 65
- 4.5 Sum of Square Difference Filter . . . 65
- 4.6 Calculation Time of the Correspondence Algorithms . . . 69
- 4.7 Selection of Correlation Algorithm for Tracking . . . 72
- 4.8 Summary . . . 73

5 Description, Implementation, and Evaluation of a Real-Time Correlation-Based Tracking Algorithm **74**
5.1 Generalized Tracking Algorithm 75
5.2 Parameters Used in Implementation of Tracking 79
5.2.1 Correspondence Algorithm and Mask Size 79
5.2.2 Search Area Size 80
5.2.3 Object Location Update Rate 81
5.2.4 Mask Replacement Criteria and Rate 82
5.3 Hardware Specific Details of The Tracking Algorithm 83
5.3.1 Experimental Testbed 84
5.3.2 Pipeline Data Pathways 86
5.3.3 Correlation on the MV20 89
5.3.4 Truncation of the Mask to Eight Bits 95
5.4 Performance Evaluation 96
5.5 Summary 100

6 Summary and Concluding Remarks **102**
6.1 Summary 102
6.2 Concluding Remarks 104

BIBLIOGRAPHY **107**

VITA **113**

List of Figures

1.1 Issues of tracking 5

2.1 Deformable contours tracking a mouth 15
2.2 Model formed from the back of a truck. 17
2.3 A simple example of road following. 20
2.4 An example of tracking based on groups of features. 22
2.5 Image subtraction using vergence. 26
2.6 Flow vectors created by optical flow techniques. 28
2.7 Correspondence based tracking. 31

3.1 Coordinate system used in this thesis. 34
3.2 A region of the image is used in deriving the MACE mask. 38
3.3 Known and unknown regions in derivation of MACE filter. 38
3.4 MACE filter correlation. 40
3.5 MINACE filter correlation. 43
3.6 Normalized cross correlation. 45
3.7 Sum of squared difference correlation 48

4.1 Vice translation series. 53
4.2 Vice rotation series. 54
4.3 Vice z-translation series. 55
4.4 Rhino translation series. 56
4.5 Rhino rotation series. 57
4.6 Rhino z-translation series. 58
4.7 Effect of horizontal translation on MACE correspondence. 60
4.8 Effect of z-translation on MACE correspondence. 61
4.9 Effect of rotation on MACE correspondence. 61
4.10 Selection of MINACE threshold. 63
4.11 Effect of horizontal translation on MINACE correspondence. 63
4.12 Effect of z-translation on MINACE correspondence. 64
4.13 Effect of rotation on MINACE correspondence. 64
4.14 Effect of horizontal translation on normalized cross correlation correspondence. 66
4.15 Effect of z-translation on normalized cross correlation correspondence. 66

4.16 Effect of rotation on normalized cross correlation correspondence. . . 67
4.17 Effect of horizontal translation on SSD correspondence. 67
4.18 Effect of z-translation on SSD correspondence. 68
4.19 Effect of rotation on SSD correspondence. 68

5.1 Flow chart of tracking algorithm. 76
5.2 Hardware platform on which the experiments were conducted. 84
5.3 Various data pathways on the pipeline image processor. 87
5.4 Correlation of an 8×8 image with a 4×4 mask. 90
5.5 A 4×4 mask can be broken into four 2×2 masks. 92
5.6 Correlation of an 8×8 image with a 2×2 mask. 92
5.7 Correlation of four 2×2 masks. 93
5.8 Sum of the results of four 2×2 correlations. 94
5.9 Correlation on a MV20 pipeline processor. 95
5.10 Stability measurement of no motion in the X direction. 98
5.11 Stability measurement of no motion in the Y direction. 98
5.12 Typical tracking result during the second stability test. 99
5.13 Final x location of object with an initial x location of 200. 100
5.14 Final y location of object with an initial y location of 200. 101

List of Tables

1.1 A summary of issues associated with tracking objects. 9

4.1 "Training" and "run" time for the four correlation algorithms. 70
4.2 Summary of overall mask performance. 72

Chapter 1

Introduction To Visual Tracking

This thesis concerns itself with visual tracking. Visual tracking is the tracking of an object in a sequence of images. In many applications like unmanned vehicles, industrial processes, aids for the handicapped and service robots, tracking of an object is essential[41]. Using the motion of the object in the image and the camera model, the location and motion of an object can be estimated. With the motion and location of the object estimated, the object can be pursued, followed, intercepted and grasped. Once the object is tracked, unmanned vehicles can follow targets, obstacles, and landmarks while moving. Objects on assembly lines do not have to be exactly placed for industrial robots to successfully manipulate these objects. An automatic wheelchair can follow doorways and furniture while navigating a room. Finally, once an object is tracked, service robots can keep track of obstacles and landmarks when cleaning a house. In all of the examples a robust, reliable algorithm is required to keep track of whatever the object of interest may be [44].

The issues that cause visual tracking to be a difficult problem can be separated

into three categories. These are the object, the camera, and the environment. The factors associated with the object include the motion of the object and what the object is. The camera factors include the motion of the camera, the fact that the camera is a two dimensional representation of a three dimensional environment and camera noise. The environment includes lighting, occlusions and the other objects in the camera view. These factors will be discussed in detail later in this chapter.

There are seven major types of tracking algorithms. These are deformable contours, model based tracking, road following, many feature based tracking, vergence, optical flow, and correlation filter based tracking. All of these algorithms will be discussed in detail in Chapter 2.

This thesis develops a tracking algorithm that will track a single object. In the development of this algorithm, four point matching algorithms are examined. These algorithms are the normalized cross correlation (NCC), minimum average correlation energy (MACE) filter, minimum noise and correlation energy (MINACE) filter, and the Sum of Squared Difference (SSD). These algorithms are explained in Chapter 3 and a comparison of these algorithms is given in Chapter 4.

The tracking algorithm was then implemented an a VME rack using a Motorola 68030 processor and a DataCube MV20 image processor. In order to get the tracking algorithm to run at a maximum speed on the available hardware, the MINACE filter was used. The reasons for this choice and the outline of the tracking algorithm is given in Chapter 4. The tracking algorithm is explained and implemented in Chapter 5.

1.1 Definition of Visual Tracking

Visual tracking is the tracking of an object in a sequence of images. The object, the camera, or both may be moving. Recognition of the object is not required for visual tracking. Although recognition of the object is not required, the tracking algorithm needs some information on the object. This could be a list of features, the initial location of it in the image, or a model of the object or any combination of this information.

A function related to visual tracking is visual servoing. Visual servoing uses the tracking information to move the camera or a manipulator. This can be used in grasping operations and applications which the object needs to be kept at a certain location in the camera view. As an example of this, both Nikolaos Papanikolopoulos [33] and Rajeev Sharma [40] use visual tracking to locate an object. Papanikolopoulos uses visual servoing to move the camera in a way to keep the object centered in the image. Sharma uses a stationary camera to control the motion of a manipulator. In both cases visual tracking was used, but the camera was not moved in both cases.

In many applications like unmanned vehicles, industrial processes, aids for the handicapped and service robots, tracking of an object is essential. Unmanned vehicles need to know where roads and obstacles are located. Advanced industrial robots need to be able to follow the object that they are manipulating. An automatic wheelchair must be able to find doorways and furniture to navigate a room. Service robots that clean houses have to be able to keep track of obstacles and landmarks when cleaning

a house. In all of the examples a robust, reliable algorithm is required to keep track of whatever the object of interest may be. The system using the tracking algorithm relies on the robustness of the algorithm. A well designed system, save for the tracking algorithm, would be an unreliable system. For example if the automatic wheel chair had an excellent path planning program, it would be worthless if the chair could not follow landmarks and determine its location in the environment.

1.2 Issues in Visual Tracking

The issues that cause visual tracking to be a difficult problem can be separated into three categories. These are the object, the camera, and the environment. The factors associated with the object include the motion of the object and what the object is. The camera factors include the motion of the camera, the fact that the image created by the camera camera is a two dimensional representation of a three dimensional environment and camera noise. The environment includes lighting, occlusions and the other objects in the camera view. Most of these issues are demonstrated in Figure 1.1.

The first category of problems in visual tracking are related to the object being tracked. The object motion can move in four different ways. The easiest to track is in plane translation. This happens when the object either moves along the x axis or y axis. The effect of this translation would be the same as if the object were just cut and placed in a different location in the image. This is demonstrated in Figure 1.1a

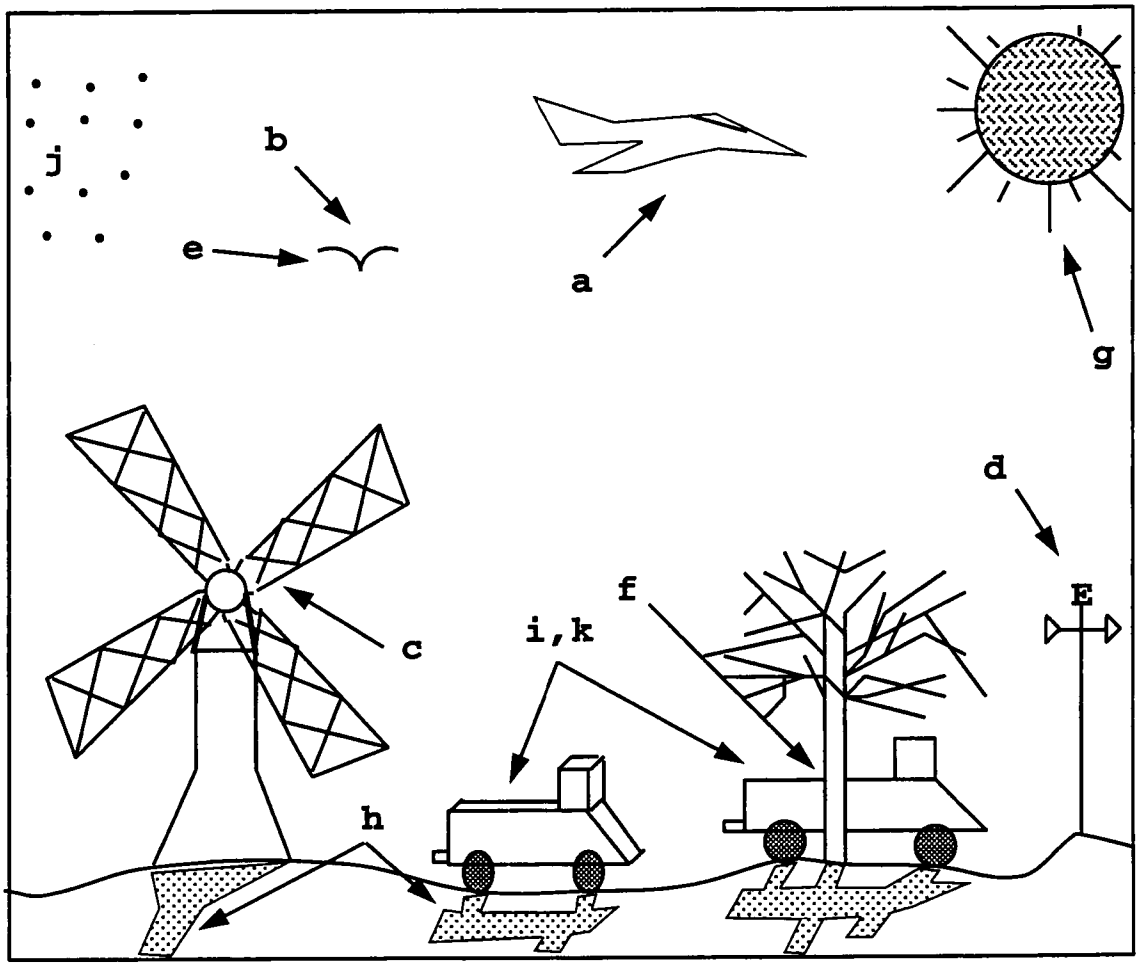


Figure 1.1: *Issues of tracking. (a) In plane translation. (b) Out of plane translation. (c) In plane rotation. (d) Out of plane rotation. (e) Lack of rigidity. (f) Occlusions. (g) Lighting intensity changes. (h) Shadows. (i) Similar objects. (j) Camera noise. (k) Two dimensional representation of three dimensional environment.*

with the jet moving across the image.

The second type of motion the object can move is an out of plane translation. This happens when the object moves directly toward or away from the camera. This causes the object to either enlarge or shrink. This is demonstrated in Figure 1.1b with a bird flying toward the viewer out of the page.

The third type of motion the object can move is an in plane rotation. This happens when the object rotates about the z axis. This is demonstrated in Figure 1.1c with the blades of the windmill rotating.

The fourth and final type of motion is an out of plane rotation. This is when the object rotates not about the z axis, but about either the x, they y axis or a combination of both axis. This is demonstrated in Figure 1.1d with the weather vane rotating around the y axis. The motion of the object can be any combination of the above listed motions.

Another factor that relates to the object is the rigidity of the object. The object is not guaranteed to stay in a certain orientation. The tracking of human lips is a simple example of this. This is demonstrated in Figure 1.1e where the wings of the bird flap causing the bird to be a non-rigid object to be tracked.

The second category of issues in visual tracking is the environment which the object being tracked is included inside. The lighting in the environment can change causing the intensity of the object and all surrounding objects to change. A matching algorithm sensitive to the intensity of the object can be affected by this change. A

source of lighting changes can be the available light as the time changes as demonstrated in Figure 1.1g. Another source of lighting intensity changes are artificial flourescent lighting. The light from these lights is cycling on and off at up to 120 Hertz. A camera could snap an image at any time during this cycle. This could cause drastic frame to from intensity changes.

The next problem caused by the environment, demonstrated in Figure 1.1h, is shadows. Unlike lighting changes that cause a uniform change in intensity, shadows can cause intensity changes local to parts of the object being tracked. This will cause false edges to appear in the image.

A third factor that the environment has in making tracking difficult is that there are objects in the environment that may partially hide, or occlude, the object being tracked. This is demonstrated in Figure 1.1f where the tree is occluding the car.

Other objects in the environment can also make tracking more difficult by having features similar to the object being tracked. This background clutter can cause false matches. This is demonstrated in Figure 1.1i where the two trucks are very similar. A tracking algorithm could track be tracking one truck and then accidentally change to the other truck.

The final category of factors in visual tracking is the camera. Like the object, the camera may be moved. It may be translated in any direction. Not only does this affect the object, but it also affects all of the other objects in the image. The camera may also me rotated about any axis. This also effects the object being tracked and

the other objects in the environment.

The camera has three other motions that the object could not move in. These motions are controlled by the zoom, the focus and the iris. Zooming changes the field of view. Focus changes the frequency information in the image and the iris will change the lighting in the image.

When following an object the largest hurdle is the fact that an image created by the camera is a two dimensional interpretation of a three dimensional environment. The background is three dimensional and the object itself is three dimensional. This is demonstrated in Figure 1.1k. The two trucks are similar objects, but they both have three dimensions of information being reduced to two dimensions. In fact, "Much work in visual motion analysis has concentrated on recovering general 3-D motion and scene structure from passively acquired 2-D image sequences" [5].

The final factor that effects visual tracking is camera noise. Random changes in the image due to the noise of the camera must be taken into account. This is demonstrated in Figure 1.1j with the random dot's appearing in the image. In a normal operation, there would be a uniform distribution of noise in the image. The noise was only placed in the corner of this figure to make the rest of the figure easier to view.

All of the listed factors can cause problems with tracking the object. In order to design a reliable tracking system these factors must either be taken into account and conquered, or they must be ruled out by constraints of the system. For example in

Table 1.1: *A summary of issues associated with tracking objects.*

Object	Environment	Camera
In Plane Translation	Occlusions	Camera Translations
Out of Plane Translation	Light Intensity Changes	Camera Rotations
In Plane Rotation	Shadows	Optical Parameters
Out of Plane Rotation	Background Clutter	Camera Noise
Object Rigidity		2-D Estimate of 3-D World

a tracking algorithm where a stationary camera watches a robot arm move, would not need to take camera motion into account. A robotic head [20, 7, 30, 1, 16] which consists of a pair of cameras which can be moved much like the eyes in a human head, would certainly have to take camera motion into account. These issues related to tracking are listed in Table 1.1.

1.3 Summary

In this chapter, visual tracking was defined to be the tracking of an object in a sequence of images. The factors that cause visual tracking to be a difficult problem were then separated into three categories. These were the object, the camera, and the environment. The factors associated with the object included the motion of the object and what the object is. The camera factors included the motion of the camera, the fact that the camera is a two dimensional representation of a three dimensional environment and camera noise. The environment includes lighting, occlusions and the other objects in the camera view. These factors were be discussed in detail in

this chapter.

Chapter 2

Visual Tracking Algorithms

This chapter will describe seven major types of tracking algorithms. The first type of tracking algorithm is deformable contours. Deformable contours map a function to contours in the image. As the image moves, this function slides to fit the image. The second type of tracking is model based tracking. Model based tracking uses lines, points, and regions that fit a model of the object being tracked. A high amount of prior knowledge of what will be required about the object being tracked. The third type of tracking is road following. When autonomous vehicles move along the road, they must keep track of the road. Although it is the camera moving and the road stays still, the vehicle must keep track of where the road is. The fourth type of tracking is many feature based tracking. In this type of tracking, several features are selected in an image. They are then matched to features in the next frame using the constraints like smoothness of motion. An interesting tracking algorithm implements camera motion to assist in following the object. With two cameras verged on the same object, small motions out of vergence can be detected. The cameras can be moved to

correct for this. The sixth tracking algorithm is based on optical flow. In optical flow, regions of motion are detected. These regions are associated with different objects. These regions can be found from frame to frame. The final tracking algorithm is correlation based tracking. In this the a region of the object in one image is searched for in the next image. This thesis uses this type of tracking algorithm.

Of the three major issues in tracking (object, environment, and camera) the most influential in the choice of the tracking algorithm is the object. An object that has a very structured composition like a plane or a car can be tracked well by a model based algorithm. A structure that is very flexible can be tracked by a deformable contours algorithm. An object that has an easily distinguishable motion can have optical flow track it. When little previous information is known about the object, a correlation based tracking algorithm is useful because it can be trained by the object to be tracked.

Once the tracking algorithm is chosen, the environment becomes a critical factor. Occlusions, lighting changes, and shadows can affect the tracking algorithm's capability to follow the object. The environment can influence the tracking algorithm chosen. All of the environmental factors affect all of the tracking algorithms. It is difficult to say that one tracking algorithm is more robust against lighting changes. This depends on how well the tracking algorithm is implemented.

The last factor, the camera, will influence the decision of the tracking algorithm least of the three factors. Noise and camera motion are parameters that all track-

ing algorithms must deal with. There is no algorithm that is exceptionally robust to noise compared to the other algorithms. No algorithm is more robust to quick motions than other algorithms. These are not issues for algorithm choice, but for the implementation capabilities of each algorithm.

2.1 Tracking Using Deformable Contours

The first type of tracking algorithm discussed is deformable contours. Deformable contours are used for objects that are not rigid. This means that deformable contours are used in tracking objects that have changing geometries. Examples of objects tracked are fingers[42], speaker's lips[21], living cells[24], eyes[51], and edges of blocks of wood[10].

Terzopoulos explained the behavior dynamic contours to be "the behavior of the model is governed by elastodynamics equations and the model is coupled to visual data through a force field computed from one or more images [42]". A simple analogy to a deformable contour is a beaded chain that is placed on a surface. This beaded chain would twist and roll until the chain stopped moving and ended up in a low energy state. The motion of the chain was governed by two forces. These were the need for it to move down the surface to the area of lowest energy, and the fact beads in the chain were connected together. The field of force in this analogy is gravity pulling the chain down the surface. The elastodynamics of the chain is the fact that the individual beads are tied together by the chain.

A simple example of a deformable contour used in tracking is shown in Figure 2.1. In this figure, the line is the deformable contour tracking two frames of a speaker whose lips are moving. The speakers lips move from a frown to a smile. Perhaps the speaker has just finished presenting a thesis. Initially, the contour has the form of the first frame (a frown). The contour then starts to twist and turn moving toward the darker area in the image. After some iterations, the contour has moved to the new shape of the lips (a smile). Thus this algorithm has succeeded in tracking the lips.

Curwen and Blake [10] used dynamic contours to track the edges of objects using real time algorithms. The computer system used was a small transputer network consisting of 11 processors. They could track three separate edges at video-rate (30hz). The system programmed the contours to have an affinity for image features such as brightness, or high intensity-gradient. The assumption is made that the feature shape will only change slowly. Initially, the contour is placed in the training image and allowed to freely change shape. Once the contour stabilized to a single shape, that shape is used to influence the next shaping of the contour in the next frame. The shape of the contour in the next frame influences the shape in the following frame and this continues for all of the consecutive frames. The reason they chose to have a shape assumption is because with no shape assumptions, the contour flows around corners. When a shape assumption is incorporated, it moves almost rigidly with the corner feature. This tracking algorithm can track a small toy vehicle moving with a maximum velocity of approximately 35 mrad in a field of view of 300 mrad.

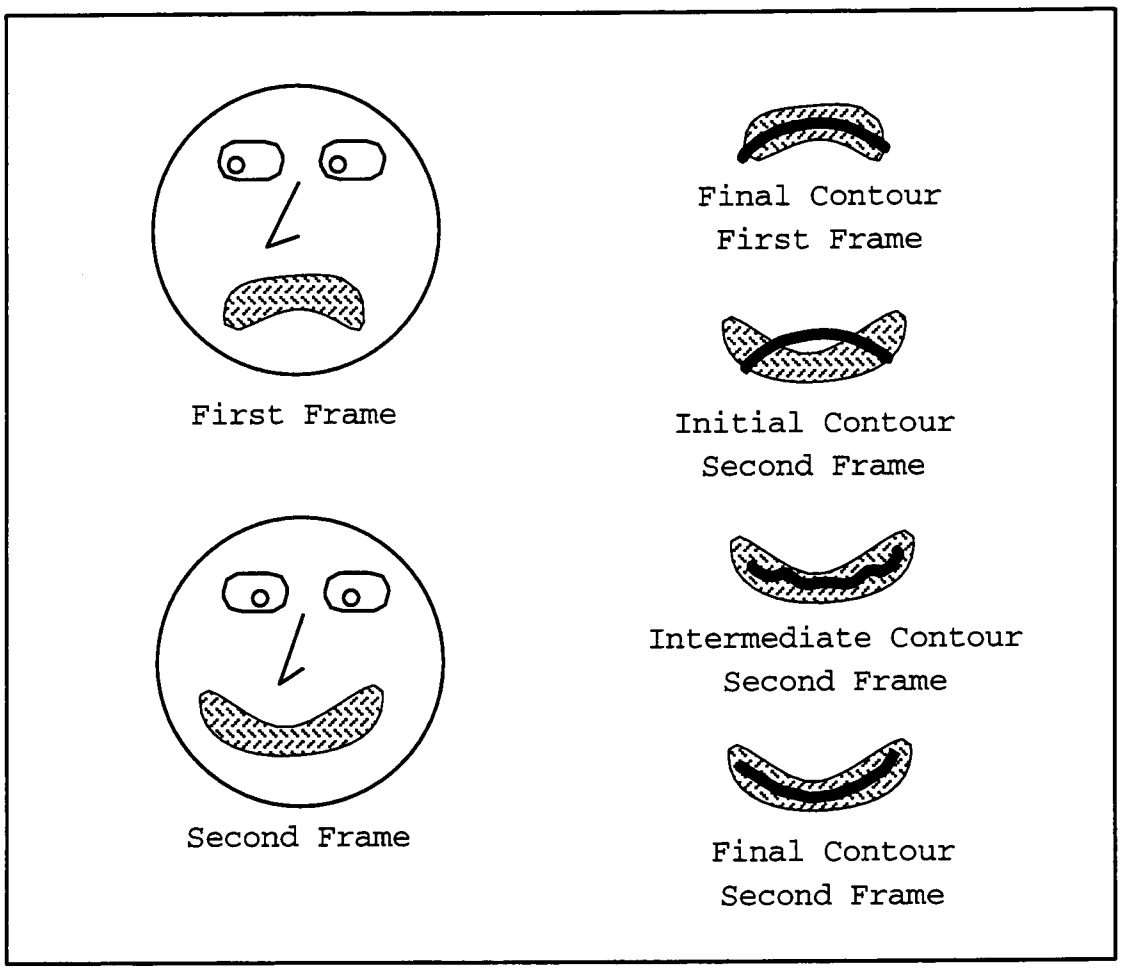


Figure 2.1: *Deformable contours tracking a mouth. Show in this figure are three phases of how a deformable contour changes to a change in the tracked target, a mouth.*

2.2 Model Based Tracking Algorithms

Model based tracking uses lines, points, and regions that fit a model of the object being tracked. Since the object being tracked is based on a model of the object, prior knowledge will be required about the object being tracked. A model is based on a collection of related features in an image. These features could be lines, regions, colors, points, or any other feature that can be extracted from the image. Figure 2.2 is example model of the back of a truck. The actual object, the truck's rear, is composed of two doors, a bumper, a bar across the top of the doors, a square target centered on the doors, and a license plate. From this, a model can be chosen that uses features that are unique to this truck. These features, six horizontal and four vertical lines, can be used to model the back of the truck. It should be noted that not all of the features need to be used. For instance, the license plate was not included in the model. Not all features are needed to form a useful model of the object. Just as a baseball player does not need to watch the threads on a baseball to hit a home run, not all features of an object need to be tracked to successfully follow the object.

One example of a model based tracking algorithm is the one developed by Harris [18]. Harris created a model based three-dimensional tracking algorithm called RAPiD (Real-time Attitude and Position Determination). The 3D object model consists of selected control points on high contrast edges, which can be surface markings, folds or profile edges. It was demonstrated that this algorithm could track at 50hz tracking films of planes, missiles, and landing strips. The points being tracked were

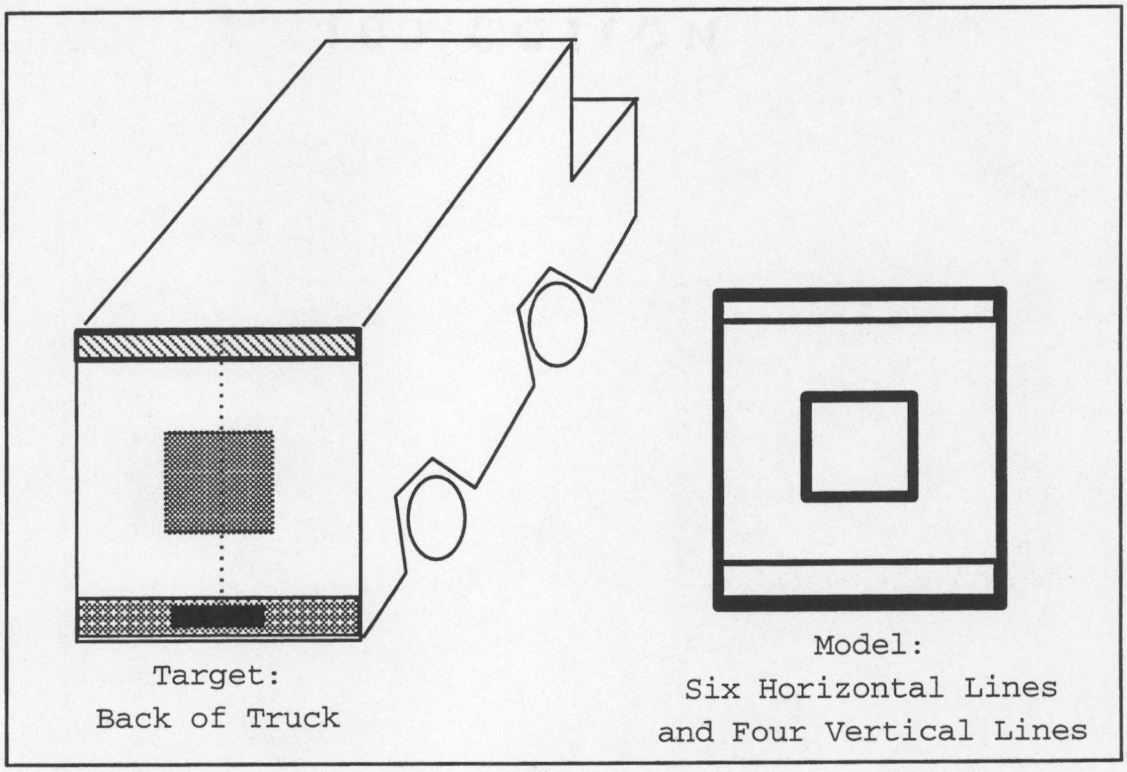


Figure 2.2: *Model formed from the back of a truck. Shown in this figure is the truck to be tracked and the model formed. The model does not use all features available.*

the profile edges of the object. Thus the algorithm relied on the fact that that there was a high contrast between the image and the image background. One useful result of this information is that since the model is known, the orientation and position can be estimated with respect to the camera.

