2005

Detection of deformable objects in a non-stationary scene

Sherif Azary

COMPUTER SCIENCE MS THESIS

DETECTION OF DEFORMABLE OBJECTS IN A NON-STATIONARY SCENE

Sherif Azary
swa6456@cs.rit.edu

Committee Chairperson: Dr. Roger Gaborski _____

Reader: Dr. Carl Reynolds _____

Observer: Dr. Edith Hemaspaandra _____

Rochester Institute of Technology

Department of Computer Science
102 Lomb memorial drive
Rochester, NY 14623-5608

TABLE OF CONTENTS

LIST OF FIGURES

1 Abstract

Image registration is the process of determining a mapping between points of interest on separate images to achieve a correspondence. This is a fundamental area of many problems in computer vision including object recognition and motion tracking. This research focuses on applying image registration to identify differences between frames in non-stationary video scenes for the purpose of motion tracking. The major stages for the image registration process include point detection, image correspondence, and an affine transformation. After applying image registration to spatially align the image frames and detect areas of motion segmentation is applied to segment the moving deformable objects in the non-stationary scenes. In this paper, specific techniques are reviewed to implement image registration. First, I will present other work related to image registration for feature point extraction, image correspondence, and spatial transformations. Then I will discuss deformable object recognition. This will be followed by a detailed description on the methods developed for this research and implementation. Included is a discussion on the Harris Corner Detection operator that allows the identification of key points on separate frames based on detecting areas in frames with strong contrasts in intensity values that can be identified as corners. These corners are the feature points that are comparable between frames. Then there will be an explanation on finding point correspondences between two separate video frames using ordinal and orientation measures. When a correspondence is made, the data acquired from the image correspondence calculations will be used to apply translation to align the video frames. With these methods, two frames of video can be properly aligned and then subtracted to detect deformable objects. Finally, areas of motions are segmented using histograms in the HSV color space. The algorithms are implemented using INTEL's open computer vision library called OpenCV. The results demonstrate that this approach is successful at detecting deformable objects in non-stationary scenes.

2 Related Research

Image registration can be implemented in many different ways but usually involves multiple steps. The first step to image registration is feature point extraction. By finding points of interest that are comparable between two similar images, a correspondence can be determined and the images can be properly registered. One common way to detect feature points in an image is to analyze the image and find corners. There are many corner algorithms that exist today, and although they usually sacrifice performance because of intense computational demands, they are still widely used because of their accuracy. Yuma Kudo, Takeshi Kajiyama, and Wataru Mitsahushi (2003) propose a new technique to extract corners that can be used for image correspondence. The proposed method is called SUSAN (Smallest Univalue Segment Assimilation Nucleus). Corners are defined as points on an image where the direction of an edge changes by a large amount. Let $f_\theta(r,\theta)$ be the angular gradient of the image intensities in polar form, and let $f_r(r,\theta)$ be the radial gradient of the image intensities in polar form. This is shown in Figure 1 below.

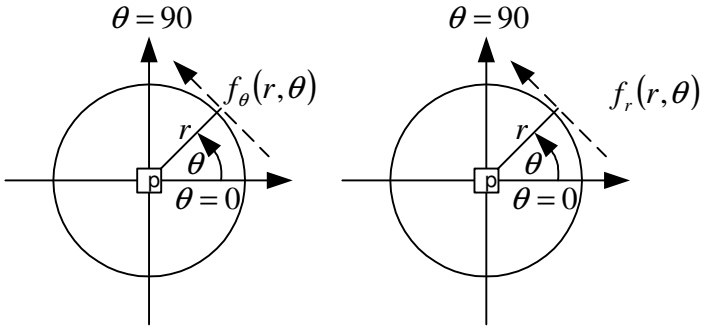

Figure 1: (L) Angular Gradient of Image Intensities and (R) Radial Gradient of Image Intensities

The center pixel 'p' is the origin. When accumulating the modulus of the angular gradient from pixel 'p' in the direction of θ, a high accumulation indicates a detected edge. The equation for the modulus accumulation measure is given in Equation 1 below. However, a high accumulation of the angular gradient will also result from noise or edges that do not intersect pixel 'p'. For that reason it is necessary to accumulate the measure of the radial gradient shown in Equation 2 below. A high radial gradient accumulation indicates the presence of noise and/or edges that do not pass through pixel 'p' while a low

radial gradient accumulation indicates the opposite conclusion. Knowing this, the measure for edge detection indicated by $J(\theta)$ can be described by Equation 3.