A group at the National Institute of Standards and Technology [38] trained their model on the back of a truck. The model used was the model shown shown in Figure 2.2. It was composed of the lines it would take to create the rectangles that would outline the back of the truck and a target placed on the back doors of the truck. The model would initially be initially placed on the first image. In the consecutive images, the tracking algorithm would detect edgels. These edgels would then be associated with one of the lines that composed the model of the back of the truck. These new groupings of lines would be used to update the orientation of the model of the truck. The lines with a smaller variance of edgels normal from the line location would have greater influence on the training of the orientation of model for the back of the truck. Thus, a line with very well contained edge locations would have a greater an influence than a line composed of a large spread of edgels. If one of the lines was partially obscured by another line perhaps from a shadow, the edgels of the line and the shadow would have an influence on the location of that line. These edgels would have a large variance. Since the algorithm trusts results with low variances, this shadow would have little effect on the orientation of the model. This created a system that was robust against occlusion. The algorithm was shown to track with

approximately sixty percent of the object occluded.

2.3 Road Following

When an autonomous vehicle is traveling down a road, it must be able to detect the road. In essence, the road is a moving object in the camera's field of view. The car must track the road in order to stay on the road. Road following can be thought of a tracking an object that is not fully in the image. It might be argued that road following is simply detection of the road in each image. This would be the case if each image were thought of as an independent detection of a road. This is not the case because the information from the past detections are also used in following the road. The previous information can be used as an initial starting point in detecting the road in the following image. This would link the detection of the road in the image to all the past detections. Also, the motor controlers do not use a single data point to control the motors. Instead, they use several past data points to estimate velocity, and reduce noise. It can be seen from this that road following is not simply the detection of an object, but the use of that detection and previous detections to follow the road. For these reasons, road following is included in this chapter.

One way of detecting the road is by using gradient techniques to find specific structured features of roads[22, 47, 48, 45]. These structured features may be road edge lines, lane markings, or road edges. Other methods try to detect the surface through histogram and thresholding[23, 46]. Another uses surface colors to detect

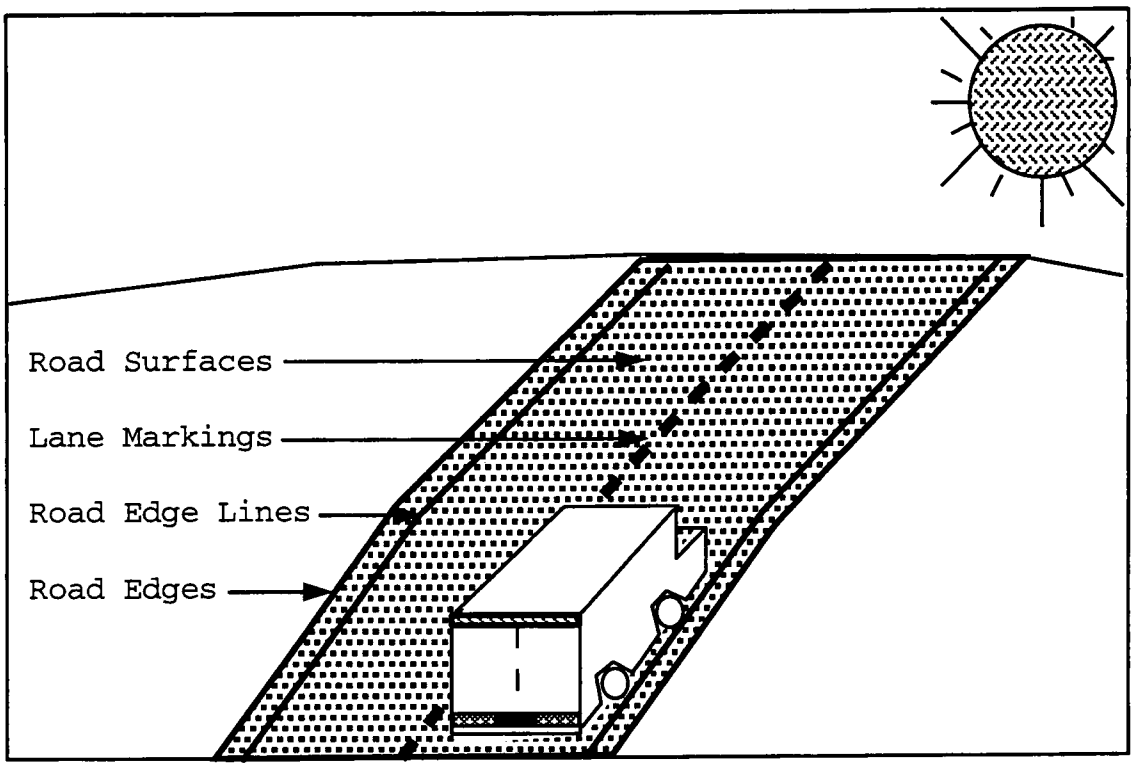


Figure 2.3: *A simple example of road following. The autonomous truck may be tracking the road surface, the road edges, the road edge lines, or the lane markings.*

the road region[9]. Shown in Figure 2.3 is a simple example of an autonomous vehicle traveling along a road. The road following algorithm could be using road edge lines, lane markings, road edges, or the region of the road surface to track the road.

One interesting approach on road following was done by Crisman [9]. Crisman used color to train a pattern recognition routine. Initially, the roadway would be marked by an expert system (a human operator). Using this region, the road pattern was trained. Using this, the roadways could be detected. Crisman claimed that this worked much better than traditional gray level thresholding. Crisman also showed that this was robust to occlusions like leaves. It also did well when shadows of trees

were cast across the road's surface.

2.4 Tracking Based on Groupings of Several Features

Another approach to tracking is a tracking based on several features. In this several simple features are extracted. These features are then matched in the next frame. These features can be anything that image processing techniques can extract quickly. These include line segments, points and corners. Since time is usually a critical factor in tracking, making each step in the tracking algorithm simple can speed the tracking of the object. These features can easily be extracted with a simple correlation mask and a threshold. Once these features are found, pairing these features with the similar features in the next frame in the sequence of images can be done. Groups of features with similar displacements can be collected and tagged as an object. Since there are several features, not all of them are needed for tracking the object. If a few of the features are matched incorrectly, these will not be grouped with the object. The tracking algorithm does not have the problem of being dependent on a single feature point. This means that if a point is occluded, it's effect on the tracking will not be significant. Also, since this algorithm uses several points, localization accuracy can be sub-pixel.

A simple example of tracking based on groups of features is shown in Figure 2.4. First, a simple correlation is used to detect the corners. In this example, the corners

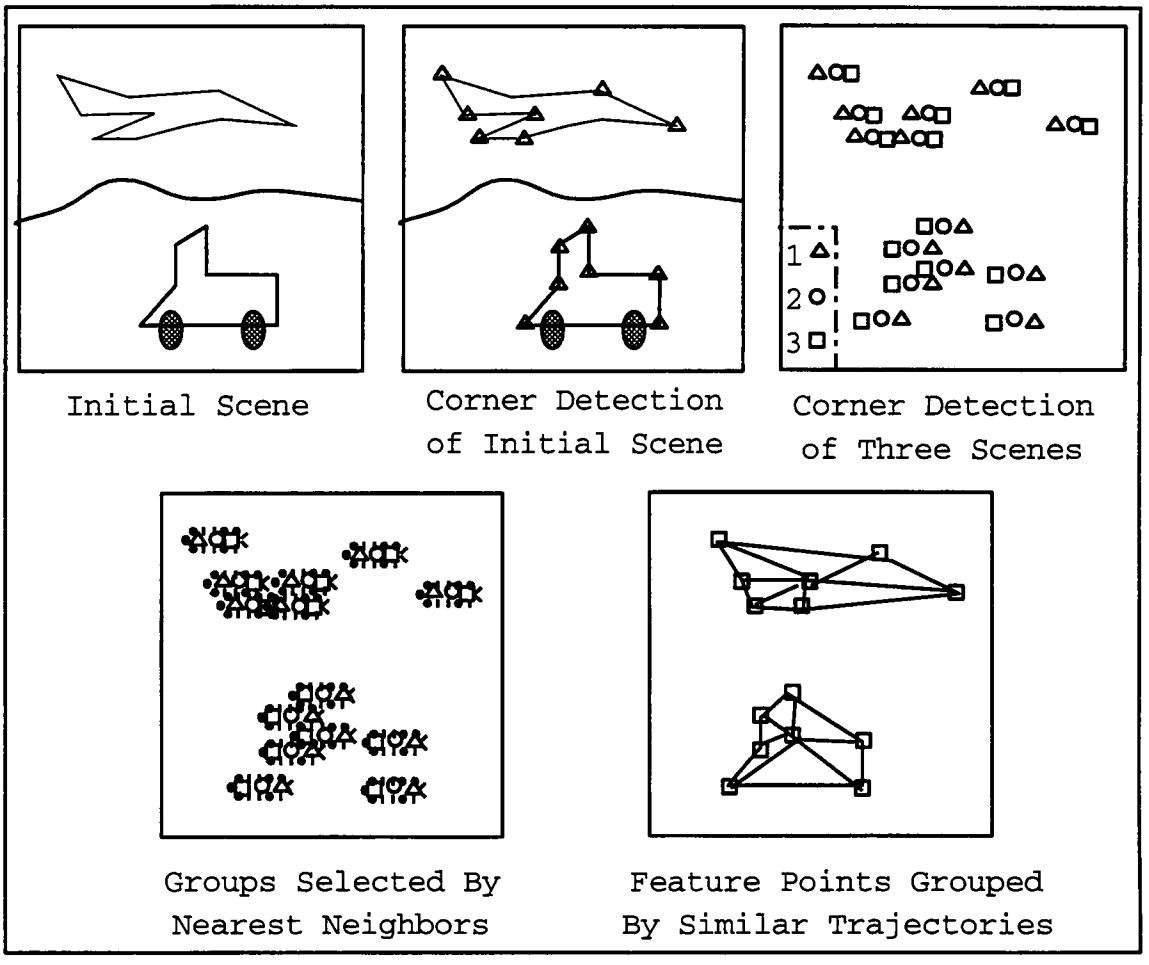


Figure 2.4: *An example of tracking based on groups of features. First corners are detected. These corners are detected in three scenes and then grouped by minimum displacement from the previous scene. From this information, a trajectory for all the corners can be found and groups of corners with similar trajectories can be grouped and tracked.*

are detected in three scenes and then grouped by minimum displacement from the previous scene. From this information, a trajectory for all the corners can be found and groups of corners with similar trajectories can be grouped. By continuing this process, the two objects may be tracked. Notice this is an ideal case. All of the corners are detect in every frame. No corners cross each other. No new corners are detected in the frames. If any of the listed happened, the process would become more complex.

One experiment that demonstrates the basic ideas of this tracking method is the one done by Sethi[39]. Sethi used feature points in a sequence of images. These feature points were either synthesized or manually selected from a sequence of images. Initially, he matched the feature points in one frame with the nearest points in the next frame of the image. He used smoothness of motion to then refine this initial matching. In this way, if there were two points very close to each other, but moving in different directions, the direction of motion would be used to influence the matchings with previous frames. The velocity of the sequence of points was estimated from the sequence of points in the previous frame. Sethi used an iterative technique in which he estimated the velocities from the correspondence of the points in the previous iteration. He would then estimate the correspondence with influence of the estimated velocities in the previous iteration. These experiments did not use any feature retrieval routines. They also only had the points being tracked. These points were present in every frame. This means that he did not address the real world problem of

points appearing during the algorithm or points disappearing during the algorithm.

Another method using line segments was the one done by Zhang and Bergholm[52] who tracked line segments in a moving stereo image. They considered three rigidity constraints in the matching of line segments in their stereo pairs of images. The first constraint was line segment length. Only line segments of similar length were matched. The second constraint was line orientation. Thus only line segments oriented in the same direction were matched. Finally smoothness of motion was used as a rigidity constraint. Because of this only line segments would be matched that would retain the velocity of the line segment to be matched. With these constraints, the matching of line segments was simplified. There was no mention of how the line segments were found. There are images shown with line segments found in them, but there is no mention of how they derived these segments from the images. The extraction of line segments alone could be a time consuming algorithm.

2.5 Tracking by Verging Two Cameras on the Object

In all of the previous algorithms and the algorithms to follow, the tracking algorithms do not depend on the motion of the camera. This means that although the cameras could be moving while the object is being tracked, it will not necessary for them to be moving for the tracking algorithm to operate. The algorithm explained in this section depends on camera motion to work. This is an actual application of active vision where the cameras are moved as an integral part of the tracking algorithm.

This system was developed by Brown and Coombs [8, 6]. In this system, there are two steps in tracking the object. In the first step, the two cameras are verged on the object. The two cameras are verged on an object when the object is located in the center of the image plane of each camera. The second step is to locating the object in the image. Since the cameras are verged, this problem can be simplified by a technique used by Brown and Coombs which is explained below.

In order to verge the cameras, the disparity of the object in the right and the left images must be zero. The cameras are moved in an attempt to make the disparity zero. The disparity of the cameras is found using power cepstrum. The power cepstrum is a technique developed for detecting echoes. The left and right images are placed side by side and the Fourier transform is taken of them. If there is a repeat from one image to the next, a large peak happens at that frequency. The large peak is caused by the object being in both images. This distance gives an error term which is used in a control circuit for the cameras. This control circuit takes the disparity to zero. The entire images are not placed beside each other. Smaller sub-images sub-images in the area the object is suspected to be are placed beside each other and the power cepstrum is calculated for them. This speeds up the algorithm since there is less information of which the fourier transform is taken.

Now that the cameras are verged, the object must be found in the image. The first step in finding the object is by subtracting the images. All areas that are the same in both images return with a result of zero. Other areas give both positive and

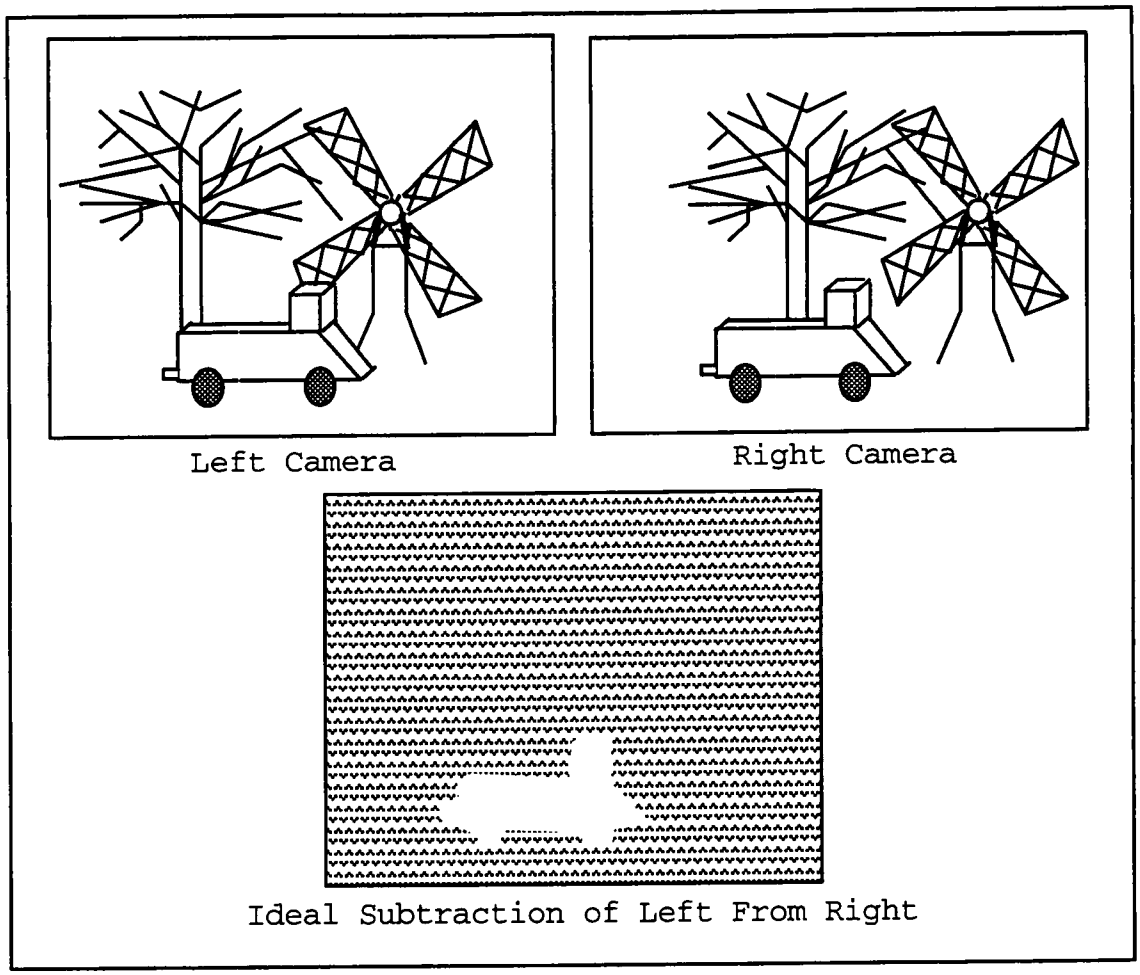


Figure 2.5: *Image subtraction using vergence. The top two images are the left and right images verged on the car. When these images are subtracted, the car will have a zero value. The other objects in the image will not be overlapping and will not subtract to zero.*

negative values not equal to zero. This is demonstrated in Figure 2.5. The left and right images are merged on the car. Notice that the car is in the same location in the image, but the tree and the windmill are in different locations in the image. When the images are subtracted, the car subtracts to zero, while all of the rest of the image results in different values.

The flaw in this algorithm is the assumption that there are no regions of similar

values. If the background sky were a constant value, it would subtract to be zero. The algorithm would then detect the sky as a part of the object.

2.6 Tracking Based on Optical Flow

Optical flow can be defined as the velocity with which image points move in a sequence of images[11]. Optical flow does not deal with structures in the image. Optical flow only deals with image points. The results of optical flow are flow vector fields taken from the image. These flow vectors are demonstrated in Figure 2.6 where there are three objects in the image. Two of them are in motion. The vehicle is moving across the image and the windmill is rotation. Using the two images, flow vectors from one image to the next can be accomplished. These are shown by the arrows in the third frame of the image. These flow vectors have both a magnitude and a direction. Objects moving quickly have larger magnitudes than slow moving objects. In the figure, the vehicle has larger flow vectors than the windmill.

Optical flow [19] is a technique which takes a sequence of images and uses techniques to find the flow of points in the image. This is a blind routine which does not do anything but give vectors showing the flow of objects in the image. Matching regions of similar flow causes the objects to be detected.

Allen [2, 3] uses optical flow to track a toy train. The algorithm has two stationary cameras locating the moving objects in the image. The only moving objects in the images are a toy train and a robotic arm. Since the arm is always exactly known, the

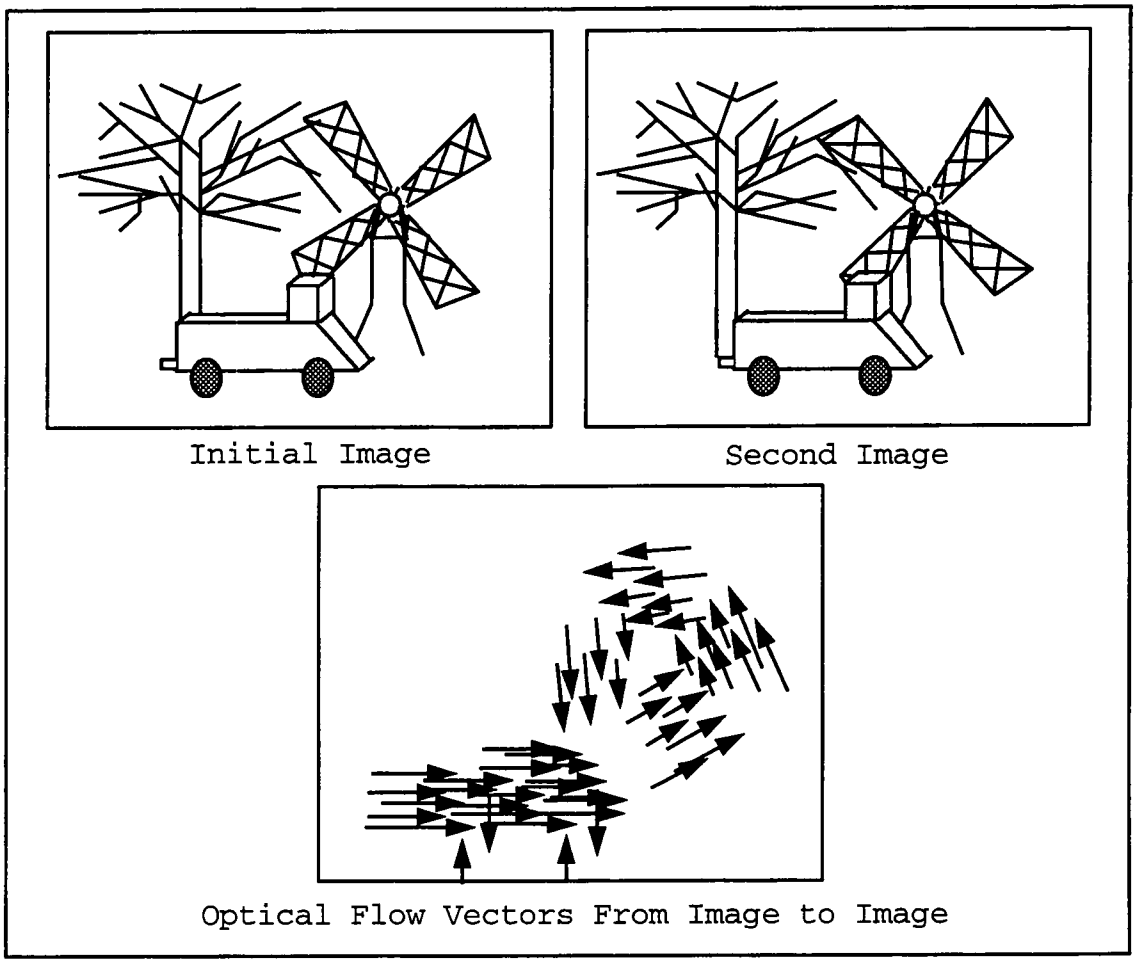


Figure 2.6: *Flow vectors created by optical flow techniques. This figure demonstrates this with a car translating and a windmill rotating. This slight change in image position can produce optical flow vectors as demonstrated in the third image. Notice that the vectors for the car have a larger magnitude than that of the windmill.*

only object to be tracked is the train. The results of the optical flow calculations may be noisy, but they give regions where motion is flowing. The only regions with flow are the robotic arm and the train. With two cameras operating, the tree dimensional location and velocity of the train can be found. This is a very constrained system. There is a robotic arm of known location and a train with a predictable trajectory. Any other motion, like a bug flying in the scene confuse the optical flow algorithm. The thrust of Allen's work is not the tracking of the object, but the control algorithm using visual feedback of the cameras.

2.7 Correlation Techniques in Tracking Objects

The final technique used in tracking is the one used in this thesis. In correlation based object tracking, a small area cut from previous image and searched for in the images to follow. This technique is popular in stereo vision applications where the object in the left image is searched for in the right image [15, 27, 43]. In order to speed up the correlation, the object is not searched for in the whole image. If the motion of the object is known to be less than some speed, the correlation search area only needs to be large enough that the maximum distance traveled in the image plane from one frame to the next need be searched. The two more common techniques of correlation, normalized cross correlation [34, 26, 28, 4] and sum of squared difference [33] are examined in this thesis. Two correlation techniques used in the optical filtering field are the minimum average correlation energy (MACE) filter [25] and the

minimum noise and correlation energy (MINACE) filter [36, 37] are also examined in this thesis.

Figure 2.7 demonstrates one step in correspondence. In this, a small region of the object to be tracked is used in a correspondence algorithm to find the corresponding region in the next image. The results of the correspondence algorithm would give values that change as to how close the region is to the region being searched for. A high value would correspond to a bright area. In this example, the bright area would be located near the front of the vehicle. Tracking then is the repetition of this correspondence algorithm.

Papanikolopoulos [29, 31, 33] uses sum of squared difference correlation to track feature points sequences of images. These feature points usually were corners, letters, or regions of high contrast. One of his concerns was having a confidence measure of the result of his correspondence. He examined this confidence measure in [32]. His confidence measure could also be used in selecting robust points to track. This technique finds of the minimum of the sum of square difference between regions in an image. The size of the region he was searching for was 20×20 pixels. The area that he searched allowed a maximum displacement of 16 pixels.

2.8 Summary

With the problems in visual tracking defined, the seven major tracking algorithms were discussed. These were deformable contours, model based tracking, road follow-

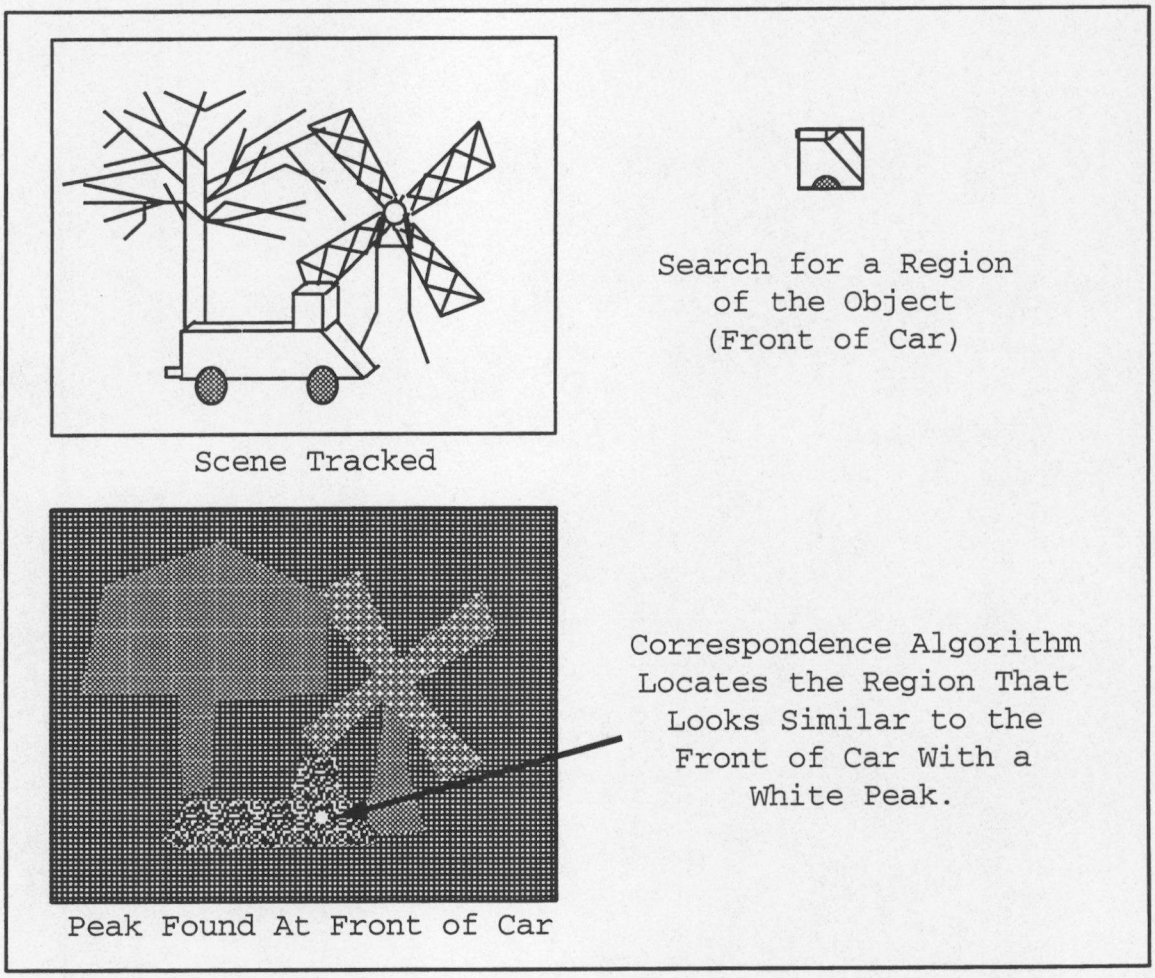


Figure 2.7: *Correspondence based tracking. Pictured here is the object to be tracked, a car. The region that is searched for, front of a car. An an example result of correspondence. The brighter the results, the closer the match. In this the region was found where the white patch is in the image.*

ing, many feature based tracking, vergence, optical flow, and correlation filter based tracking.

Each of these techniques have their advantages and disadvantages. Some work better with rigid objects. Some are useful for tracking objects that deform. One consideration is the speed at which the algorithms can be implemented. When considering speed, the hardware available must be considered.

Chapter 3

Description of Four Correspondence Algorithms

In Chapter 2 the several tracking algorithms were discussed. The algorithm used in this thesis is based on correspondence. In this chapter four correspondence techniques are introduced. In the first section of this chapter, terminology is given. This terminology will be used in deriving the correspondence algorithms in the sections to follow. In the four sections following the terminology section, four correspondence algorithms are described. An example of the result of each algorithm is also given with each section. These algorithms include the minimum average correlation energy (MACE) filter, minimum noise and correlation energy (MINACE) filter, Sum of Squared Difference (SSD), and the Normalized Cross Correlation (NCC) Filter.

The next chapter that follows this one runs a battery of tests to compare the tracking algorithms. This information will be used in deciding which algorithm to use with the real time tracking algorithm. This correspondence is used when implementing the tracking algorithm described in Chapter 5

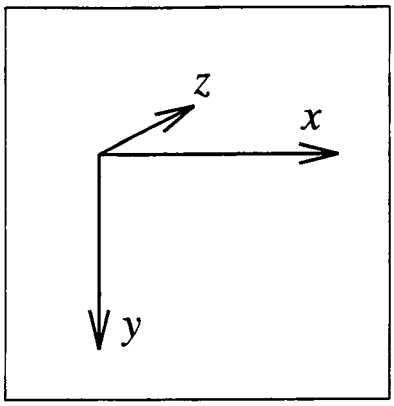


Figure 3.1: *Coordinate system used in this thesis.*

3.1 Terminology

It is assumed that the reader has knowledge of basic image processing techniques. Nevertheless, since terminology varies from book to book, this section will state some of the terminology and the symbols used in this thesis. Much of the symbols are taken from *Digital Image Processing* [17]. Using this terminology, the four correspondence algorithms will be described.

The image in image processing is a two dimensional, array of values of magnitude. These can represent range data or intensity data. This thesis will only be concerned with the intensity data. The coordinate space in this thesis is right handed. The top-left portion of the image is coordinate $(0, 0)$. The x coordinate increases toward the right of the image. The y axis increases toward the bottom of the image. The z axis refers to distance from the camera with the origin located at the camera This coordinate system can be seen in Figure 3.1.

The discrete Fourier transform is used in the creation of some of the masks in this thesis. The two dimensional discrete Fourier transform is defined as

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}. \tag{3.1}$$

M is the width in the x dimension. N is the width in the y direction. $f(x,y)$ is the image of which the Fourier transform is taken. The inverse two dimensional discrete Fourier transform which transforms from the frequency domain to the spatial domain is

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(\frac{ux}{M}+\frac{vy}{N})}. \tag{3.2}$$

One property of the discrete Fourier transform is that the result of the transformation is periodic. using the transform in Equation 3.2 it can easily be shown that

$$f(x,y) = f(x+iM, y+jN) \tag{3.3}$$

where $i, j \in 0, \pm 1, \pm 2, \cdots$.

The fourier transform creates complex numbers as its components. The real values can be represented as $R(u,v)$ and the imaginary numbers can be represented as $I(u,v)$. When the Fourier transform is displayed as an image in this thesis, it is actually the power spectrum of the Fourier transform. The power spectrum is

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v). \tag{3.4}$$

The complex conjugate of a function is the negation of the imaginary part of the function. Mathematically this is

$$F^*(u,v) = R(u,v) - I(u,v) \tag{3.5}$$

Correlation is a useful function for finding features in an image. In correlation, a mask is slid across an image to create a result. Mathematically this function can be shown as

$$g(x,y) = f(x,y) \circ h(x,y) = \sum_{-\infty}^{\alpha=\infty} \sum_{-\infty}^{\beta=\infty} f^*(\alpha,\beta)h(x+\alpha, y+\beta). \tag{3.6}$$

For this thesis, $h(x,y)$ would correspond to a mask slid across the image. $f(x,y)$ would correspond to an image. Usually the mask has dimensions much smaller than the image. The result of this correlation is given as $g(x,y)$. Since $f(\alpha,\beta)$ is a real image taken by the camera, the conjugate of $f(\alpha,\beta)$ is the same as $f(\alpha,\beta)$. In the frequency domain correlation is simply a multiplication as

$$f(x,y) \circ h(x,y) \Leftrightarrow F^*(u,v)H(u,v). \tag{3.7}$$

3.2 Minimum Average Correlation Energy Filter

The minimum average correlation energy filter (MACE) was used by Abhijit Mahalanobis [25] to detect objects of different orientations in an image. This was done by training the mask with several versions of the same object at different orientations. The mask used here was trained by a single image. Since there is the assumption of no prior information, and the time to wait for multiple images to come could be detrimental to tracking the object, one image was used to train the mask.

The driving idea of the MACE filter is simple. The idea is to create a mask in which the result has a peak where the object is located and low values everywhere else. A correlation mask is created by using a small region of the image. This region is shown in Figure 3.2 which demonstrates that the image is $f(x, y)$, and the region is $\mathbf{x}(x, y)$ and of size $m \times n$. In the derivation of the MACE mask ($h(x, y)$) the region and the desired result are known and it is the mask that must be derived. This is demonstrated in Figure 3.3 where the region and result are known, but the mask needs to be derived.

Ideally this desired result would be a single peak surrounded by zeroes. In essence this is a discrete delta function or

$$g(x, y) = \begin{cases} 1 & \text{if } x = 0, y = 0 \\ 0 & \text{otherwise} \end{cases} . \tag{3.8}$$

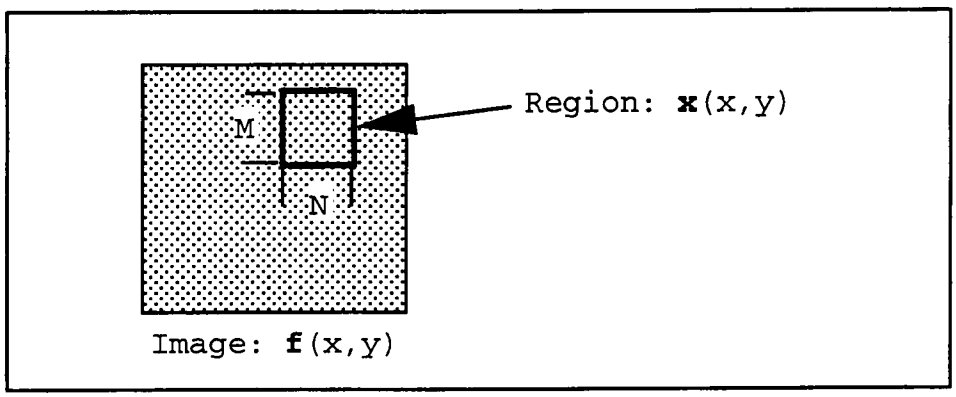


Figure 3.2: *A region of the image is used in deriving the MACE mask.*

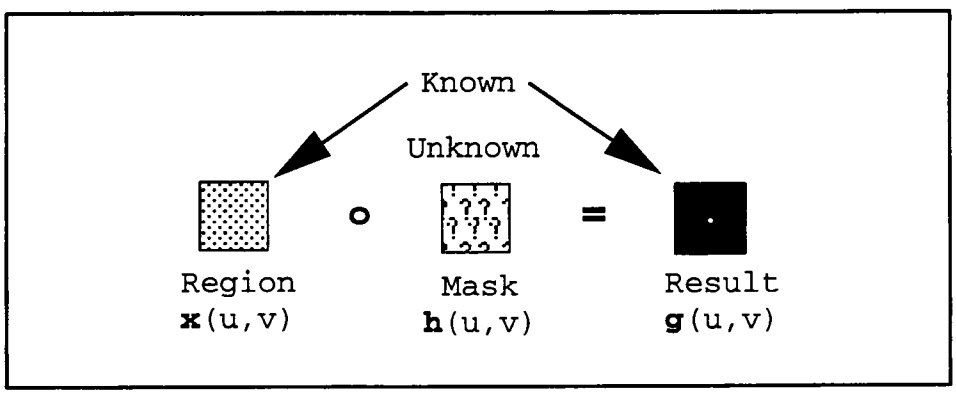


Figure 3.3: *Known and unknown regions in derivation of MACE filter. The mask must be derived from the region taken from the image and the desired result.*

The Fourier transform of this gives

$$G(u,v) = \frac{1}{MN} \tag{3.9}$$

where M and N are the width and height of the region. In this case the correlation will be represented in the frequency domain as

$$G(u,v) = X^*(u,v)H(u,v). \tag{3.10}$$

From this result a correlation filter can be derived that gives the desired effect. Substituting Equation 3.9 into Equation 3.10 and solving for $H^*(u,v)$ the MACE mask becomes

$$H^*(u,v) = \frac{1}{MNX(u,v)}. \tag{3.11}$$

The mask created emphasizes features of the image that are not prevalent. For example, since all the values of the image are greater than zero, the image will have a large DC component. The MACE filter will not emphasize this frequency. It will emphasize the frequencies that contain less energy. These frequencies will also be the frequencies that the images are most likely to not share. The problem with this mask is that it may emphasize one insignificant feature too greatly. This means that if another object has this frequency of energy at a higher value, the simple fact that that frequency exists may cause a false peak to occur.

Listed in Figure 3.4 are four samples of the creation and results of the MACE

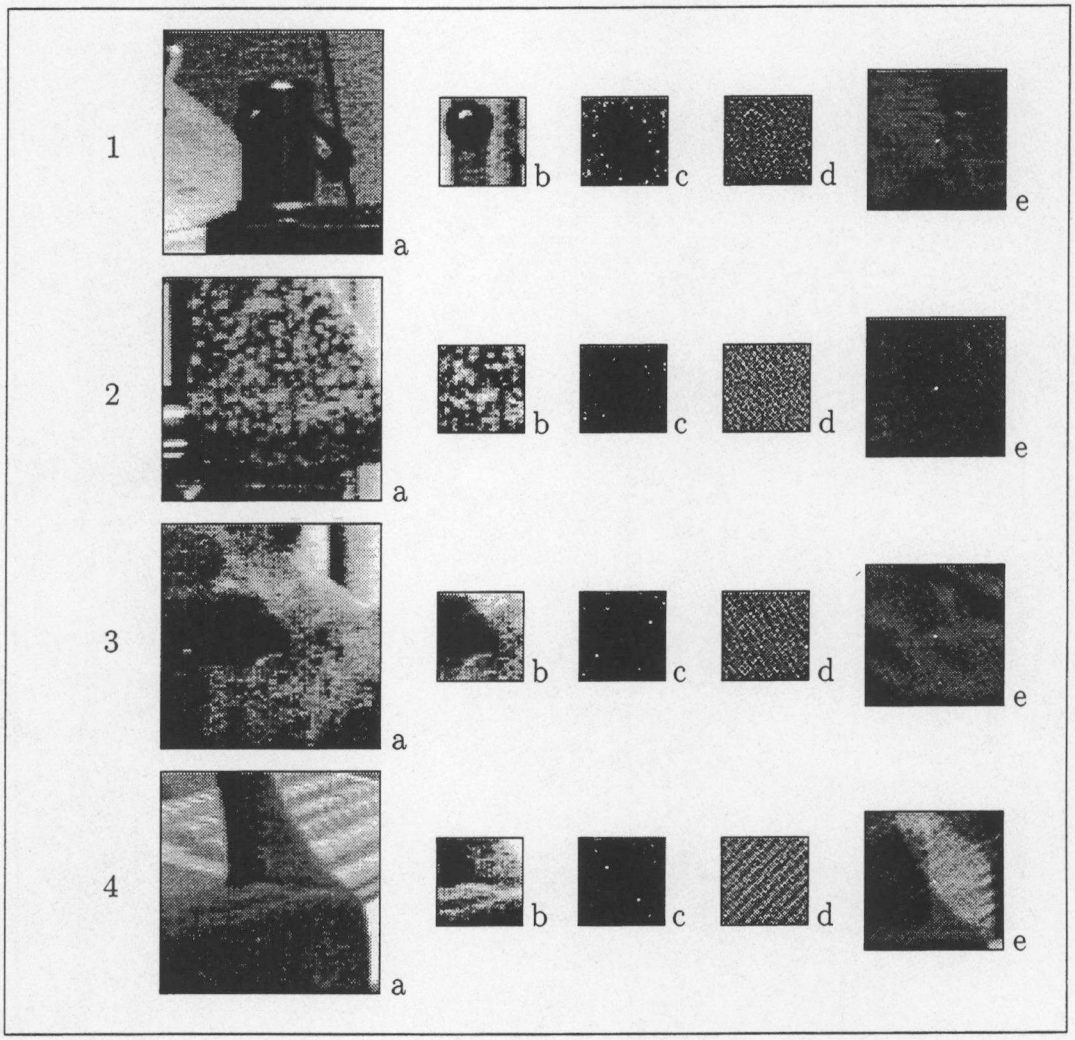


Figure 3.4: *MACE filter correlation. (a) Image to be searched. (b) Area to search for. (c) Fourier transform of mask created from* b. *(d) Correlation mask from* c. *(e) Result of correlation of* a *with* c.

filter. In these examples, the object that trains the mask is the same as the object which is to be found. In this case, this is the ideal result of the MACE filter. In a real tracking algorithm the images would not be the same. The object to be searched for (shown in frame b) is simply cut from the center of the area to be searched. The mask size is 32×32 pixels. The size of the image in Frame a is 82×82 pixels. The resulting image in Frame e is 51×51. The reason for these choices of sizes is given in Chapter 5. Frame a and Frame b were both histogram equalized before being displayed. This is the reason they may have differing intensities. All of the calculations including the mask creation and the correlation were done on images that were not histogram equalized. The correlation mask shown in Frame d has very little structure to it. It can be seen for the third and fourth series, there was a dominant frequency moving diagonally in the images. In all of these examples the result has a sharp peak in the center of the image. All of these images would correspond to a match.

3.3 Minimum Noise and Correlation Energy Filter

When considering the MACE filter, it emphasizes the frequencies that have low energies. For example, if an object has no energy at a certain frequency, the mask will have infinite energy at that frequency. This is because the MACE filter contains energy at each frequency that is inversely proportional to the energy at each frequency of the object that creates it. The MINACE filter uses the basic idea of the MACE

filter, but it places a maximum for the energy of each frequency in the filter. In this way, it still emphasizes the less prevalent frequencies, but it now emphasizes more of them. The phase is still preserved so that the location of the frequency will remain the same.

The equation for this filter is

$$H(u,v) = \begin{cases} \frac{1}{MNX^*(u,v)} & \text{if } \left|\frac{1}{MNX^*(u,v)}\right| < \alpha \\ \frac{\alpha|X(u,v)|}{X^*(u,v)} & \text{otherwise} \end{cases} . \tag{3.12}$$

The threshold is α which is the maximum value any frequency in the MINACE mask may have. If the value is larger than this threshold, it will be set to that value.

It should be noticed that in this equation that depending on the threshold and the number of frequencies that have values that exceed this threshold, the result of an exact match will be less than that of the MACE. In the MACE filter, the result of an exact match will always be the same value, one.

Listed in Figure 3.5 are four samples of the creation and results of the MINACE. The threshold used for the MINACE filter is 5.0×10^{-6}. The reason for the choice of this threshold will be explained in Chapter 4. These figures were created with the same parameters as Figure 3.4 with the exception that MINACE filter was used. The MINACE filter (Frame d) is more smooth than the MACE filter. The results of the MINACE were also more smooth than the results by using the MACE filter. The separation of the peak and the other values is less crisp. The advantage of using

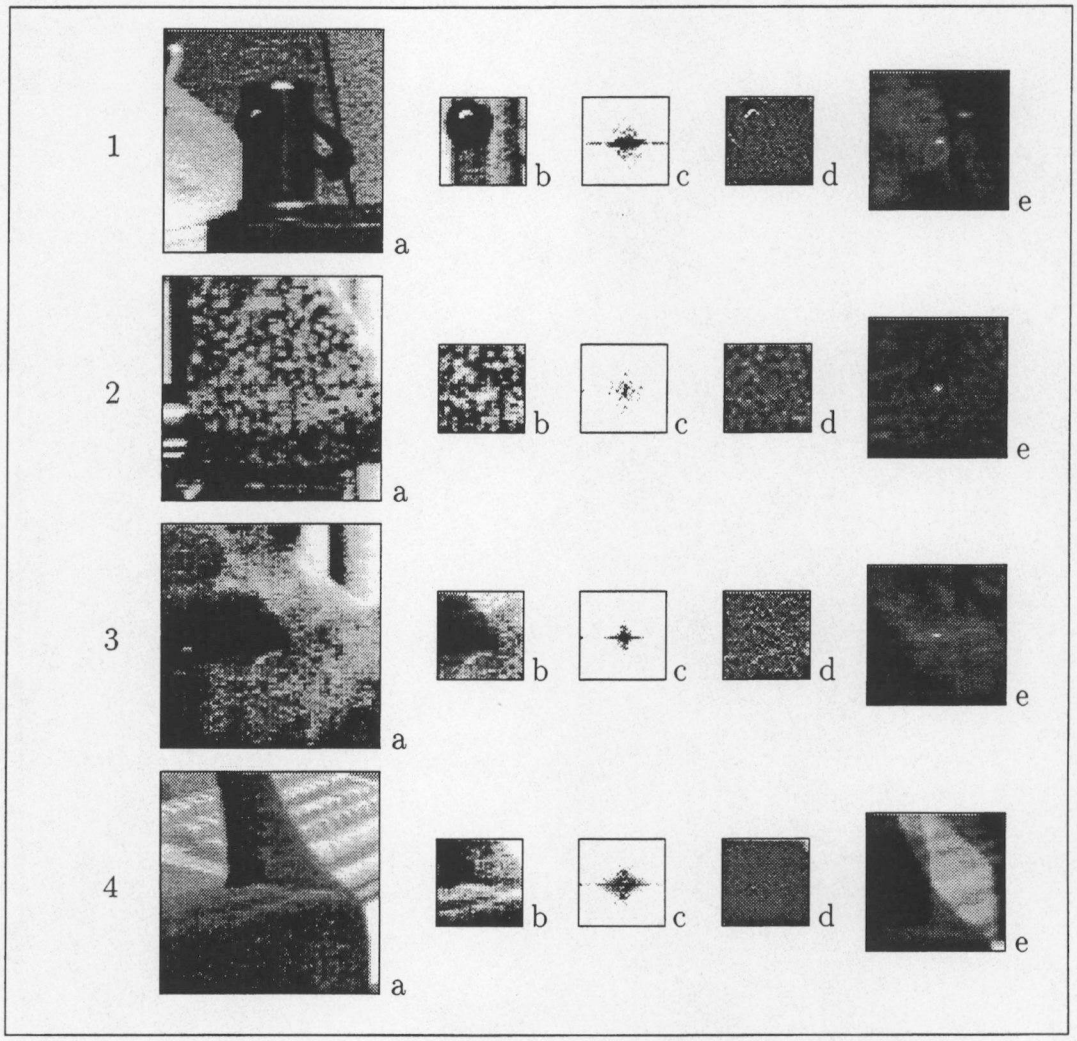


Figure 3.5: *MINACE filter correlation. (a) Image to be searched. (b) Area to search for. (c) Fourier transform of mask created from* b. *(d) Correlation mask from* c. *(e) Result of correlation of* a *with* c.

the MINACE filter is that the MINACE filter will be less sensitive to noise than the MACE filter [36]. Just like the MACE, all of these images would correspond to a match.

3.4 Normalized Cross Correlation

The third correspondence algorithm is the normalized cross correlation algorithm. In this algorithm, the mask and the image are both given zero mean. They are then correlated. Finally, this result is divided by the the square root of the mean by both the mask and the image. The mean of a region in an image will be defined as

$$A_f(x,y) = \frac{1}{MN} \sum_{\alpha=0}^{M-1} \sum_{\beta=0}^{N-1} f(x+\alpha, y+\beta). \tag{3.13}$$

where M is the width of the region, N is the length of the region, and f is the image. Using this function for the mean of a region in an image, normalized cross correlation can be represented as

$$g(x,y) = \sum_{\alpha=0}^{M-1} \sum_{\beta=0}^{N-1} \frac{(f(x+\alpha, y+\beta) - A_f(x,y)) \times (h(\alpha,\beta) - A_h(0,0))}{A_f(x,y) \times A_h(0,0)}. \tag{3.14}$$

Listed in Figure 3.6 are four samples of the creation and results of the normalized cross correlation filter. These figures were created with the same parameters as Figure 3.4 with the exception that normalized cross correlation filter was used. There is no correlation mask shown in these figures because the mask is the same

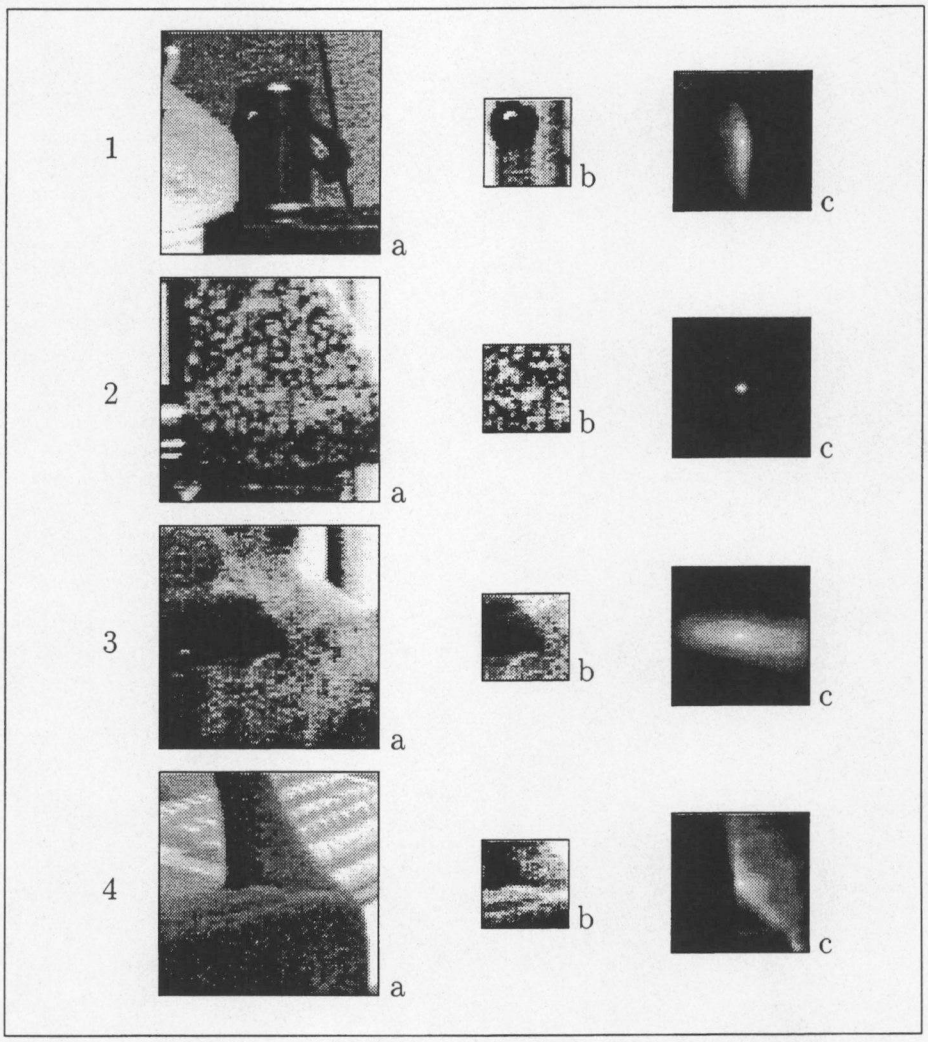


Figure 3.6: *Normalized cross correlation. (a) Image to be searched. (b) Area to search for. (c) Result of normalized cross correlation.*

as the training region except that it has the mean subtracted from it. The peak of these result of the normalized cross correlation is less separate from the surrounding values than that of either the MACE or the MINACE. There is more areas that do not have a high match in this filter than with the MACE or the MINACE. Like the MINACE and the MACE filters, the results of the normalized cross correlation would correspond to a match.