Equation 1: $$J_\theta(\theta) = \sum_{r=1}^{R} |f_\theta(r,\theta)|$$

Equation 2: $$J_r(\theta) = \sum_{r=1}^{R} |f_r(r,\theta)|$$

Equation 3: $$J(\theta) = \begin{cases} J_\theta(\theta) - J_r(\theta) for (J_\theta(\theta) - J_r(\theta) > 0) \\ 0 for (J_\theta(\theta) - J_r(\theta) \le 0) \end{cases}$$

This demonstrates the process of edge detection. Knowing that a corner is a point on an image at which the direction of an edge changes by a large amount, corners can be detected at pixel 'p' knowing that $J(\theta)$ becomes larger at more than one direction of θ. Linear edges indicate non-existent corners and can be eliminated when knowing that a linear edge will have a large value of $J(\theta)$ at two opposite directions that differ by 180°. This technique for corner detection is not used in the algorithms implemented for this research but it demonstrates one of many different ways for feature point extraction.

Similarly, David Lowe (2004) offers a different method for feature point extraction and continues on to describe image correspondence, the second stage for image registration. Many people are researching different approaches to image registration and are looking into ways of extracting distinctive invariant features between different views of an object or a scene to perform matching. David Lowe discusses such techniques for recognizing invariant features to simplify image correspondence. To find a match of different views, Lowe states that features from different images must be found that are invariant to image scale, rotation, affine distortion, and illumination. To be efficient, only features that pass an initial test are then passed on to more computational expensive operations for further analysis. The stages involved to find a correspondence between two images invariant to such features following this cascading filtering approach are (1) scale-space extrema detection, (2) keypoint localization, (3) orientation assignment, (4) and keypoint descriptor.

The scale-space extrema detection is the first stage mentioned by David Lowe that scans each image over a range of scales to identify points of interest that are repeatedly recognizable. The function used to detect such locations in an image is called the scale space function. Using a Gaussian function $G(x, y, \sigma)$ convolved with an image $I(x, y)$ at different scales σ, a difference of Gaussian function convolved with the image is used to compute the difference of scales. This scale space function is given by Equation 4 below for which the symbol '*' indicates convolution. Equation 5 shows the difference of Gaussian (DoG) function. Observing the image over different Gaussian convolutions, features that are stable across all possible scales can be extracted and analyzed further since they appear invariant to scale.

Equation 4: $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$

Equation 5: $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$

After scale-space extrema detection is used to detect points of interest in different comparable images, keypoint localization is used to analyze neighboring pixels in terms of location, scale, and ratio of principal curves in order to eliminate detected keypoints that are sensitive to noise and are poorly localized along edges. This differs from the method taken by Yuma Kudo, Takeshi Kajiyama, and Wataru Mitsahushi (2003) for which a high radial gradient accumulation from the pixel of interest in the direction θ indicates the presence of noise and/or edges that do not pass through the pixel of interest. However, both methods intend to eliminate weak keypoints affected by noise. Next, the remaining detected keypoints are each assigned an orientation to achieve an invariance to image rotation. A scale of each keypoint is used to select the smoothed image $L(x, y)$ for which computations will appear scale-invariant. For each image at the selected scale the gradient magnitude and the orientation is calculated as a function of the neighboring pixels as demonstrated in Equation 6 and Equation 7 respectively.

Equation 6: $$m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)+L(x,y-1))^2}$$

Equation 7: $$\theta(x,y)=\tan^{-1}\left(\frac{(L(x,y+1)-L(x,y-1))}{(L(x+1,y)-L(x-1,y))}\right)$$

From this information an orientation histogram is created for which each sample point is weighted by factors including the gradient magnitude. High peaks in the orientation histogram are selected and defined as keypoints invariant to orientation. Finally, a keypoint descriptor is used to describe the remaining keypoints by image gradients that are invariant to shape distortion and change in illumination.