3.5 Sum of Squared Difference Correlation

The final region matching algorithm sum of squared difference correlation algorithm. As the name implies, the image and mask are subtracted, squared and this result is summed. This can be represented as

$$g(x,y) = \sum_{\alpha=0}^{M-1} \sum_{\beta=0}^{N-1} (f(x+\alpha, y+\beta) - h(\alpha,\beta))^2 \tag{3.15}$$

where M is the width of the mask and N is the height of the mask. Unlike the previous matching methods, the minimum value represents a match. An exact match would result in a value of zero. To make this mask act like the other masks the equation

$$g'(x,y) = \frac{\gamma MN}{\gamma MN + g(x,y)} \tag{3.16}$$

is used. The value given to γ changes how the values are scaled. For this thesis a value of 100 was used for gamma. This equation inverts and scales the results. All

values given in this paper correspond to this equation. In this inversion and scale, an exact match would correspond to a value of one.

Listed in Figure 3.7 are four samples of the creation and results of the sum of squared differences filter. These figures were created with the same parameters as Figure 3.4 with the exception that sum of square difference filter was used. There is no correlation mask shown in these figures because the mask is the same as the training object. The results of the sum of squared difference correlation closer match the results of the normalized cross correlation algorithm than the MACE or the MINACE algorithms. The regions near the peak have larger values than the regions farther away from the peak. This is different than the MACE and the MINACE where regions outside of the peak have similar level of brightness.

3.6 Summary

In this chapter four matching algorithms were introduced. The equations for each algorithm were given. Four examples of what the results of the matching algorithms would be were shown. These should give the reader a feel of how they operate. The MINACE and MACE filters tended to give peaks much greater than the surrounding values. The normalized cross correlation and the sum of squared differences algorithms have peaks that are closer to than the surrounding values. Unlike the MACE and MINACE, the normalized cross correlation and the sum of square differences algorithms tend to decay from the peaks. In the following chapter, these four masks will

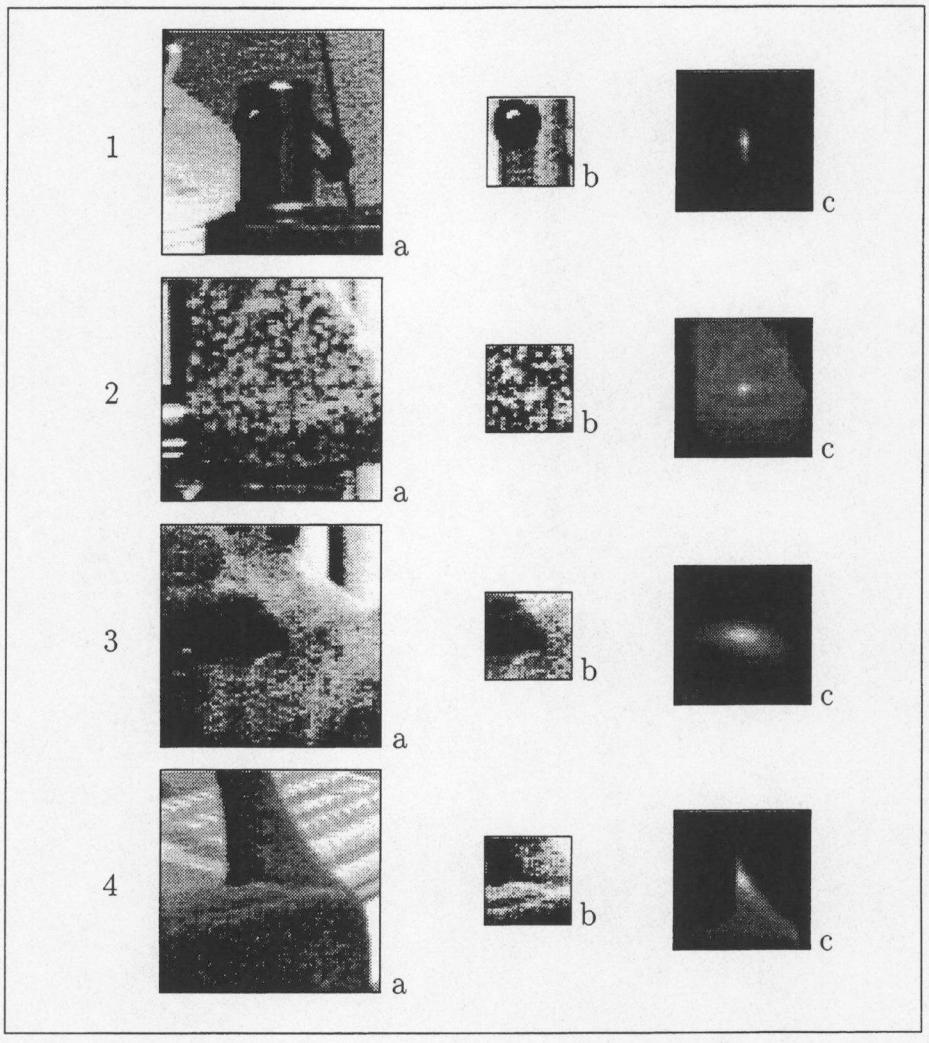


Figure 3.7: *Sum of squared difference correlation (a) Image to be searched. (b) Area to search for. (c) Result of sum of squared difference correlation.*

be compared for accuracy on any processor and implementation speed on a pipeline image processor. One of these will then be chosen to be implemented in the tracking algorithm described in Chapter 5.

Chapter 4

Numerical Analysis of Four Correlation Algorithms

In Chapter 2, tracking algorithms were discussed. Tracking by use of a correlation technique was decided. Chapter 3 introduced four different tracking algorithms. In this chapter, these algorithms will be tested for tracking capabilities where the object being tracked has in plane translation, out of plane z-translation, and out of plane rotation. The speed at which these algorithms can be implemented are also discussed. Finally at the end of this chapter, one correspondence algorithm will be chosen to be used in Chapter 5 as the main part of its tracking algorithm.

This chapter first specifies how the tests will be conducted. This gives the test images used, how the test points were specified, and what is required for a match. Next it shows the results of these tests for the MACE, MINACE, NCC, and the SSD correlation algorithms. Finally, the algorithms are compared for speed and accuracy.

4.1 Experimental Setup

In order to test the capabilities of the four correlation algorithms, a consistent testing situation had to be designed for all of the algorithms. The tests in this thesis used six series of image sequences. Since these were the same test sequences for each algorithm, the algorithms could be trained and search for the exact same region in an image. These image sequences are of real images in which the object in the image has been given a constrained motion. In these sequences, two test objects were either translated along the x axis, translated along the y axis, or rotated about an offset y axis. The initial location of the object being tracked was one half meter from the camera.

There were two different objects being tracked. The first was a vice. This was chosen because it contained many distinct lines and a textured surface. The second was a wooden rhino. This object had less features and a texture that was not as prevalent. These objects were common objects available in the laboratory. They were not targets that were specifically designed to be tracked. By choosing these objects, it was hoped that a tracking algorithm could be developed that could follow any common objects and not a target specifically designed to be tracked.

On each object there were five points selected to be tracked. These were selected because they had something unique about them. Usually this meant that they had both vertical and horizontal features or were located within the object but close to an edge in some way.

There was a series of six images for each motion. This series included the initial image and five motions. For the translation, the object translated four pixels horizontally in the image. There was a total distance of twenty pixels traveled in the five images. For the rotation, the object rotated two degrees. There was a total of ten degrees rotated. For the z translation of the object, the object was initially one half meter away. The object was moved one half centimeter closer each motion. This means that the object will have traveled 2.5cm toward the camera. This is five percent closer to the camera than the original image. All of the series and the points being tracked are shown in Figure 4.1 through Figure 4.6.

For these sets of experiments, there were five points per series and six series of points being tracked. This gave a total of thirty points being tracked. For each of these thirty points, there were six images in a series. This means that one hundred eighty total data values would be available to test each correlation algorithm at each mask size.

The choice of mask size was influenced by the Fourier transform algorithm. Both the MACE and the MINACE filters would require the use of the Fourier transform. This thesis implemented the Fourier transform by using the fast Fourier transform (FFT) [35] because of the speed at which it implemented the Fourier transform. Because a characteristic of the FFT was that it required the width and height of the image to be a power of two, the mask had to have a width and height being a power of two. Rectangular masks were considered, but initial experiments showed

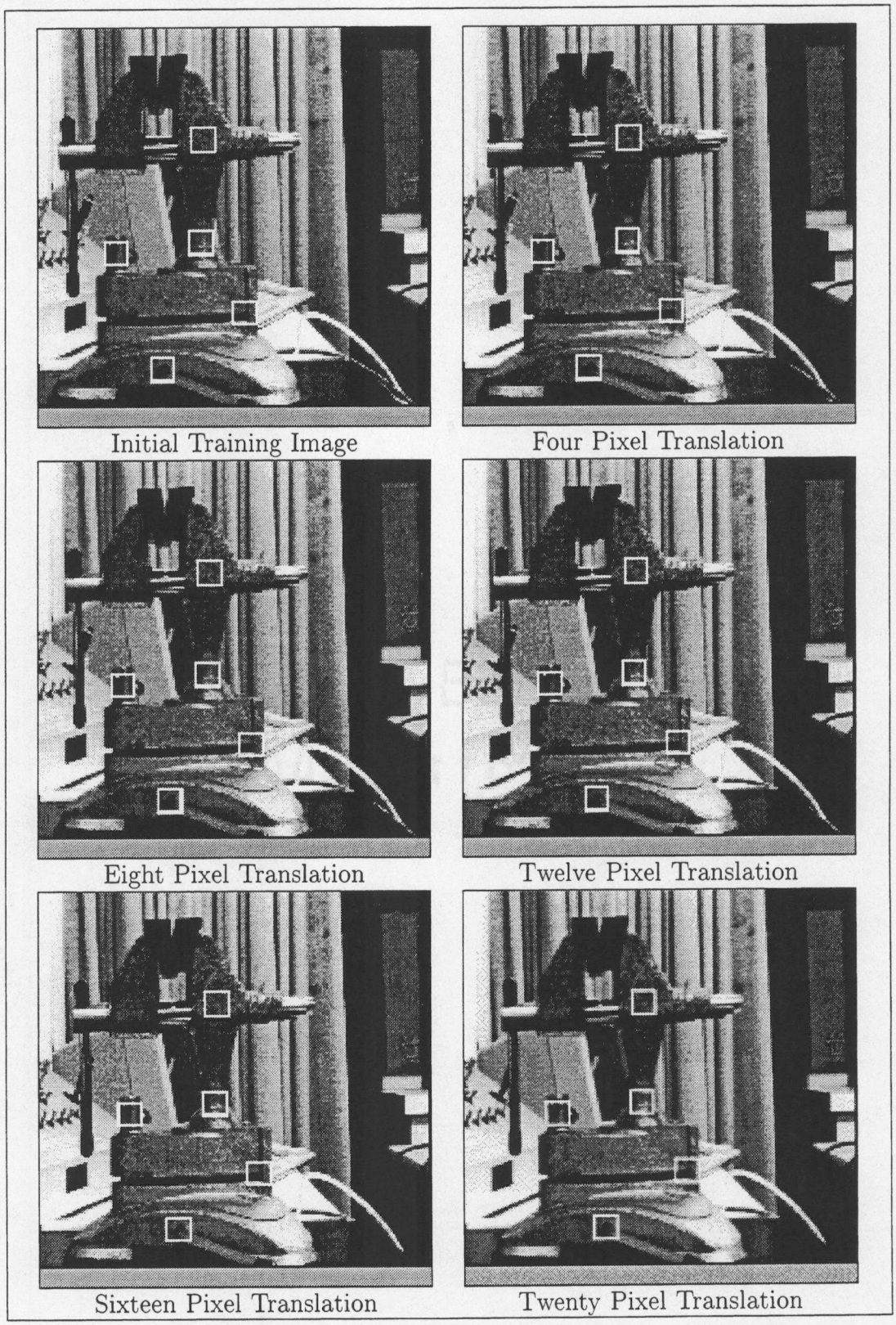
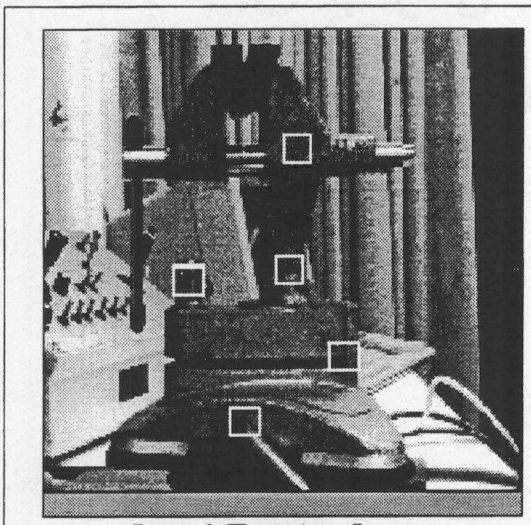
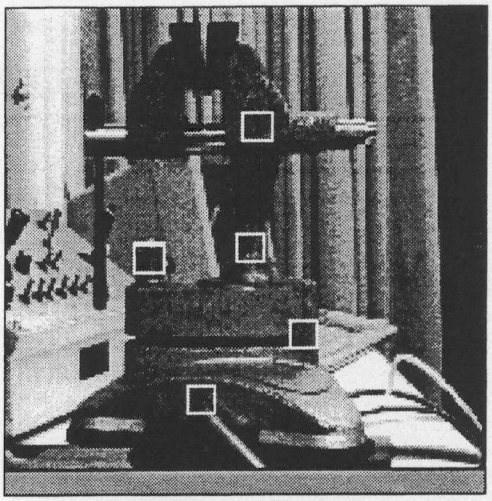


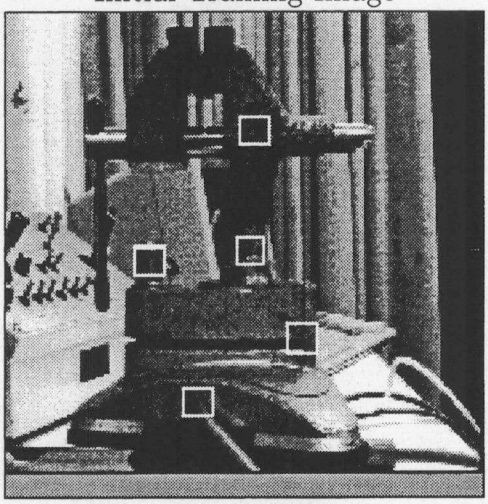
Figure 4.1: *Vice translation series.*



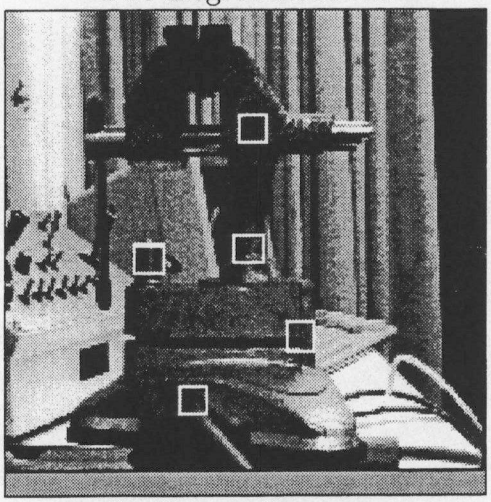
Initial Training Image



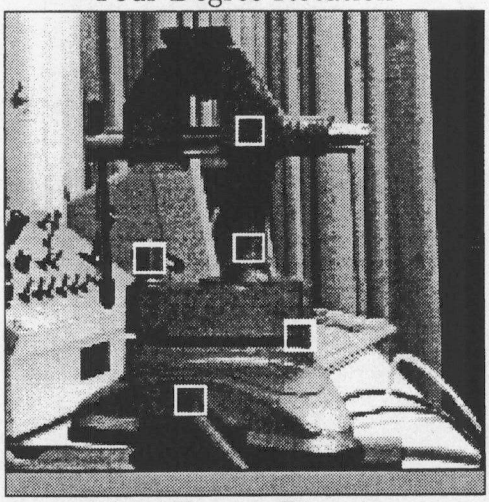
Two Degree Rotation



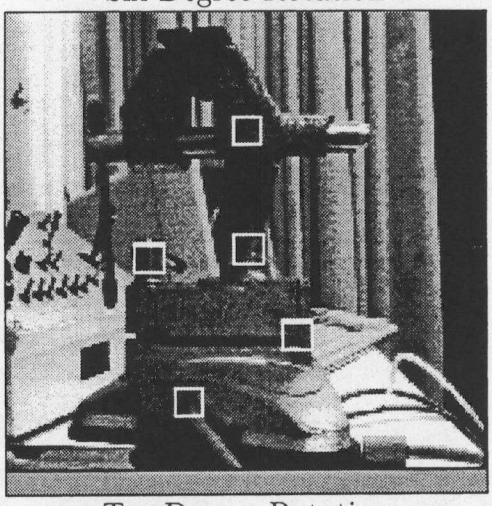
Four Degree Rotation



Six Degree Rotation



Eight Degree Rotation



Ten Degree Rotation

Figure 4.2: *Vice rotation series.*

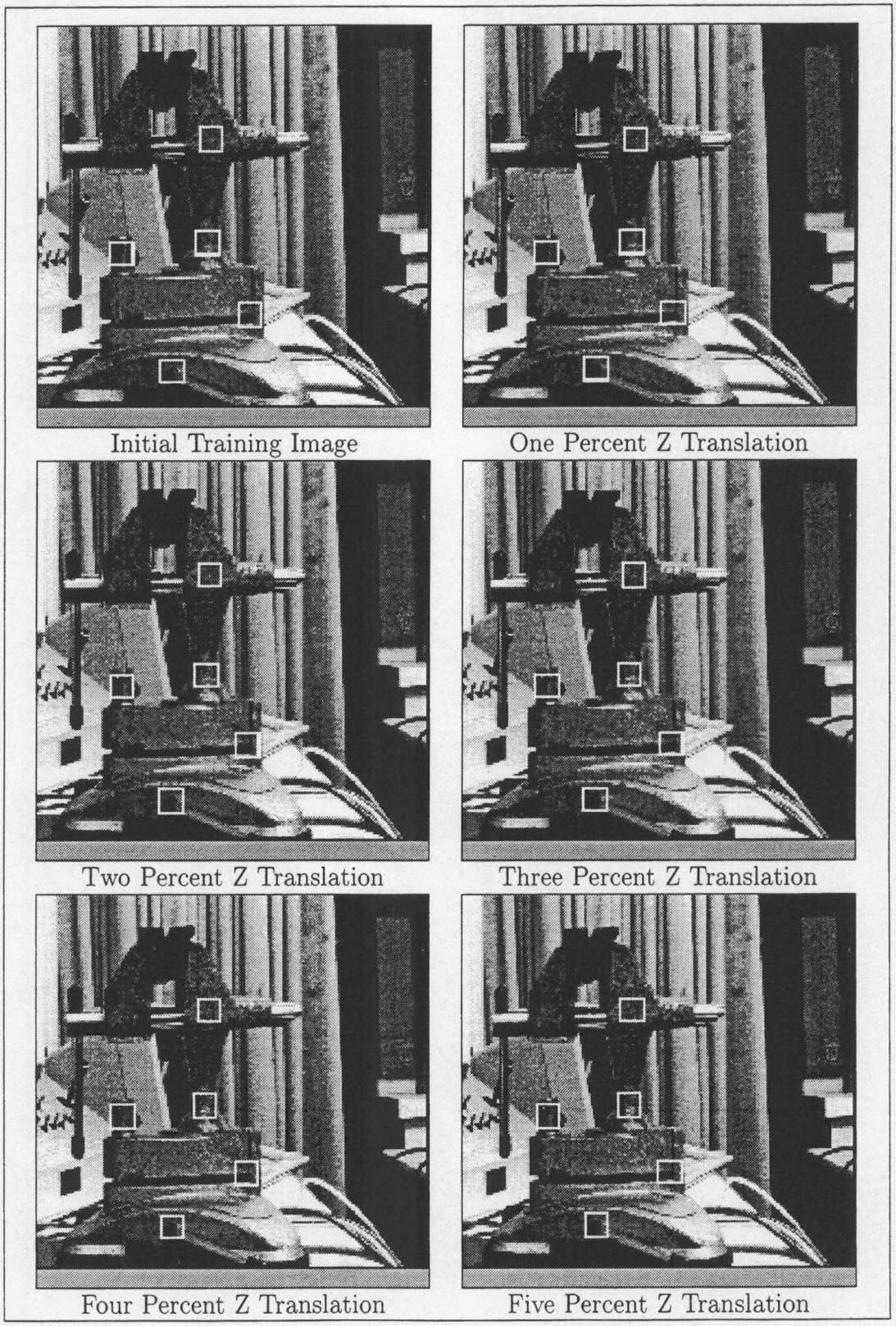


Figure 4.3: *Vice z-translation series.*

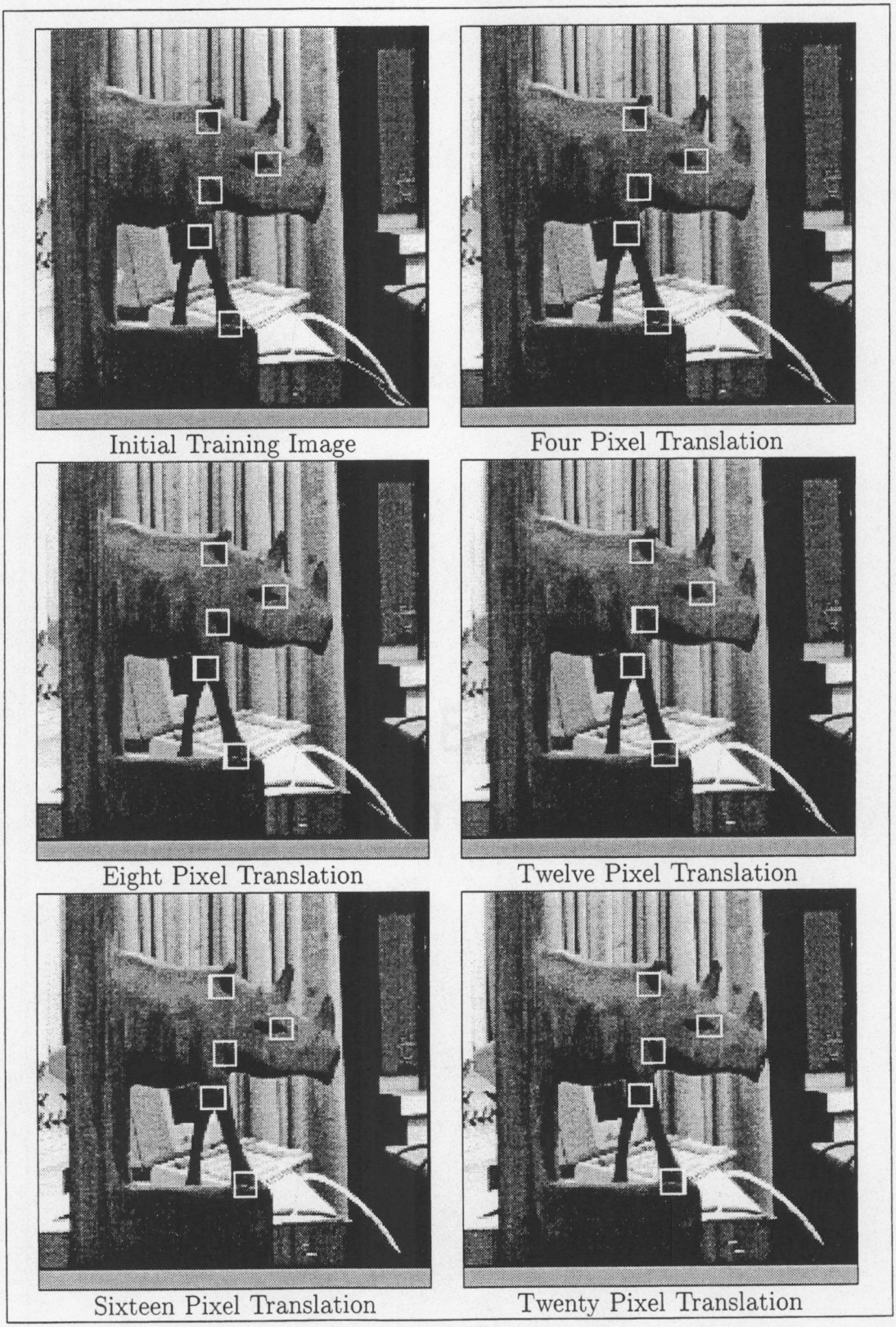
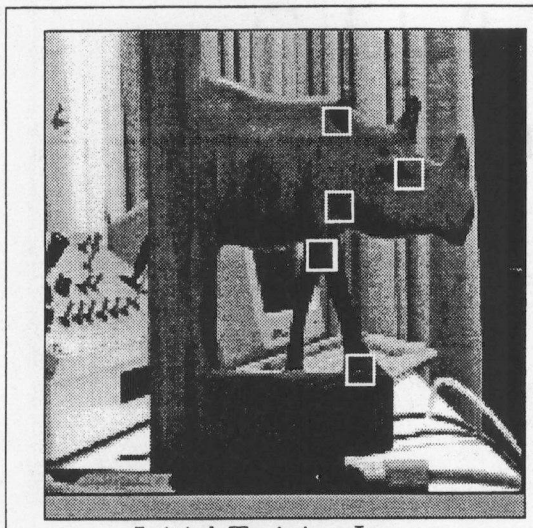
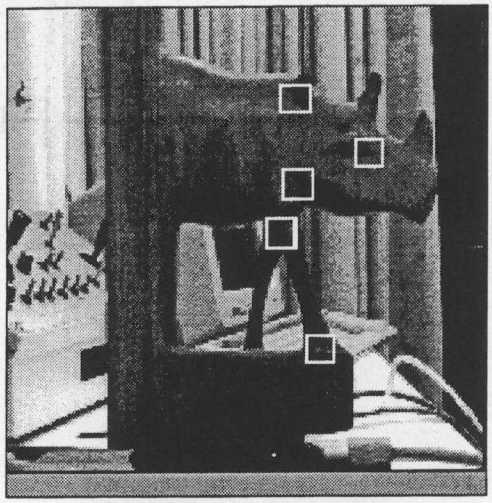


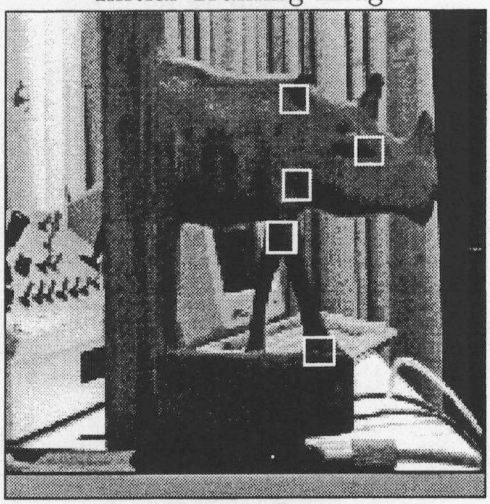
Figure 4.4: *Rhino translation series.*



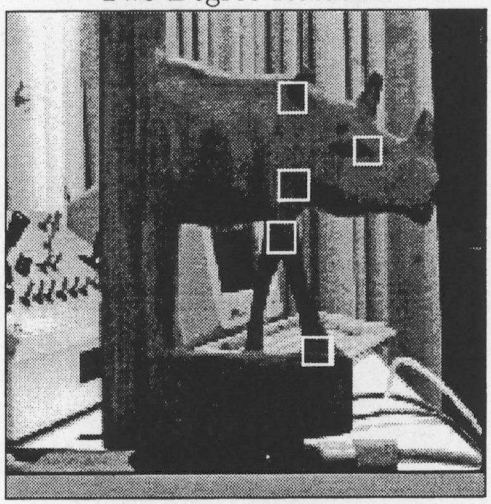
Initial Training Image



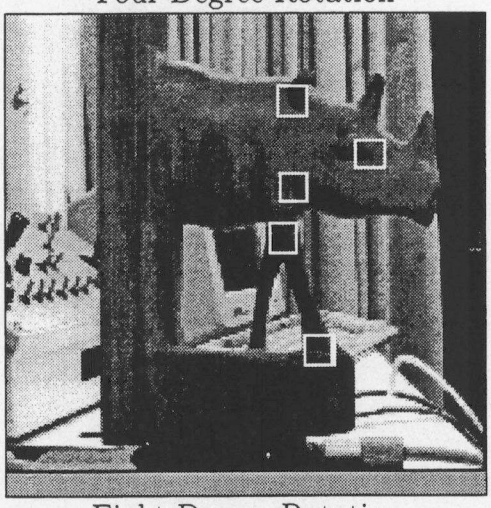
Two Degree Rotation



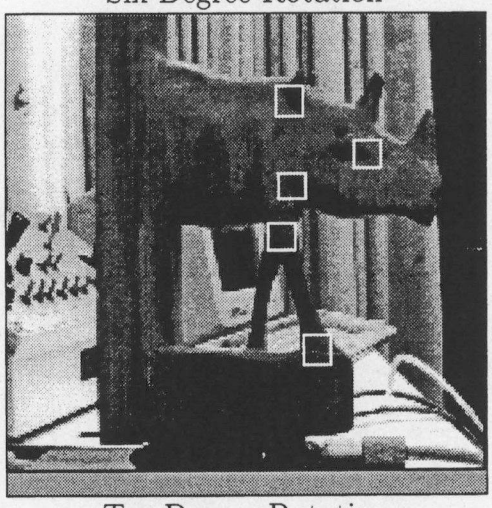
Four Degree Rotation



Six Degree Rotation



Eight Degree Rotation



Ten Degree Rotation

Figure 4.5: *Rhino rotation series.*

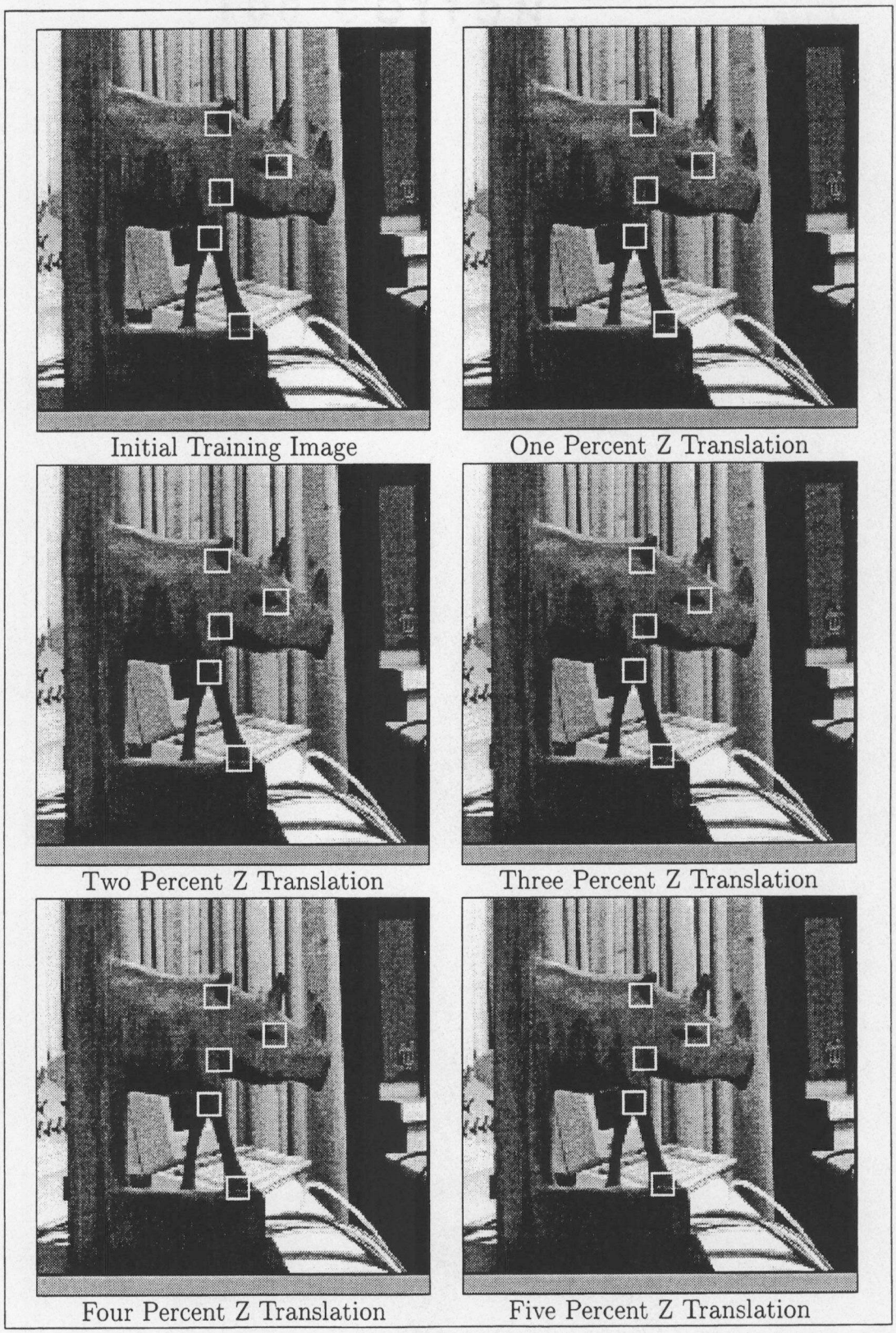


Figure 4.6: *Rhino z-translation series.*

that these masks worked not much better than a square mask having the same size as the shorter side. The maximum size of the mask was dictated by the size of the object in the image. A search mask the same size of the image would not be practical since the correlation would only end up with a single match value. As the mask became larger, the computing time would increase. For these reasons, mask sizes of 32×32, 16×16, and 8×8 were used.

This experiments is supposed to test the capabilities of these correlation filters at finding where the region has moved in the following image. The irony is that in order to test these results, the location of the match regions must be known. In this experiment, they can be estimated, but not known. The researcher had to choose manually where a correct match in a following frame would be located. The criteria for an exact match was then defined to be any point an euclidean distance less than three pixels from the manually selected object location. This gives some area for error in the researchers locating of the regions in each series of images.

4.2 Minimum Average Correlation Energy Filter

Of the four correlation algorithms, the MACE had results that were the worse. Shown in Figures 4.7, 4.8, and 4.9 are the results of translation, out of plane translation, and out of plane rotation respectively. Even with no motion (zero pixels translation, 100% distance from camera, or zero degree rotation) and a 32×32 mask size, the MACE had a 10% incorrect matches. After this, the MACE performance quickly

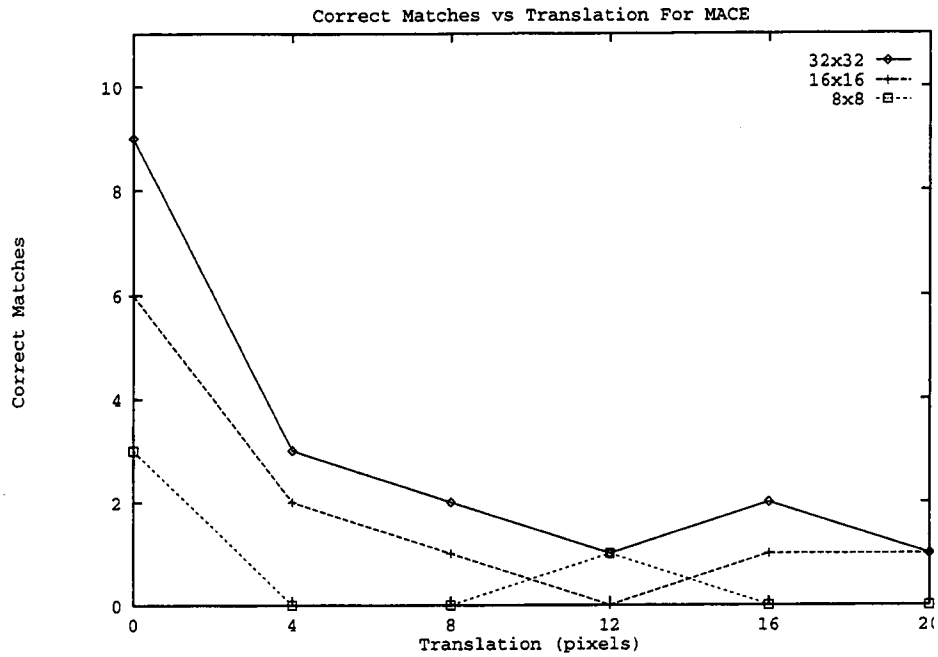


Figure 4.7: *Effect of horizontal translation on MACE correspondence.*

degraded. Typically, the number of correct matches was less than 40%. Surprisingly, the best performance of the MACE filter was with the out of plane translation. This performed from 40% to 20%. It would be expected that a simple motion like in plane translation would be the easiest to track. With the MACE for this data set this was not true. The results for translation were equal to or less than 30% for any in plane translation.

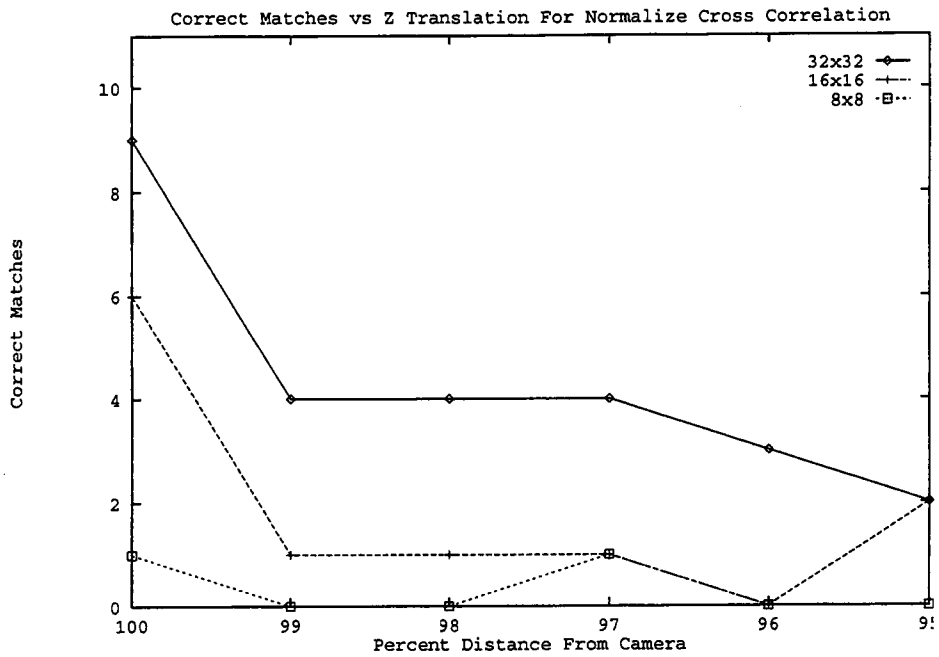


Figure 4.8: *Effect of z-translation on MACE correspondence.*

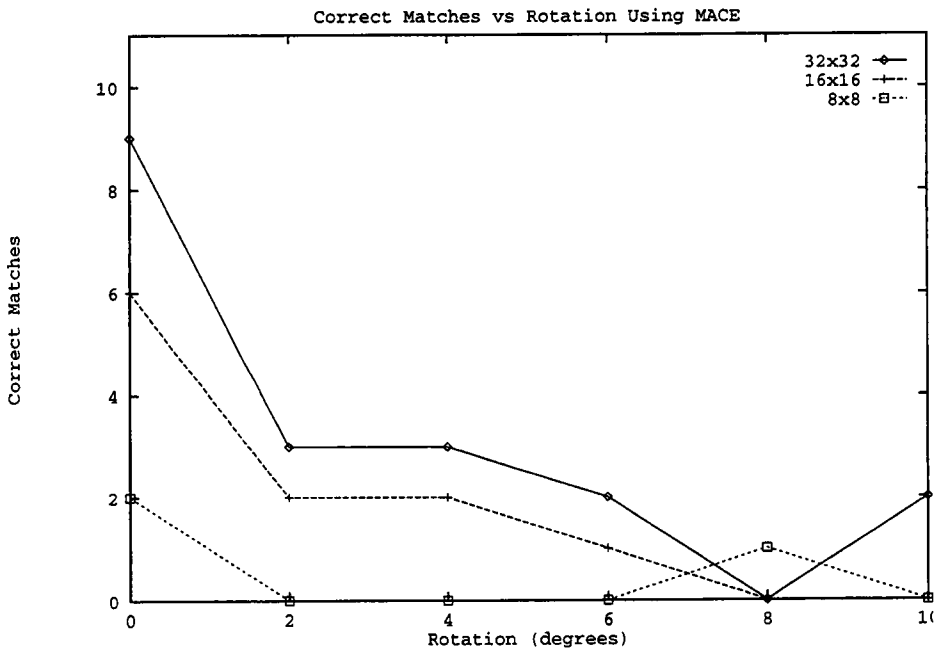


Figure 4.9: *Effect of rotation on MACE correspondence.*

4.3 Minimum Noise and Average Correlation Energy Filter

The disadvantage of the MINACE algorithm is that there is a threshold that must be selected. There may be some way to look at the statistics of the image an the background for selecting the MINACE threshold. This is a complex problem. Much time could be spent finding the way to select the threshold and also calculating the threshold. Some initial experimentation showed that a single threshold for each mask size could give results that were satisfactory. For this reason, a single threshold for each mask size was chosen and used for the remainder of this thesis.

The data series were used to select the thresholds used in all of the rest of the experiments in this thesis. For each series, the five points in the initial series was selected. These were used as training masks for searching for the points in the next image. There were one hundred eighty possible matches. The threshold vs correct matches is graphed in Figure 4.10.

For all three mask sizes, there was a maximum value. The values decrease nearly monotonically to either side of this maximum. This maximum was selected to be the threshold for the creation of the *minimace* mask for the rest of the experiments. The thresholds for mask sizes of $32, 16$, and 8 were 5.0×10^{-6}, 3.0×10^{-7}, and 1.0×10^{-4} respectively.

Shown in Figures 4.11, 4.12, and 4.13 are the results of translation, out of plane translation, and out of plane rotation respectively. The performance of the MINACE

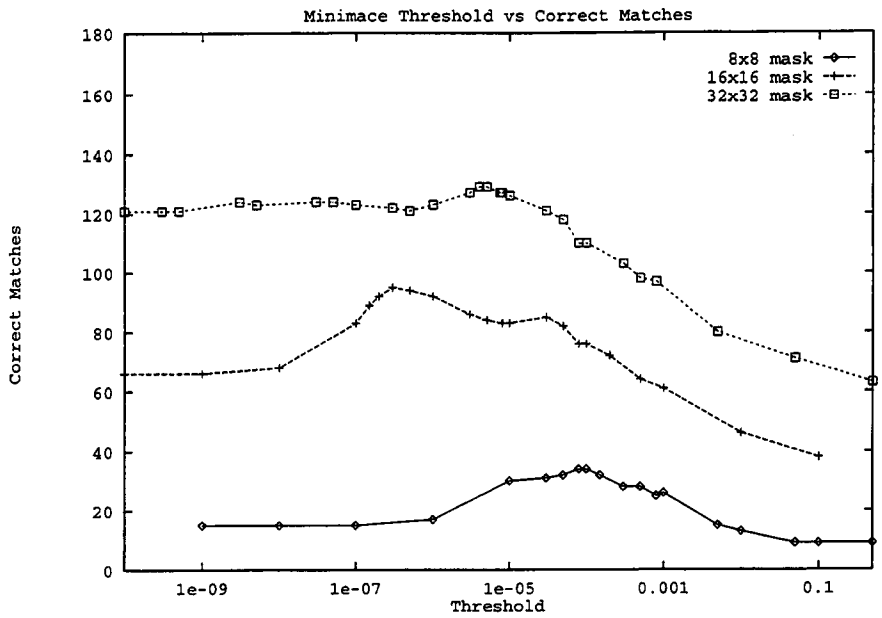


Figure 4.10: *Selection of MINACE threshold.*

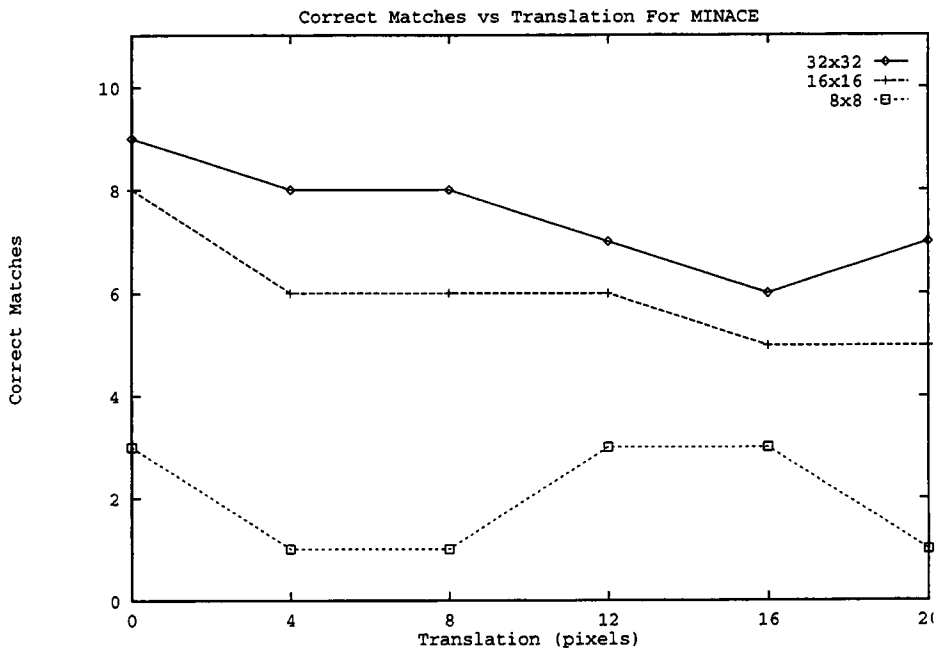


Figure 4.11: *Effect of horizontal translation on MINACE correspondence.*

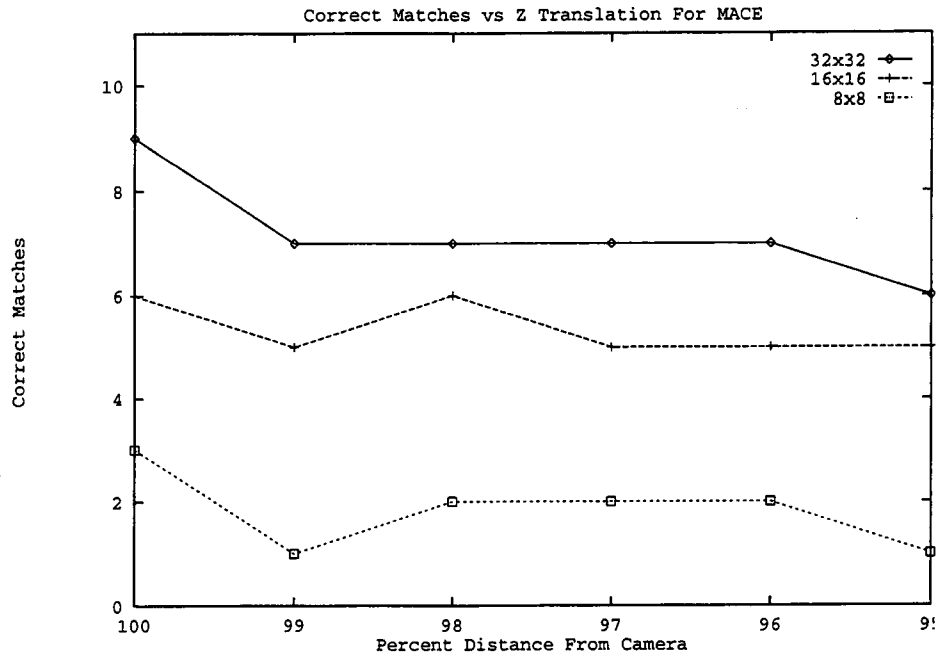


Figure 4.12: *Effect of z-translation on MINACE correspondence.*

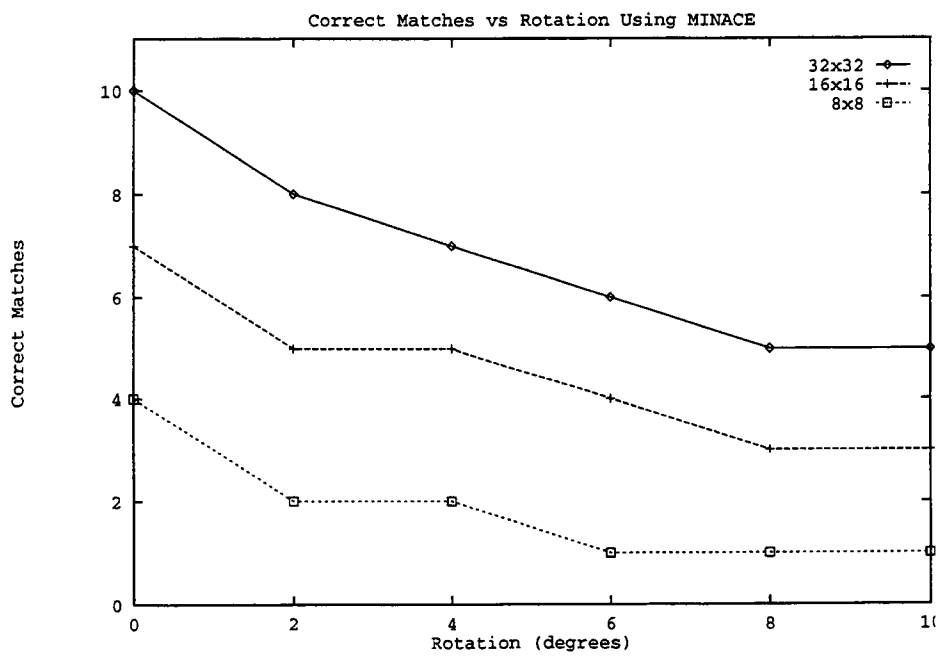


Figure 4.13: *Effect of rotation on MINACE correspondence.*

was much better than that of the MACE. For translation the MINACE has results of 60% or better for mask sizes of 32×32 and 16×16. For out of plane translation, the MINACE had 60% or better match rate on the 32×32 mask size. Rotation of the object caused the MINACE to have a constant drop in matches. For the 32×32 mask, the match rate decreased from 100% to 50%. These results have low match ratings, but are high enough to track an object.

4.4 Normalized Cross Correlation Filter

Shown in Figures 4.14, 4.15, and 4.16 are the results of translation, out of plane translation, and out of plane rotation respectively. The results for the normalized cross correlation filter were much better than the preceding two. At a mask size of 32×32 there were 100% correct matches for in plane translation and out of plane translation. The out of plane rotation had some effect with the correct matches dropping to 90% for 6, 8, and 10 degrees. Even for a mask size of 8×8 the worse correct matches gotten were 60%. A filter with a 8×8 mask size performed better than either the MINACE or MACE filters.

4.5 Sum of Square Difference Filter

Shown in Figures 4.17, 4.18, and 4.19 are the results of translation, out of plane translation, and out of plane rotation respectively. Like the NCC, the sum of square

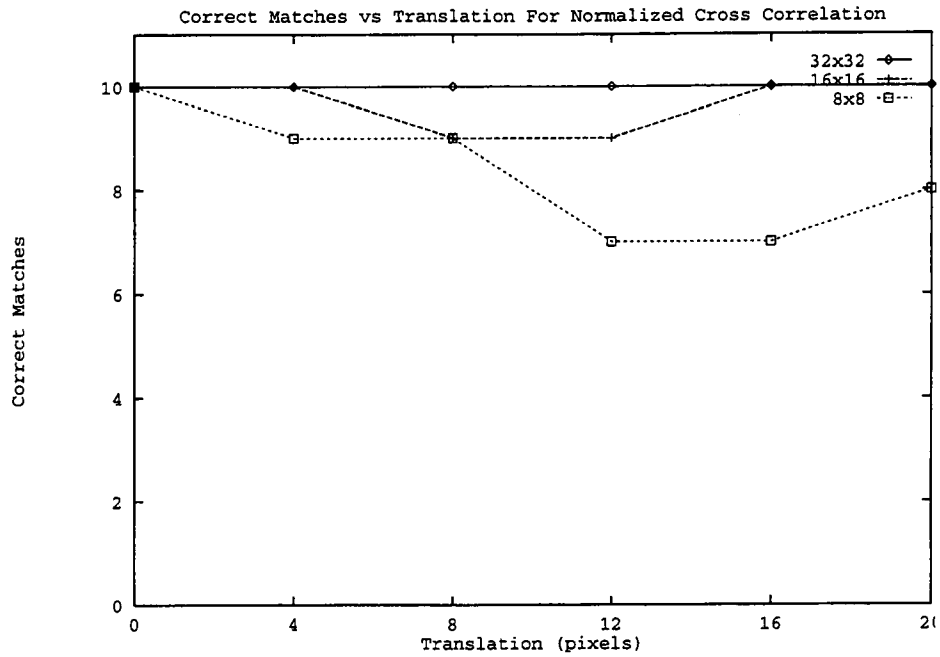


Figure 4.14: *Effect of horizontal translation on normalized cross correlation correspondence.*

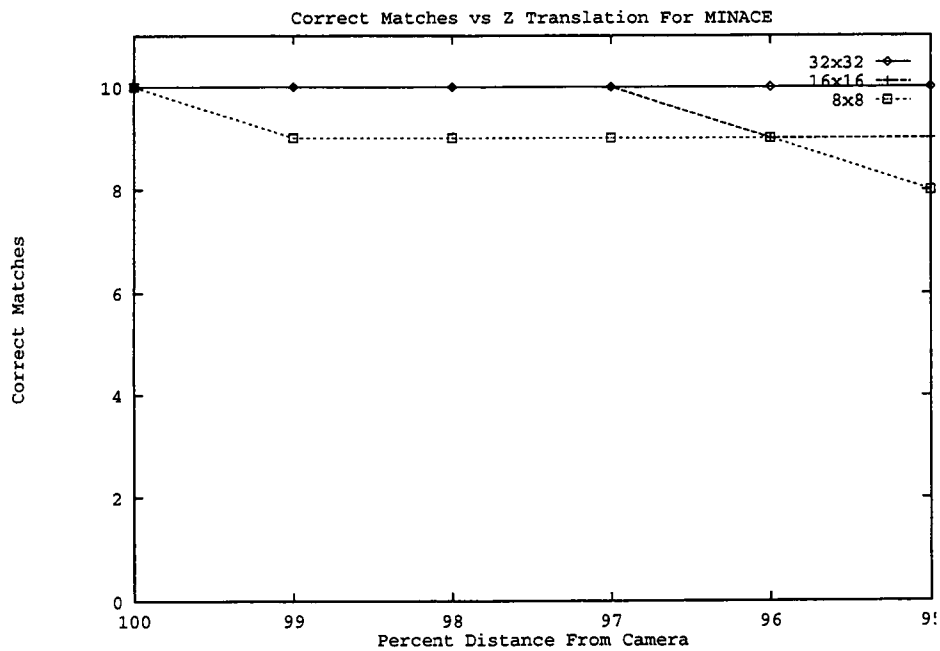


Figure 4.15: *Effect of z-translation on normalized cross correlation correspondence.*

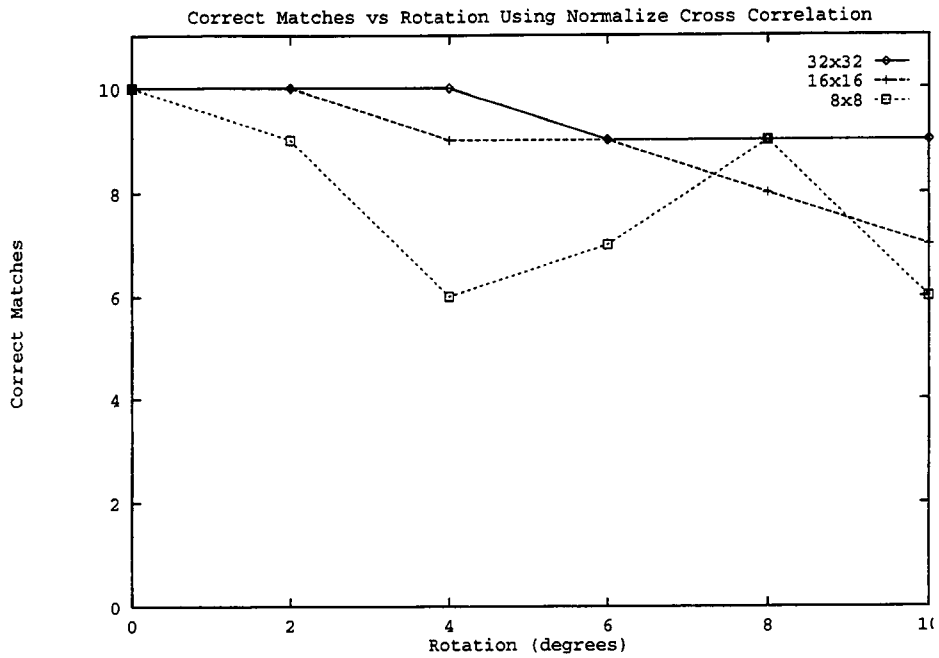


Figure 4.16: *Effect of rotation on normalized cross correlation correspondence.*

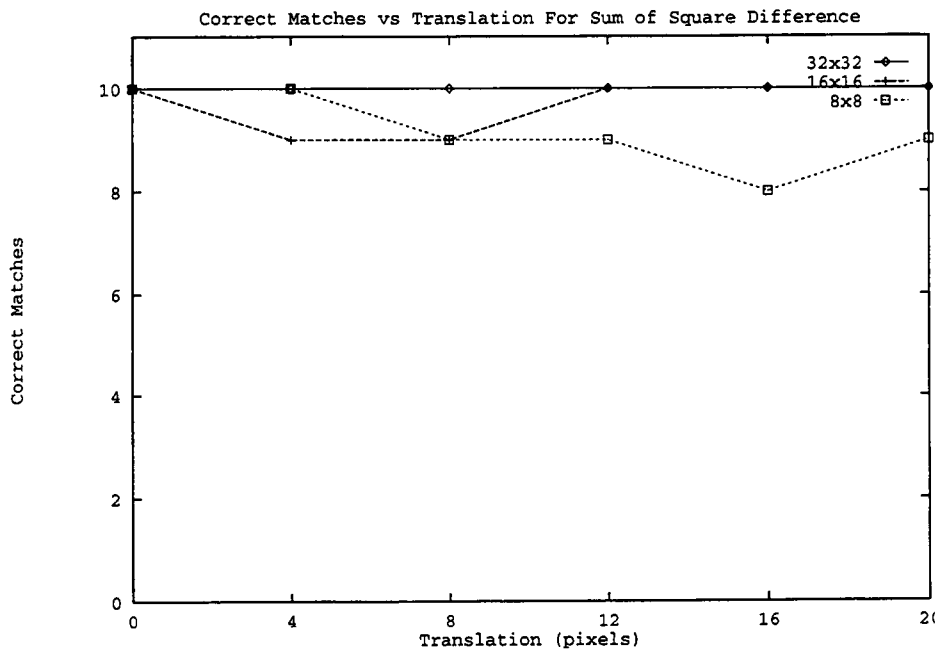


Figure 4.17: *Effect of horizontal translation on SSD correspondence.*

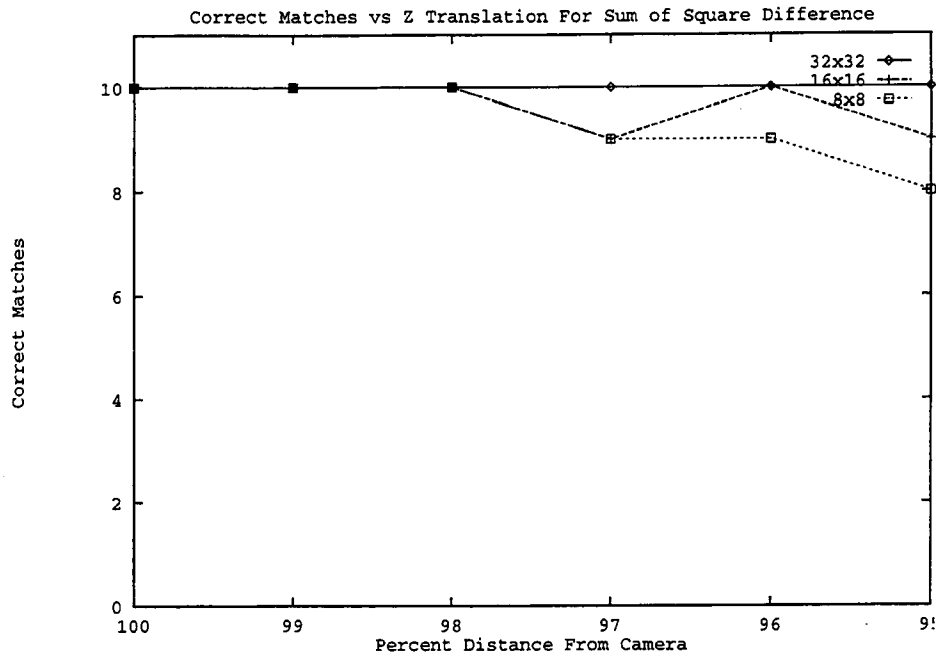


Figure 4.18: *Effect of z-translation on SSD correspondence.*

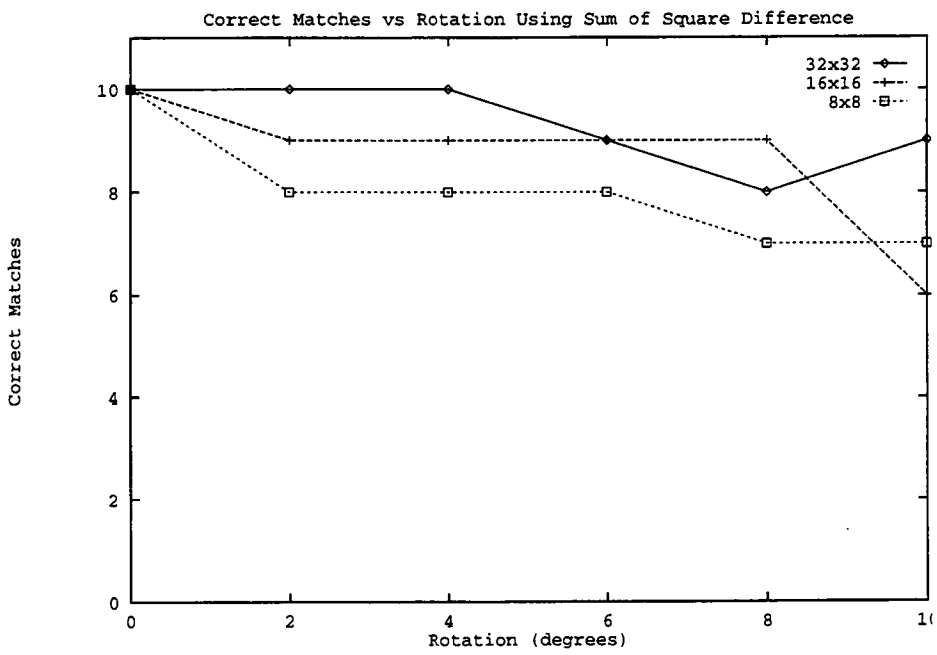


Figure 4.19: *Effect of rotation on SSD correspondence.*

differences filter had results much better than the MACE or the MINACE. The results of the SSD were similar to that of the NCC filter. Like the NCC, the SSD had 100% correct matches for both of the translations. For rotation, the results dropped to 80%.

4.6 Calculation Time of the Correspondence Algorithms

There is a second factor other than the capabilities of the algorithm to correctly match a region in a preceding image. The second factor is the speed at which the algorithm can be implemented. The focus of this thesis was to implement a tracking algorithm. If the tracking algorithm takes two minutes to locate the object in the preceding image, the object being tracked must be moving very slowly. This might be good for tracking lava flow but most applications would require a faster update rate.

The target system for the tracking algorithm was a Motorola 68030 processor with a Datacube MV20 pipeline processor. Thus all timing tests must use this equipment in the calculations. This is exactly what was done in generating Table 4.1. Table 4.1 calculated the training and run time for the four correspondence algorithms. The "training" time is the time taken from when the mask is cut from the image to the time it is ready to be placed in the correspondence algorithm. The "run" time is the time for the correspondence algorithm to locate the object in the image. Each

Table 4.1: *"Training" and "run" time for the four correlation algorithms. The results have* 51×51 *values. There are three mask sizes considered with widths and heights of 32, 16, and 8.*

	32×32		16×16		8×8	
	Train	**Run**	**Train**	**Run**	**Train**	**Run**
MACE	366ms	11200ms	75ms	11200ms	15.6ms	2433ms
MACE*	366ms	43ms	75ms	34ms	15.6ms	32ms
MINACE	376ms	11200ms	78ms	11200ms	16.7ms	2433ms
MINACE*	376ms	43ms	78ms	34ms	16.7ms	32ms
NCC	0ms	37033ms	0ms	11450ms	0ms	3066ms
SSD	0ms	23566ms	0ms	5950ms	0ms	1533ms
SSD*	0ms	5900ms	0ms	3600ms	0ms	2900ms

*When implemented using Datacube MV20 pipeline architecture.

algorithm would have a result that had 51×51 values.

There are several items of interest that should be noted about this graph. The first is that the "run" times for the MACE and the MINACE algorithms are exactly the same. This because after the mask is created, the MACE and MINACE algorithms only have to use correlation to find the object. This means that they use exactly the same correlation algorithm.

Next, it might be noticed that the MACE and MINACE "training" times for both the 32×32 mask size and the 16×16 mask size require the same amount of time to calculate when implemented without the Datacube pipeline architecture. This is because the correlation was done by taking the FFT of the objects, multiplying them, and then taking the inverse FFT. When using the 32×32 mask size, the dimensions of the object being correlated are 82×82 $(32 + 51 - 1)$. When using the 16×16 mask size, the dimensions are 56×56 $(16 + 51 - 1)$. In implementing the FFT, the size of

the object being correlated must be of the next value having a power of two. For the 32×32 and 16×16 mask sizes, this would be 128×128.

Also, it might be noticed that for the SSD and NCC algorithms, the creation time is $0ms$. This is because there is no preparation time in creating the mask. The mask cut from the image is the mask used in the correlation algorithm.

When looking at the time to calculate the SSD with and without the Datacube pipeline architecture, a curious event happens. The Datacube pipeline architecture takes less time to calculate the SSD when using a 32×32 mask size, but takes more time when doing the 8×8 mask size. This is because the Datacube pipeline architecture has a minimum set up time. When using the larger masks, this time was insignificant compared to the actual calculation time. When calculating the 8×8 mask, this set up time became a significant factor in the calculations.

The NCC was not implemented on the pipeline architecture. The first reason for this is that the NCC uses many floating point calculations. The pipeline processor only uses integer math. Secondly, the NCC could not be implemented with simple pipes. It is a very complex algorithm to implement on pipeline architecture. The researcher is confident that even if someone was able to find a way to implement the NCC on the Datacube pipeline architecture, it would not be calculated any faster than the SSD algorithm.

Table 4.2: *Summary of overall mask performance. This table summarizes the overall performance of the masks by giving the percentage correct matches comparing four correspondence algorithms having three different mask sizes.*

	32×32	16×16	8×8
MACE	35.0%	18.3%	5.0%
MINACE	71.7%	57.7%	18.8%
NCC	98.3%	93.9%	83.8%
SSD	97.8%	93.9%	88.3%

4.7 Selection of Correlation Algorithm for Tracking

There are two main factors in that should be considered when choosing a correspondence algorithm for a tracking algorithm. The first is obvious. A correspondence algorithm must be able to find the occurance of an image region in the following image. If the algorithm is unable to find the region in the following image, it will be of no use in tracking the object. This was examined for the four algorithms in the first four sections. The second factor is the speed at which an algorithm can be implemented. This was examined in the preceding section.

Table 4.2 summarizes the results of the previous experimentation. It can be seen from this table that the NCC and SSD algorithms performed equally well. When using a 32×32 mask size, they matched nearly 100% correctly. Even with a 8×8 mask size, they did better than either the MACE or the MINACE with a 32×32 mask size. If the tracking capabilities of the correlation mask were the only criterion, it would be obvious that the only choice would either be the NCC or the SSD at a 32×32 mask size.

Using the reliability factor in choosing a correspondence algorithm, the MACE can be immediately thrown out. The best match rate was 35% which was much too low to be considered. This same criteria can be used to remove the 16×16 and 8×8 mask sizes of the MINACE. The speed factor can be used to remove the NCC and SSD algorithms with mask sizes of 32×32 and 16×16.

This only left the SSD at 8×8, the NCC at 8×8 and the MINACE at 32×32. The NCC can be removed because it performs equally well as the SSD, but takes twice the time to calculate. This only left the MINACE and the SSD. The MINACE was chosen for this thesis because the calculation time was orders of magnitude better than the SSD and approached that of frame rate.

4.8 Summary

In this chapter, the capabilities of four correspondence algorithms were examined. First the abilities of the algorithms of correctly matching a region when an object was subjected to in plane translation, out of plane translation, or out of plane rotation. Next the calculation time of each of these algorithms on a target system consisting of a Motorola 68030 processor and a Datacube MV20 processor was considered. Finally, using the quality of the correspondence algorithm and the speed of the algorithm, the MINACE was chosen to be the algorithm used in this thesis.

Chapter 5

Description, Implementation, and Evaluation of a Real-Time Correlation-Based Tracking Algorithm

All of the preceding chapters were presented in order to decide a tracking algorithm. In Chapter 2, seven different tracking algorithms were described. From this information, the capabilities of each algorithm could be explored. In Chapter 3 four correlation algorithms were introduced. In Chapter 4 these algorithms were compared. This chapter is the culmination of this information of the previous chapters. This is the chapter which uses the information in those chapters and designs and implements a tracking algorithm.

This chapter is divided into four main sections. Each of the first three sections is more specific about the tracking algorithm than the previous. In the first section, the general tracking algorithm is given. This general tracking algorithm is not targeted for any hardware platform or correspondence algorithm. Instead, it is a template

which any hardware platform and correspondence algorithm can be expanded upon. The next section describes parameters used in the tracking algorithm implemented in this thesis. These parameters include mask size, update rate, search area, and how the object location was updated. In the third section, the details of how this algorithm was implemented on the hardware platform are given. Included in this section is a description of the hardware platform and how this hardware was utilized in the tracking algorithm. The final section evaluates the performance of the tracking algorithm. This campares the consistancy of the tracking algorithm in two ways.

5.1 Generalized Tracking Algorithm

The tracking algorithm using a correspondence algorithm has three parts. Show in Figure 5.1 is an overview of the tracking algorithm. The first is the initialization of the correspondence mask. This is where a training area is selected from the initial image. The second part is finding the mask in the next image and updating the location of the object in the image. The third part which runs concurrently with the second is cutting a new region from the image and comparing the result from this to the result of the initial mask. The last two parts continue until either there is no need to continue tracking the object, or the algorithm is unable to track the object.

The first part of tracking the object is selecting a region of the object to track. This can be selected automatically by the computer, or manually by the operator. The selection criteria of the region may depend on characteristics of the correspondence

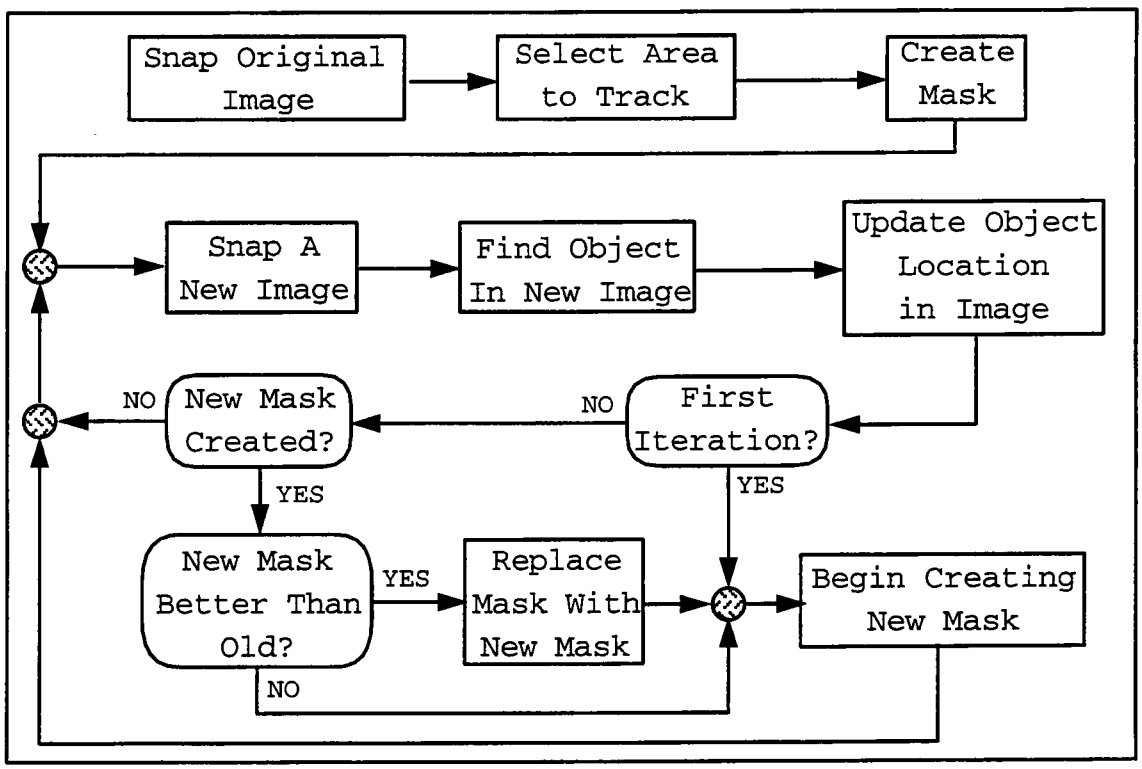


Figure 5.1: *Flow chart of tracking algorithm.*

algorithm. Some criteria are the energy of the region, the presence of both features in the horizontal and vertical directions in the image, the presence or absence of energy at a certain complex frequency, or the presence of a texture in the image. Any of these criterion may be used or a combination of them.

Once the region is selected, the mask must be prepared for the correspondence algorithm. For some algorithms like the SSD and the NCC, this is simply copying the region from the image. For the MACE and MINACE, this is a more complex process where complex manipulations of the region must be performed in the Fourier domain. With this done, the initialization is complete. This process is only executed once to start the algorithm and now the tracking and mask updating may begin and operate concurrently.

The second step is to snap a new image and locate the object in this image. This is accomplished using some type of correspondence algorithm. The algorithm will give a result stating where the object is moved in the image. This new location is remembered and used in the next iteration of this step.

The location of the search area will be updated using the previous information about the location of the object. The simplest way to use this information is to search in an area that is centered at the location of the object in the previous image. Another way would be to use the previous locations of the object in the previous series of images, find a trajectory for the object, and predict where the object will be. The algorithm can then search in an area centered around where the object is

predicted to be located. A final, and very time consuming way of tracking the object is to search the entire image for the object. This increases the chances of finding false matches because there is more area to give false matches.

Operating simultaneously with the tracking of the object is updating of the mask. The reason the mask is updated is because the object in the environment is changing in the scene. The camera creates a two dimensional representation of a three dimensional world. All of the problems in tracking discussed in Chapter 2 come into effect. The object, camera, and the environment are all changing and this means that the object will be changing in the camera view. With a mask updating, the object being tracked can be tracked with a correspondence algorithm.

Even though a new mask may be created, it does not necessarily replace the previous one. It may be a worse one than the previous one. This could be because the new mask was taken from a region that was partially occluded. The object could also have rotated in one direction, but now the object was moving back. Some criteria in deciding if this new mask is better than the previous must be used. One of these would be to test both masks in the next image. If they both selected the same region, then the mask with a larger peak might be selected. Instead of choosing between the two masks, they might be combined in some way that some of the old information and some of the new information exists.

This creation of the new mask will operate simultaneously with the finding of the object. It is possible that the mask creation will take longer than the calculations of

finding the object. This just means that the mask update rate would be lower than the object localization update rate.

5.2 Parameters Used in Implementation of Tracking

With the generalized tracking algorithm used in this thesis, the parameters used in the algorithm will now be described. These logic for the choice of some of these parameters will be given. Some different parameters will also be described. These reason for the choice of each parameter will also be given.

5.2.1 Correspondence Algorithm and Mask Size

The correspondence algorithm used was correlation using the MINACE filter. The reasons for the choice of this filter are given in Chapter 4. The accuracy and speed of the various algorithms considered were taken into account. The MINACE with a 32×32 mask size was chosen because it had a good combination of speed and accuracy. The disadvantage of the MINACE algorithm is that it required a threshold. The threshold was chosen experimentally and set to a constant of 5.0×10^{-6}. This gave satisfactory results.

5.2.2 Search Area Size

The tracking algorithm in this thesis searches within an area centered on the last location the object was found. This means that in order to find the object, the search area must be large enough that the object will not have moved outside of it. A 51×51 size of the search area was chosen to be the mask sized used in this thesis. This size was designed to track an object moving two meters away from the camera at five kilometers per hour. This is approximately walking speed. The camera has a focal length of 12.5 millimeters which gives a field of view of approximately 27^o.

The pixel density (ρ) at a distance (d) from the camera with resolution of (α) can be calculated with the equation

$$\rho = \frac{2 \times d \times tan(\frac{\theta}{2})}{\alpha}. \tag{5.1}$$

At a distance of five meters, a field of view of 27^o, and a pixel resolution of 512 pixels, the pixel density is 1.88 millimeters per pixel. The number of pixels traveled each frame (s) can be calculated with the object's velocity (v), a pixel density of ρ, and a camera frame update frequency (f) by using the equation

$$s = \frac{v}{\rho f}. \tag{5.2}$$

Using this equation, a ρ of 1.88 millimeters per pixel, v of five meters per second, and a f of 30 Hertz, s becomes 24.6 pixels traveled each frame. This means with no

prediction, the object can move 25 pixels up, down, left, or right. This means that the search area must be as large as 25 pixels (up or left) plus 25 pixels (right or down) plus 1 additional pixel for no motion at all. This gives a search area that must be a size of 51×51.

5.2.3 Object Location Update Rate

The speed at which the location of the object can be updated is critical in the tracking of an object. With a faster update rate, faster moving objects can be tracked. There are two factors affecting the update rate of the tracking algorithm. These are the rate at which the correlation can be calculated and the rate which the images become available.

With this version of the tracking algorithm, the object can be found in 43ms. This was listed in Table 4.1. This is the time it takes to correlate a 32×32 mask with an 82×82 image giving a 51×51 result area. A large part of this time is searching for the maximum value. This is presently computed on the Motorola 68030 processor. It may be possible to place this part on the Datacube MV20 processor and speed the update rate even farther.

The camera is a ccd camera that broadcasts composite video at 30 hz. This is a continuous broadcast of information. The camera does not sit idle and wait for the frame grabbing equipment to request a frame to be sent. Instead, the camera broadcasts the images at 30 frames per second. If the image processing hardware can

process 60 frames per second, it is no use because the camera does not broadcast that quickly. Also, if the hardware takes slightly longer than one thirtyth of a second, it will miss a frame. This means that instead of going at 29 frames per second, it will only go at 15 frames per second. This is because it can only process every other frame available.

For this tracking algorithm, the update rate is 15 times per second. This is because the correlation algorithm takes 43ms and the camera will not slow down for this rate. Since the correlation algorithm takes longer than 33ms, the algorithm can only use every other frame. Every other frame from the camera gives a rate of 15hz.

5.2.4 Mask Replacement Criteria and Rate

While the object was being tracked, a new mask was being created. This mask was created from an image while the object was being tracked. This means that periodically, after the object was located, this area would be saved and a new mask would begin to be created. This would be done at a lower priority that the actual tracking of the object. A new mask would be created approximately every second.

The new mask would not necessarily replace the previous mask. This new mask could be worse than the previous one. A couple criteria for the replacement of the mask were considered. One was if the scaling factor that made the mask fit inside eight bits was similar to the previous one. This scaling factor will be explained in the hardware section to follow. Another criteria was that the expected value was similar

in both the previous mask and the present one. When either of these values were compared to with masks created from a similar region, and a region that was not related to the region, there was no clear threshold that could be set. The expected values and the scales were similar for masks created from several regions. Because of this, neither of these were used as a mask replacement criteria.

Lacking a better criteria, this algorithm simply checks if the previous mask and the new mask give a peak value at the same location. If they both do, then the new mask replaces the old mask. This is not the best criteria, but it works most of the time. A better criteria needs to be found. It has been observed that sometimes the mask will be replaced and will have many more false matches than the previous mask.

5.3 Hardware Specific Details of The Tracking Algorithm

This third section describes how the hardware was used in implementing the tracking algorithm. First the equipment will be described. Then individual aspects of implementing the algorithm on this hardware will be described. The main use of the pipeline processor is correlation. The pipeline processor is able to to 8×8 correlation as fast as it could do normal multiplication. The way a 32×32 correlation was done will be described in a subsection.

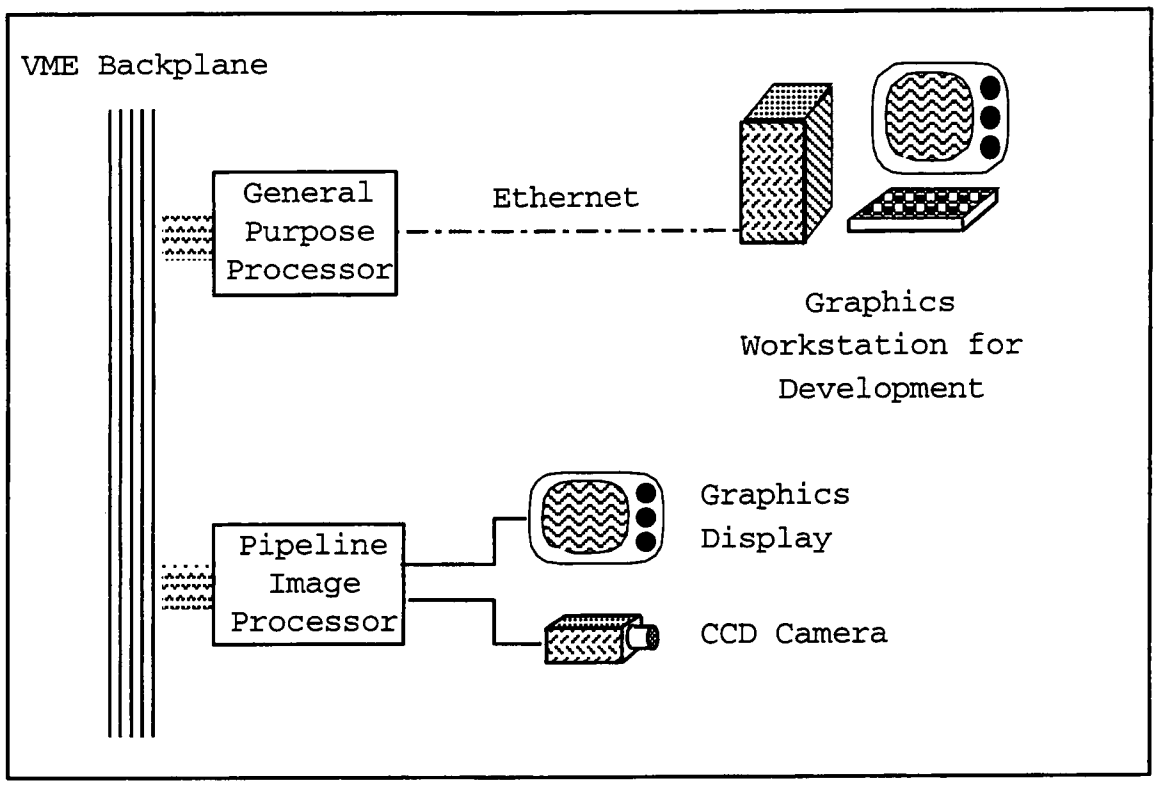


Figure 5.2: *Hardware platform on which the experiments were conducted. The general purpose processor communicated with the pipeline image processor through the VME backplane. The pipeline processor was connected to a camera and a display. The general purpose processor was connected to ethernet and the programs could be loaded from a graphics computer.*

5.3.1 Experimental Testbed

The hardware platform consists of a Datacube MV20 processor, a Motorola 68030 processor and a CCD camera. All of this equipment is located on a VME rack. A Silicon Graphics workstation was used to develop the code and load it to the Motorola 68030. This combination of equipment are a powerful image processing combination. All of this was implemented using VxWorks [49, 50], the real time operating system. Shown in Figure 5.2 is a diagram of how the system components are connected.

The Datacube is a pipeline processor [13, 12, 14] can do simple image manipulations at frame rate. In a pipeline processor, the image can be thought of a linear stream of data bits. A few examples of ways these bit streams can be manipulated are addition of two image streams, subtraction of images, multiplication of images, averaging an image, thresholding the image, or changing one image according to a different image. The timing of these actions can be very quick. As an image stream flows into and out of an arithmetic element at the same rate. There is a period of time between when an individual pixel enters and leaves the element. The pipeline processor also can be have multiple elements run in series or parallel. This means that one stream can be multiplied, thresholded and averaged in one pipe firing. All of this makes the pipeline architecture very powerful.

The pipeline processor can operate on whole images quickly. It does not work well when it must do calculations depending on single pixels. An example of this is a region growth program that searches for similar neighbor pixels and clusters them together. Since the course of the algorithm depends on previous and past pixels, the pipeline processor can not implement this region growth algorithm.

Another limitation of the Datacube MV20 is that it does not do floating point operations. The results of the operations are limited to either eight or sixteen bits. Internally, the processor can have up to 20 bits, but this data must be truncated to eight or sixteen bits before being saved to a memory surface. Algorithms like the NCC which have floating point division and multiplication are difficult to implement

on the MV20.

For operations that the MV20 is not capable of doing, the Motorola 68030 microprocessor is useful. This processor can be used in floating point operations like the FFT. It can act as the brains of the tracking routine. While the MV20 is the workhorse of the image processing routines, the 68030 will be controlling the operations and commanding operations depending on the results of the image processing elements.

5.3.2 Pipeline Data Pathways

The Datacube MV20 has can have operations happening on it simultaneously. These data used in these operations will flow through pipes. These pipes can operate simultaneously as long as they do not share any elements. Figure 5.3 shows how the data pathways move on the Datacube MV20 processor.

The MV20 has six-eight bit image memories. These memories can be directly accessed by the Motorola 68030 processor. They are triple ported which means they can input data, output data and be accessed by the 68030 processor simultaneously. This means that the processor has access to any of the memories at any time. Memories zero through three can have their output connected to the display. Memory zero through two can be set up to be the red, green, and blue planes or each one can be set up as a gray scale memory. Memory three can be used as an overlay connected to the display.

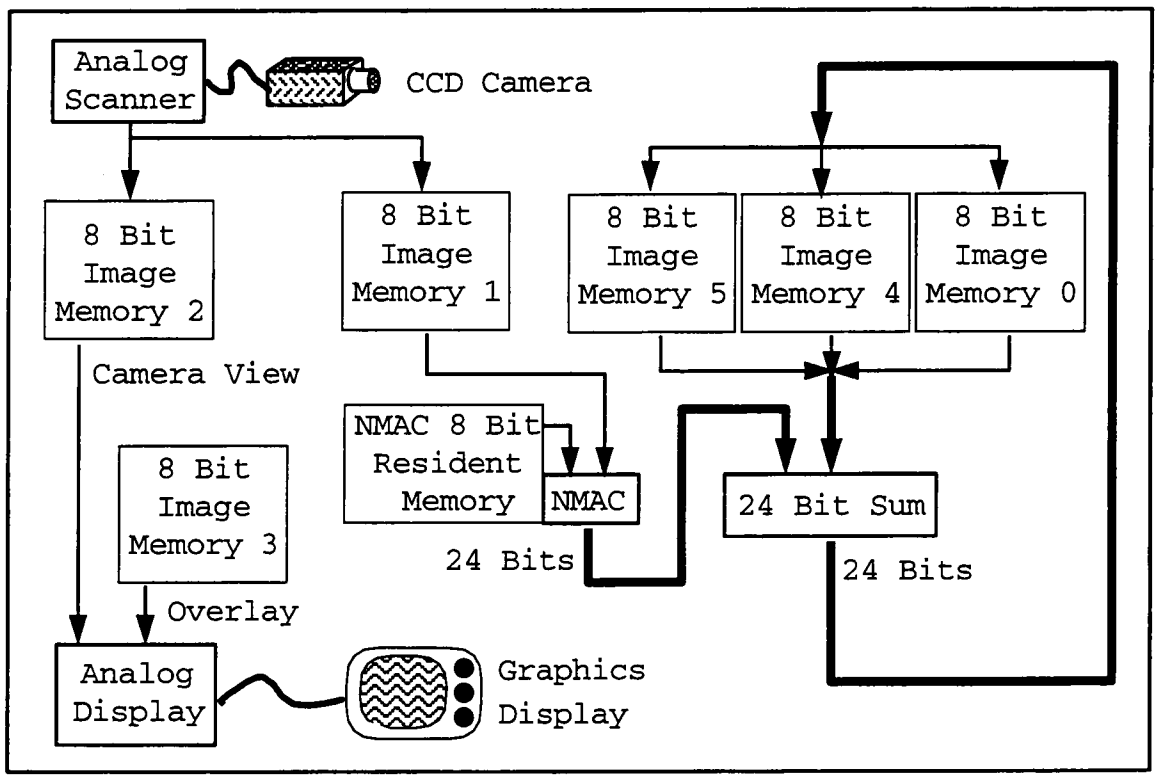


Figure 5.3: *Various data pathways on the pipeline image processor. Note three specific operations in this figure. These are the image acquisition, the image display and correlation.*

The display is set up so that memory two displays the information from the camera. Memory three is used as an overlay. For the tracking algorithm, memory three is used to display the crosshairs showing the location of the object. The display runs at 40 mhz.

The analog scanner reads the information taken from the camera and converts it to a digital image. This image is then broadcast into memory one and two. Memory two then sends it to the graphics display. Memory one is the source for a correlation with the neighborhood multiplier and accumulator (NMAC). The information coming from the analog scanner is double buffered. This is to prevent the data from being changed by the analog scanner while it is being output to the correlator. The data from the scanner is either broadcast to the left or the right half of the image memory. The memory area used by the NMAC comes from the other half. Whenever a new image is needed, the memory location that the scanner sends to is switched. This way, when new data is needed, the most recent data will be saved and the scanner will begin writing to the other memory.

The third operation is the correlation. In this operation, an image from memory one is correlated with the resident memory of the NMAC. This information is then sent out with twenty four bits of data. These twenty four bits are then summed with the information in memories five, four and zero. Memory five contains the most significant bits and memory zero contains the least. This process is repeated several times giving a running sum. This is to allow correlation of greater than 8×8 size.

This will be explained in more detail in the next subsection.

5.3.3 Correlation on the MV20

The Datacube MV20 is equipped with an element called the "neighborhood accumulator and multiplier' (NMAC). This element is a correlation element that will correlate a mask of up to an 8×8 size against the image or a region of the image. The NMAC can do these operations on a 512×512 image at 30 Hertz. It can operate much faster when the search area is just 51×51. This NMAC which can only operate on an 8×8 mask size can be used to compute a 32×32 mask size correlation. This section will explain how the NMAC is used to do 32×32 correlations.

In order to understand the concept of using a smaller correlation mask size to compute a resulting correlation with larger mask size, an example of how a 4×4 correlation with an image can be computed using four 2×2 masks will be given. Once the concept of using four 2×2 masks to do a 4×4 correlation is understood, the concept can be be expanded to larger mask sizes. This is what is done in the Datacube MV20 to do a 32×32 correlation using sixteen 8×8 correlation masks.

The image in this example has a 8×8 size. The mask is of size 4×4. If the 8×8 image were correlated with the 4×4 mask in one step, the resulting valid data would be of size 5×5. This is demonstrated in figure 5.4. The reason that the valid data is not 8×8 is because the 4×4 correlator must be completely inside the image to be able to correlate. When the correlator has slid to the first row and the fifth column,

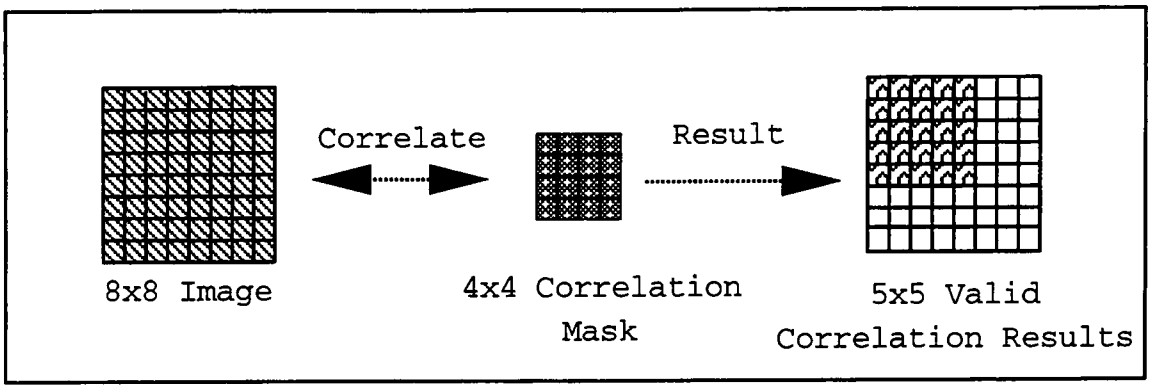


Figure 5.4: *Correlation of an* 8×8 *image with a* 4×4 *mask. The result region will have* 5×5 *valid data points. This is shown in the above figure where the* 8×8 *image is correlated with a* 4×4 *mask size and has* 5×5 *valid data points.*

the mask's final row will be outside of the image. It is unclear what value should be given to the area outside the image. It could be a repeat of the nearest neighbor, it could be set to zero, it could be set so that the image wraps around and the value given is the value in the first column or the result of this correlation can be called invalid and removed. At first glance, one might think that the number of valid results is the image size minus the mask size. This is not correct. Imagine a $n \times n$ image correlating with a $n \times n$ mask. The result will have one valid correlation where the mask fits exactly over the image. The valid correlation size is the size of the image minus the size of the mask plus one. For the case of an 8×8 image and a 4×4 mask this would give 5×5 valid data points in the image.

The reason for this example is to demonstrate using a 4×4 correlation using a 2×2 mask size. For this reason, imagine that Datacube has a smaller version of the MV20 called the MV5 that is only capable of doing a correlation using a 2×2 mask

size. The 4×4 mask must be broken into four 2×2 masks. Figure 5.5 shows how the 4×4 mask is broken into four 2×2 masks labeled a-d. When the MV5 correlation correlates a 2×2 mask with an 8×8 image the results would occupy a 7×7 area. This is demonstrated in Figure 5.6.

The next step with the 4×4 mask broken into four 2×2 masks, is to correlate the four masks with the image. The result of this is shown in Figure 5.7. Each result has an 5×5 region (labeled A-D) that is extracted from these correlation results and saved. These regions will be used to give the results of the 4×4 correlation.

Correlation is simply the sum of a pixel by pixel multiplication between the mask and the image. The results of the origin come when the mask is situated beginning at the origin. The results the next pixel to the right come from when the mask is shifted a pixel to the right. When the 4×4 correlation mask is broken into four 2×2 correlation masks, the results of these four 2×2 must be added to give the results of the 4×4 correlation. Mask a in Figure 5.5 is the top left part of the 4×4 correlation. The results of a correlation of mask a at the origin overlap the results of the 4×4 correlation at the origin. Mask b on the other hand is two pixels to the right of the origin. The results of the 4×4 correlation at the origin require that the mask b be located two pixels to the right of the origin. This is the reason that the 5×5 region cut in Figure 5.7 for region A has its corner located in the top right and region B is two pixels to the right. Similarly, region C is two pixels down and region D is two pixels down and two pixels to the right.

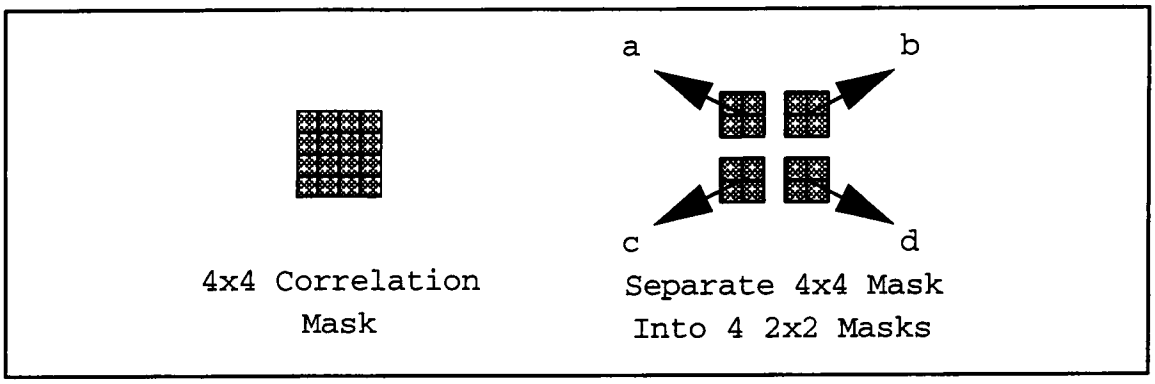


Figure 5.5: *A* 4×4 *mask can be broken into four* 2×2 *masks. This is done above and the masks are labeled a-d.*

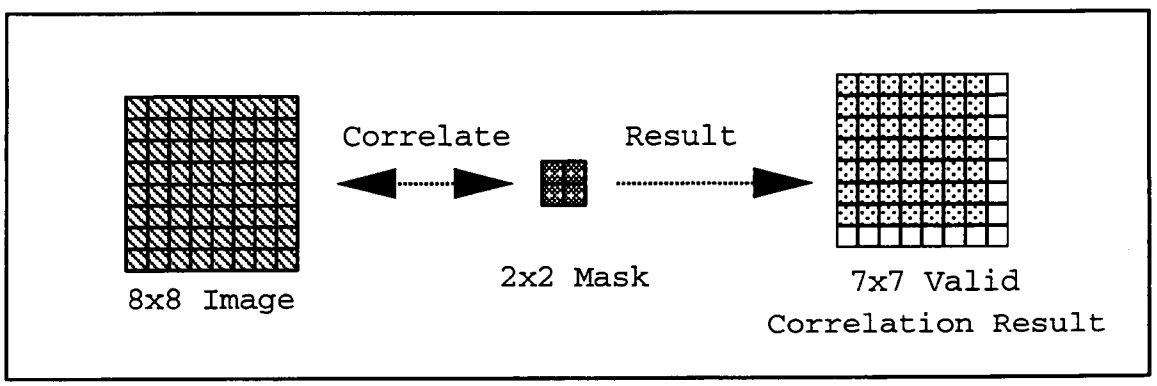


Figure 5.6: *Correlation of an* 8×8 *image with a* 2×2 *mask. The resulting region has* 7×7 *valid data points. This is shown in the above figure where the* 8×8 *image is correlated with a* 2×2 *mask size and has* 7×7 *valid data points.*

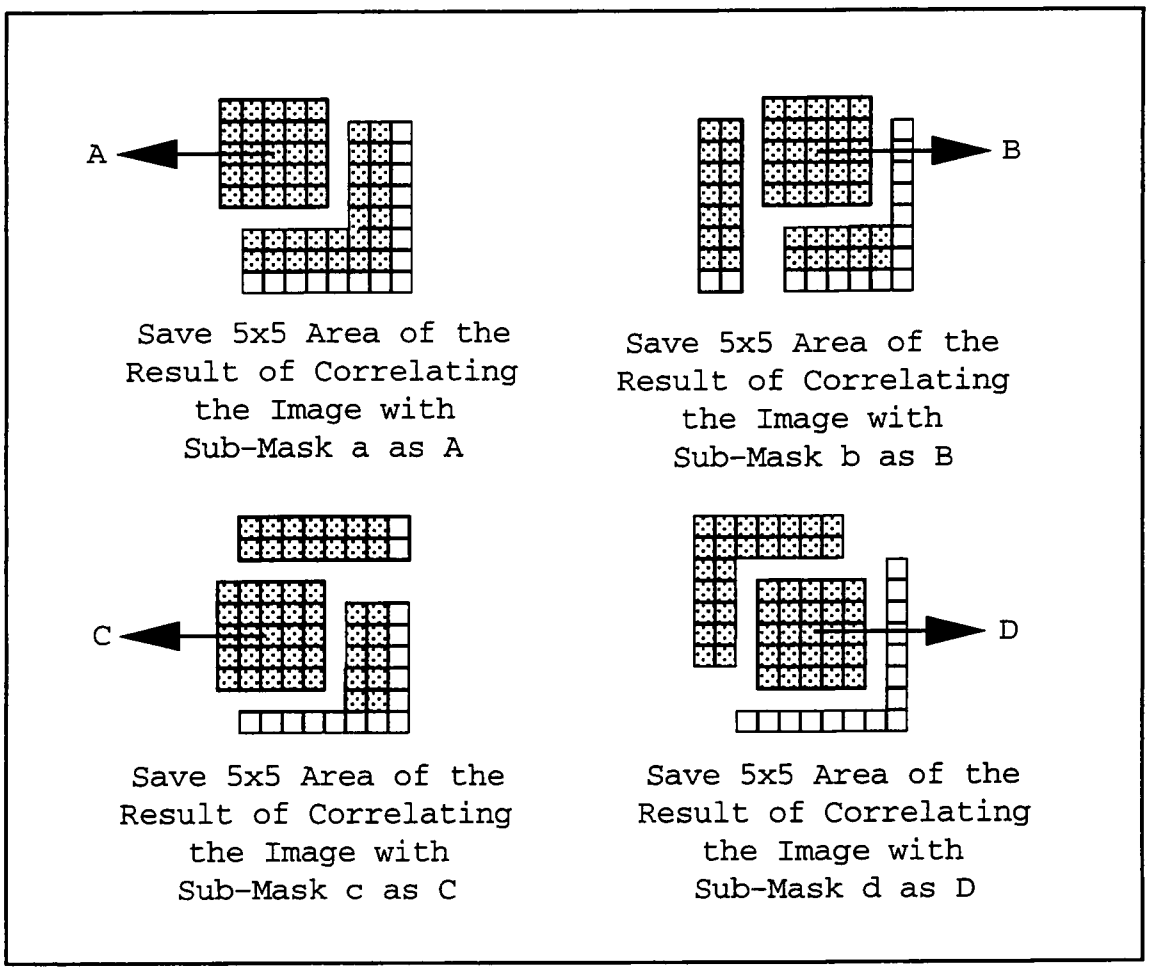


Figure 5.7: *Correlation of four* 2×2*. This gives* 7×7 *valid result points. Notice that a region of* 5×5 *has been cut out of each result and labeled A-D. These regions will be used in using four* 2×2 *masks to do a* 4×4 *correlation with an* 8×8 *image.*

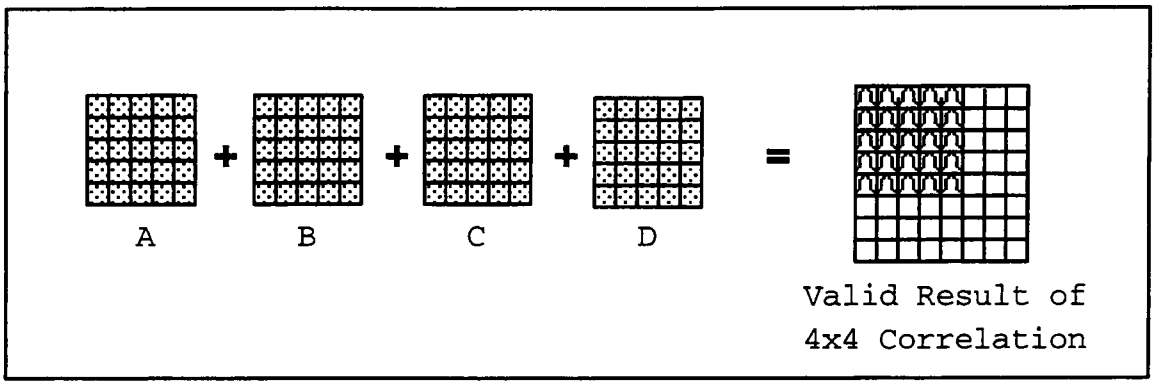


Figure 5.8: *Sum of the results of four* 2 × 2 *correlations. This will give a* 5 × 5 *region. This region contains the same values as the results when the* 4 × 4 *mask which is composed of the four* 2 × 2 *masks is correlated in one step.*

With the correct regions of the results of the correlations saved, the final step is to sum the regions. This sum will give results that are the same as a correlation of an 4 × 4 mask. This final step is shown in Figure 5.8. In this way, a 4 × 4 mask can be broken into four 2 × 2 masks, correlated, summed and have the same result as the 4 × 4 mask would give if it were correlated in one step.

This idea of breaking a correlation of a larger mask with the sum of correlations of smaller masks was extended for a 32 × 32 mask being correlated using an 8 × 8 mask. Shown in Figure 5.9 is a representation of how the Datacube MV20 pipeline architecture would implement this correlation. When an 82 × 82 image is correlated with an 32 × 32 mask, the resulting valid data is 51 × 51 in area. The Datacube processor would correlate an 8 × 8 piece of the 32 × 32 mask with the search area and sum this to the previous results. The initial 51 × 51 result area is initialized to zeros before any correlation happens. The sixteen 8 × 8 masks are then correlated

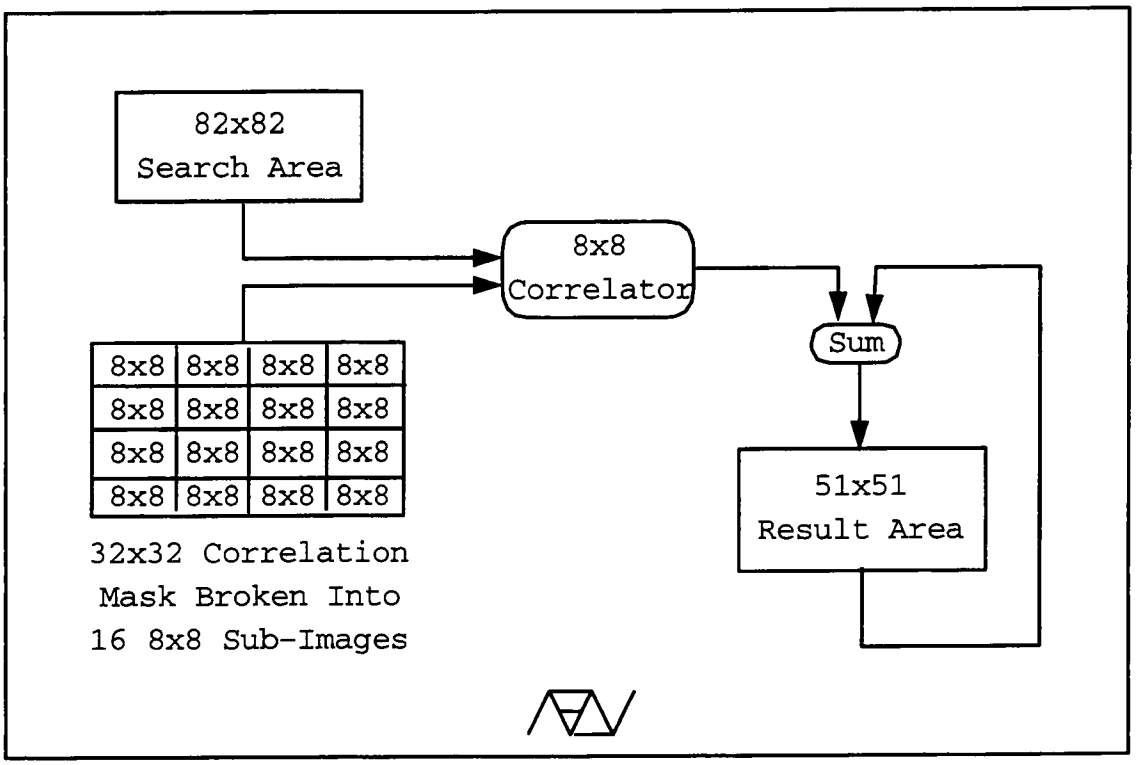


Figure 5.9: *Correlation on a MV20 pipeline processor. This uses the built in* 8×8 *correlator (NMAC) and a running sum. The result area is initialized to zeros before the first correlation happens.*

one at a time and summed in the running total in the result area. This means that the 8×8 masks are fired through the 8×8 correlator and into the sum element sixteen times before the results are valid.

5.3.4 Truncation of the Mask to Eight Bits

When the MINACE mask is created, the values of it are floating point. Usually, the values are much less than one. in order to be used in the correlation algorithm, these values are converted to have eight bits of resolution. This means that they are

scaled to fit from -128 to 127. The scale factor is the absolute value of the inverse of the maximum value. This scaling causes a reduction in precision. This precision reduction has not adversely affected the performance of the mask. The result of the 32×32 correlation could have a maximum value of

$$32 \times 32 \times -128 \times 255 = -37748736 > -67108864 = -(2^{26}). \tag{5.3}$$

This corresponds to a 32×32 mask with a value of all -128 correlating with an image that contains all 255. This could be fit into a number containing 27 bits which would correspond to 26 bits to contain the value and one sign bit. Both a mask and a resulting value this high are very unlikely.

Instead of worrying about this worse case, the expected value was taken into account. The expected value was only allowed to occupy 23 bits. This gave 22 bits for magnitude and a sign bit. If the expected value was greater that twenty three bits, the result of the summation would be shifted until the expected value could occupy 23 bits. This allowed the true value to be twice that of the expected value. When the algorithm ran, the expected value always would fit into 23 bits.

5.4 Performance Evaluation

The performance of the tracking algorithm was evaluated for consistancy in two ways. In the first, an object that was not moving was tracked and the variation in the

resulting locations was compared. In the second, the algorithm tracked an object that was moved away from an initial position and then returns to to that position. The location of the object after the return is then recorded. These give results for the stability of the tracking algorithm with and without object motion.

The first test evaluated the stability of the tracking algorithm when the object had no motion. The experimental setup of this test was simple. An object was placed in front of the camera. The algorithm was told to track the object for one thousand counts. The results of these tracking values were recorded. It should be noted that the mask was allowed to update during this tracking. Since the mask could change, this was then a test of the tracking algorithm and not a single MINACE mask.

A graph showing these resulting values is given in Figure 5.10 for the x motion and Figure 5.11 for the y values. The values along the abscissa correspond to the value of a counter with 60 hz ticks. It would be assumed that with a fifteen hz update rate, 1000 iterations would end after 4000 60Hz ticks. The results are slightly longer than 4000 clicks. This is because every time the mask is created, an extra run of this mask must be done si its results can be compared to the current mask. At this time, the update rate is 7.5Hz. This causes the total number of clicks to be slightly more than 4000 clicks.

The standard deviation in the x direction was 0.0266 pixels. The standard deviation in the y direction was 0.0111 pixels. This small deviation of the tracking algorithm demonstrates the consistancy of the tracking algorithm.

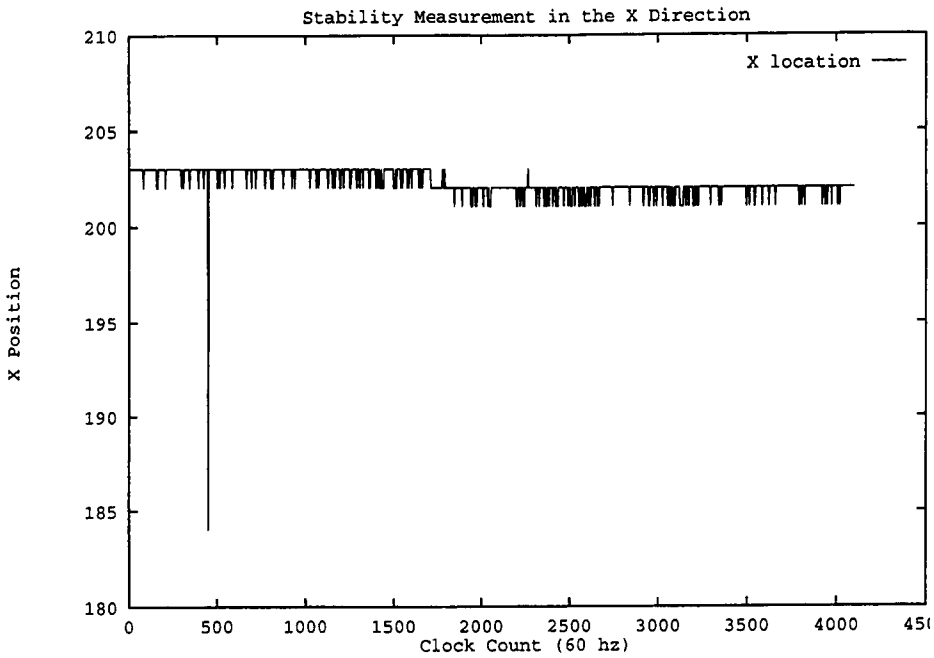


Figure 5.10: *Stability measurement of no motion in the X direction.*

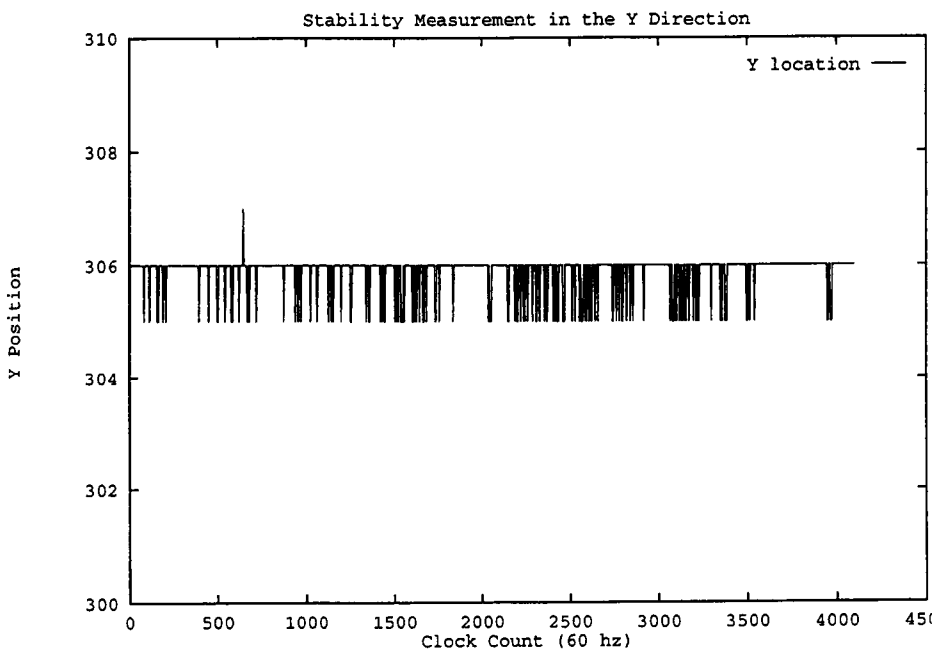


Figure 5.11: *Stability measurement of no motion in the Y direction.*

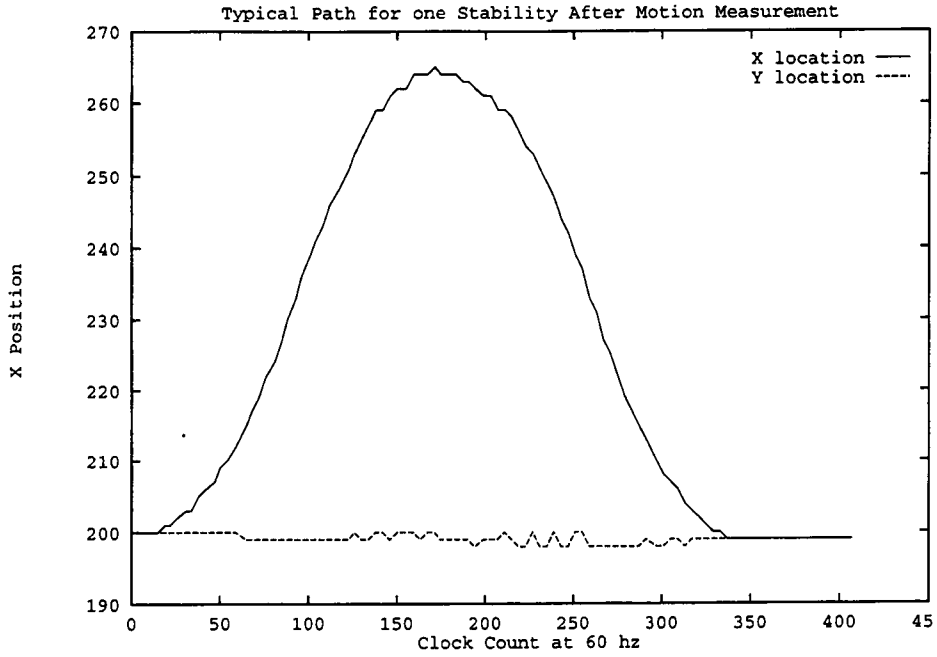


Figure 5.12: *Typical tracking result during the second stability test.*

The second test evaluated the stability of tracking an object in which a motion was imposed upon the object. This motion demonstrated in Figure 5.12. The object is located in an initial position, it is then moved away from that position along the x axis, and finally the object is moved back to the initial position. This motion takes approximately 5 seconds and moves 65 pixels from the initial position. The tracking algorithm tracks from the initial position, during the motion, and back to the initial position. The initial position is always the same. Before each motion, the tracking algorithm is trained on the current object at the original initial position. This means that if the tracking algorithm gave incorrect results in one run of the experiment, the next run of the experiment would not rely on those results.

The results of these experiments along the x axis are given in Figure 5.13. The

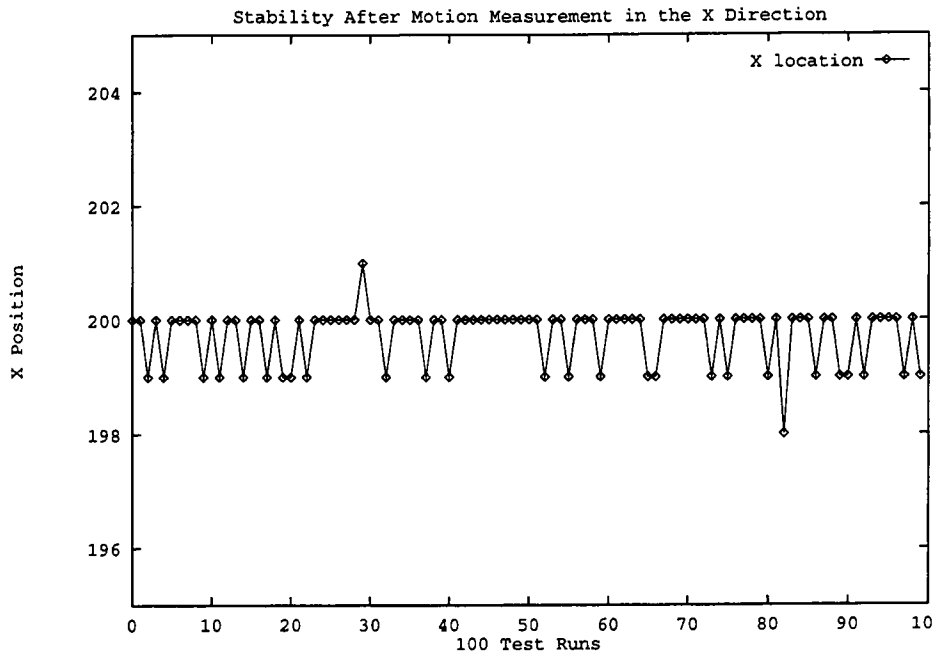


Figure 5.13: *Final x location of object with an initial x location of 200.*

results of these experiments along the y axis are given in Figure 5.14. Each data point corresponds to the final location after a full motion of the object. The initial position was at $(200, 200)$. The standard deviation from this initial position was 0.0487 for the x direction and 0.0577 for the y direction. These low deviations demonstrate how robust the algorithm is when tracking the object.

5.5 Summary

This chapter described the tracking algorithm implemented on the pipeline architecture. It began by describing the general tracking algorithm. It next gave the parameters used in the implemented tracking algorithm. Third, it detailed some in-

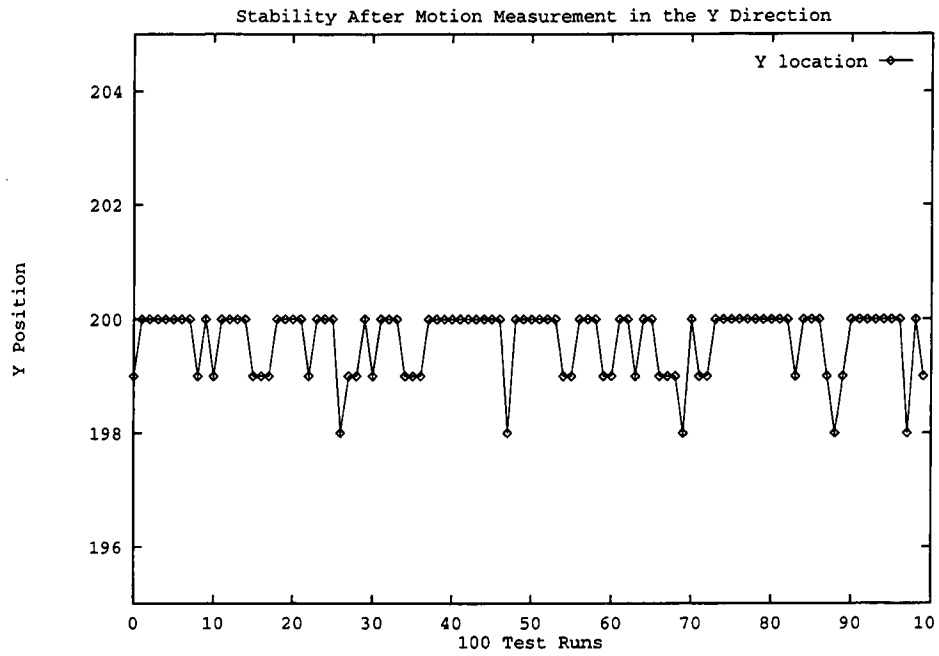


Figure 5.14: *Final y location of object with an initial y location of 200.*

dividual aspects of implementing this algorithm on a VME rack with a Datacube MV20 processor and a Motorola 68030 CPU. The platform was able to track the object at 15hz. Finally, the performance of the tracking algorithm was evaluated for consistancy. These tests showed how little the results deviated from the true values.

Chapter 6

Summary and Concluding Remarks

6.1 Summary

In this thesis, a correspondence based tracking algorithm was detailed. In order to create the tracking algorithm, several concepts of tracking had to be examined, different tracking algorithms had to be explored, and the available hardware had to be considered. With all this information, the algorithm then had to be designed and built on the available hardware platform. The six steps in creating the tracking algorithm are as follows:

1. Defined the issues associated with tracking. These issues were related to the object, the environment, and the camera. It was fundamental that these concepts were acknowledged before a tracking algorithm could be described.

2. Examined seven different tracking algorithms. After looking at the capabilities of these algorithms, it was decided that an algorithm based on correspondence

would be best for an object which little prior knowledge.

3. Introduced four correspondence algorithms. Some of these have been commonly used in tracking for years. Others came from frequency based approaches. These were the MACE, the MINACE, the SSD, and the NCC algorithms.

4. Designed tests to compare four different correspondence algorithms. These algorithms were compared for accuracy and speed. The target hardware was included in the tests for speed. This way, the correspondence that had the best combination of accuracy and speed on the target hardware system could be used.

5. Chose the MINACE algorithm to be the best combination of speed and accuracy for the targeted hardware system.

6. Built and evaluated a system that used this tracking algorithm. It used a pipeline processor, a general purpose processor, a camera and a display. It was able to track at 15hz.

The system designed is fast. At a rate of 15hz, many tracking applications can be done. The system is adaptive. Since the algorithm is trained by the object to be tracked, it will use the information that is there. It does not rely on a previous model for the object. The system works. The system will track an object consistantly as demonstrated by the low deviation in the evaluation of the system. There are

instances where it will lose track of the object. Many of these instances of losing track of the object can be eliminated by the suggestions given in the final section.

6.2 Concluding Remarks

There are several areas which could be explored using this platform for testing. This tracking algorithm works well, but the researcher is confident that the capabilities can be improved. There are also some ways that could improve the testing of the algorithm. Several of the directions that could be explored will be listed in this section.

At the moment, the algorithm has no way of knowing if the data values are valid. There needs to be some criteria to tell if the results of correlation are good. One way of doing this is predicting where the location of the object should be. By estimating the velocity and the acceleration of the object, the expected location of the object can be predicted. In this way, results of correlation that are not near this predicted location can be ignored or their effect on the location of the object can be reduced.

Another way to improve the algorithm is to find way to examine the results of the correlation. This might take into account the resulting values in the immediate vicinity of the result. It might consider the peak to side-lobe ratio. It might also find the top candidates instead of the one peak and then use prediction to choose the better one. It is quite possible that the result values other than the peak value could be useful in finding the true location of the object.

In this thesis, the threshold for the MINACE mask was set and used. It may be possible to improve the tracking capabilities of the mask by changing this threshold. This might be done by considering the training image for the correlation or possibly the surrounding values of the mask. A change in threshold might cause something that could not be tracked with a fixed threshold be easily tracked with the new threshold.

Four correspondence algorithms were examined in this thesis. It is very possible that another correspondence algorithm would work better. It is possible that a rotation invariant correlation filter will work better. It is also possible that some algorithms will work better for some regions and other algorithms will work better for others. The tracking algorithm could even switch correspondence algorithms depending on the region being searched for.

Another direction to go would be to try to make the correlation masks examined in this thesis operate quicker. The sum of squared differences is a very good candidate. If someone used very large scale integration (VLSI) to make a chip that could to SSD in much the same way as the NMAC worked on the MV20, a powerful tracking system could be created. This would have the speed of the MINACE filter and the accuracy of the SSD.

A final way of improving the tracking algorithm is to place more of the tracking algorithm on the Datacube processor. The Motorola still has to copy the result mask from the MV20 and search for the maximum. It may be possible to place this

operation on the MV20. If the speed could be taken from 43ms to 29ms, the algorithm could be run on every image from the camera. This would mean that the tracking would be at 30hz instead of 15hz.

BIBLIOGRAPHY

Bibliography

[1] A. L. Abbott and N. Ahuja. The university of illinois active vision system. In *Intelligent Robots and Computer Vision XI*, volume 1825, pages 757–768. SPIE, 1992.

[2] P. K. Allen, A. T., B. Yoshimi, and P. Michelman. Trajectory filtering and prediction for automated tracking and grasping of a moving object. In *International Conference on Robotics and Automation*, pages 1850–1856. IEEE Computer Society Press, May 1992.

[3] P. K. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. In *International Conference on Robotics and Automation*, pages 851–856. IEEE Computer Society Press, April 1991.

[4] P. E. Anuta. Digital registration of multispectral video imagery. *Selected Papers on Digital Image Processing*, MS 17:227–234, 1990.

[5] M. Barth and S. Tsuji. Egomotion determination through and intelligent gaze control strategy. Technical report, Osaka University, January 1992.

[6] C. Brown, D. Coombs, and J. Soong. Real-time smooth pursuit tracking. In *Active Vision*, pages 123–136. MIT Press, 1992.

[7] H. I. Christensen. The auc robot camera head. In *Intelligent Robots and Computer Vision X*, volume 1708, pages 26–32. SPIE, 1992.

[8] D. Coombs and C. Brown. Real-time smooth pursuit tracking for a moving binocular robot. In *Computer Vision and Pattern Recognition*, pages 23–28. IEEE Computer Society Press, June 1992.

[9] J. D. Crisman. Color region tracking for vehicle guidance. In *Active Vision*, pages 107–120. MIT Press, 1992.

[10] R. Curwen and A. Blake. Dynamic contours: real-time active splines. In *Active Vision*, pages 39–57. MIT Press, 1992.

[11] A. K. Dalmia and M. M. Trivedi. Structure and motion analysis: a critical review. Technical Report TR-CVRR93-1, CVRR Lab, University of Tennessee, August 1993.

[12] Datacube Inc. *Datacube Image Processing Manual*, sm101-32 edition, March 1993.

[13] Datacube Inc. *ImageFlow Reference Manual*, if103-6.0 edition, October 1993.

[14] Datacube Inc. *MaxVideo 200 Hardware Reference Manual*, hw599-2.1 edition, March 1993.

[15] U. R. Dhond and J. K. Aggarwal. Structure from stereo: a review. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 19(6), December 1989.

[16] N. J. Ferrier. The harvard binocular head. In *Applications of Artificial Intelligence X: Machine Vision and Robotics*, volume 1708, pages 2–13. SPIE, 1992.

[17] R. C. Gonzalez and R. C. Woods. *Digital Image Processing.* Addison-Wesley, United States of America, 1992.

[18] C. Harris. Tracking with rigid models. In *Active Vision*, pages 59–73. MIT Press, 1992.

[19] B. K. P. Horn and B. G. Schunk. Determining optical flow . *Artificial Intelligence*, 17:185–203, 1981.

[20] M. Jenkin, E. Milios, and J. Tsotsos. Trish: the toronto-iris stereo head. In *Applications of Artificial Intelligence X: Machine Vision and Robotics*, volume 1708, pages 36–46. SPIE, 1992.

[21] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *Int. Journal of Computer Vision*, 1(4):321–331, 1987.

[22] S. Kenue. Lanelok: detection of lane boundaries and vehicle tracking using image processing techniques: parts i and ii. In *Mobile Robots IV*. SPIE, 1989.

[23] K. Kuan, G. Phipps, and A. C. Hsueh. Autonomous land vehicle road following. In *First International Conference on Computer Vision*, pages 557–566, London England, 1989. SPIE.

[24] F. Leymaire. Tracking and describing deformable objects using active contour models. Technical Report TR-CIM-90-9, McGill Research Center for Intelligent Machines, McGill University, Montreal, 1990.

[25] A. Mahalanobis, B. Kumar, and D. Casasent. Minimum average correlation energy filters. *Applied Optics*, 26:3633–3640, September 1987.

[26] S. B. Marapane and M. M Trivedi. Active vision system for depth extraction using multi-primitive hierarchial stereo analysis and multiple depth cues. In *Proc of SPIE Sensor Fusion and Aerospace Applications*, Orlando, FL, April 1993.

[27] S. B. Marapane and M. M Trivedi. Multi-primative hierarchical stereo analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 16, 1994.

[28] S. B. Marapane and M. M Trivedi. Simulation and visualization of integrated sensory motor systmes. In *IEEE Workshop on Visualization and Machine Vision*, Seattle, WA, June 1994.

[29] B. N., N.P. Papanikolopoulos, and P. K. Khosla. Dynamic sensor placement using controlled active vision. In *1993 World Congress*, Sydney, Australia, July 1993. International Federation of Automatic Control.

[30] K. Pahlavan, T. Uhlin, and J. Eklundh. Heads, eyes and head-eye systems. In *SPIE Conference on Applications of Artificial Intelligence X: Machine Vision and Robotics*, volume 1708, 1992.

[31] N.P. Papanikolopoulos and P. K. Khosla. Adaptive robotic visual tracking: theory and experiments. *IEEE Transactions on Automatic Control*, 38:429–445, 1993.

[32] N.P. Papanikolopoulos, P. K. Khosla, and T. Kanade. Applying the controlled active vision framework to the visual servoing problem. In *Americal Control Convferance (ACC '93)*, June 1993.

[33] N.P. Papanikolopoulos, P. K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9:14–32, 1993.

[34] W. K. Pratt. Correlation techniques of image registration. *Selected Papers on Digital Image Processing*, MS 17:243–248, 1990.

[35] W. H. Press, B. P. Flannery, S. A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, New York, 1988.

[36] G. Ravichandran and D. Casasent. Noise and discrimination performance of the minace correlation filter. In *Proceedings of SPIE*, pages 233–248, April 1991.

[37] G. Ravichandran and D. Casasent. Minimum noise and correlation energy (minace) optical correlation filter. *Applied Optics*, 31(11):1823–1833, April 1992.

[38] H. Schneiderman, M. Nashman, and R. Lumia. Model-based vision for car following. In *SPIE Conference on Intelligent Robots and Computer Vision*, volume 2059, pages 420–431, 1993.

[39] I. K. Sethi and R. Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions Pattern Analysis and Machine Intelligence*, PAMI-9 Number 1:56–73, January 1987.

[40] R. Sharma, J. Hervé, and P. Cucka. Dynamic robot manipulation using visual tracking. In *International Conference on Robotics and Automation*, pages 1844–1849. IEEE Computer Society Press, May 1992.

[41] M. J. Swain and M. Stricker. Promising directions in active vision. Technical report, University of Chicago, 1991.

[42] D. Terzopoulos and R. Szeliski. Tracking nonrigid 3d objects. In *Active Vision*, pages 75–89. MIT Press, 1992.

[43] S. M. Thayer. Design and analysis of human-machine and automatic vision systems for cylinder detection and localization. Master's thesis, University of Tennessee, 1993.

[44] M. M. Trivedi. Intelligent machines. In *Science and Technology*, pages 226–229. McGraw Hill, N.Y., 1994.

[45] S. Tugawa, T. Yatabe, T. Hirose, and S. Matsumoto. An automobile with artificial intelligence. In *6th International Joint Converence on Artificial Intelligence*, volume IJCAI-6, pages 839–895, Tokyo, Japan, 1979.

[46] M. Turk, D. Morgenthaler, K. Gremban, and M. Marra. Vitts: A vision system for autonomous land vehicle navigation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume PAMI-10(3), pages 342–361. IEEE, 1988.

[47] R. Wallace, K. Matsuzaki, Y. Goto, J. Crisman, J. Webb, and T.Kanade. Progress in robot road following. In *IEEE International Conference Robotics and Automation*, pages 1615–1621, San Francisco, CA, 1986.

[48] A. M. Waxman, J. LeMoigne, L. Davis, B. Srinivasan, T. Kusher, E. Liang, and T. Siddalinaiah. A visual navigation system for autonomous land vehicles. In *IEEE Transactions on Robotics and Automation*, volume RA-3(2), pages 124–141, 1987.

[49] Wind River Systems. *VXWorks Programmers Guide*, 1990.

[50] Wind River Systems. *VXWorks Reference Manual*, May 1991.

[51] A. Yuille and P. Hallinan. Deformable templates. In *Active Vision*, pages 21–38. MIT Press, 1992.

[52] W. Zhang and F. Bergholm. An extension of marr's signature based edge classification. Technical report, Royal Institute of Technology, Stockholm, 1992.

VITA

Mark Eklund was born in Columbus Ohio on September 14, 1969. In 1976 his family moved to Waverly Tennessee (70 miles west of Nashville) and he was educated from grades one through twelve in public schools located in Waverly. He graduated with his bachelor of science in electrical engineering from the University of Tennessee, Knoxville in 1992. Upon graduation with his master of science, he will begin work as an image processing engineer at Perceptics, a small company located in Knoxville.