This cascading filtering approach recommended by David Lowe is not used in this research for detecting motion in panning video because video frames are not expected to drastically vary in scale. When analyzing a video file for which the frame rate is 30 FPS, 30 frames will have been analyzed for one second of time, and ignoring scale-invariance computations will improve system performance. This and other restrictions and assumptions will be discusses later on in this paper.

Image correspondence is often used for template matching and pattern recognition. A pattern or a template is compared between images and their location and orientation are determined before applying the registration. Point mapping can be used for spatial transformation, and the general approach is to determine similar points of interest and aligning those points using rigid and affine transformations. These keypoints play an important role because the accuracy of these point comparisons between images will indicate the success of the image registration process. Although it is not desirable to use keypoints that are invariant to scale in this research, it is useful to find keypoints that are invariant to illumination because lighting conditions can change frequently between video scenes because of shadows and sunlight in outdoor and indoor scenes. Dinkar Bhat and Shree Nayar (1998) propose a unique idea that involves comparing the order of intensity values between points of interest on separate images. They define the term ordinal measure as the measure of correspondence of points based on the similarity of the order of intensity values around a point on two separate images. Ordinal measures can be used to accurately find correspondences between video frames, because this technique seems accurate and is invariant to illumination, which is desirable as previously stated.

Suppose the same corners are detected on two separate outdoor images. One image may have been taken during a sunny day and the other image may have been taken on a cloudy day. A comparison of the intensity values at the detected corners may not be comparable because of the different lighting conditions. A method of correspondence based on invariance to illumination will be helpful for the image registration process. Rather than comparing the intensity values of the corners to achieve a correspondence, a permutation rank is assigned around each corner of a user specified window size. Suppose a 3x3 window is analyzed around each detected corner in an image. An example of a 3x3 window around a corner point is shown in Figure 2 below. This means that 9 permutation ranks will be assigned for each corner as shown in Figure 3 below for this example.

10	30	70
20	50	80
40	60	100

Figure 2: 3x3 Window of a Corner Point

1	3	7
2	5	8
4	6	10

Figure 3: Rank Ordering

A composition permutation based on two sets of corner rank information can be determined from Equation 8. Equation 9 is used to determine the distance vector that defines the distance between the ranks.

Equation 8: $s^i = \pi_2^k, k = \left(\pi_1^{-1}\right)^i$

π_1 =Rank of image 1 data, π_2 =Rank of image 2 data

π_1^{-1} = Inverse Permutation of π_1

Equation 9: $d_m^i = i - \sum_{j=1}^{i} J(s^j \le i)$

i = Element index in window

s^j = Composition Permutation at j = $\pi_2^{\pi_1 index}$

$J(x)$ = Indicator function returning 1 when true and 0 when false

This equation returns a distance measure vector that can be put into a measure of correlation as shown in Equation 10.

Equation 10: $\aleph(I_1, I_2) = 1 - \frac{2\max_{i=1}^{n} d_m^i}{\left\lfloor \frac{n}{2} \right\rfloor}$

$\aleph(I_1, I_2)$=Measure of Correlation between image 1 and image 2

$\max_{i=1}^{n}$ =Maximum value in the distance vector

n = Number of elements in the window

The ordinal measure returns a value of 1 for perfect correlation if the order of the intensity values of a corner on one image exactly matches the order of the intensity values of a corner on the second image. A -1 correlation measure indicates perfect inverse correlation. Because the order of the intensity values around a point are analyzed rather than the actual values of the intensity values, this method is invariant to intensity value which is an advantage for comparing images with different lighting conditions. The steps and example to obtain the ordinal measure are explained in section 4.3.6.2: Steps to Calculate Ordinal Measure.

After feature point detection and image correspondence, transformations are the third major stage of image registration that is needed to register the images. Lisa Gottesfeld Brown (1992) discusses variations in images and appropriate image registration techniques that can be applied as a framework to address image registration transformation problems. The first image variation type is spatial alignment. Two images when averaged may not align in some manner because of differences in rotation, scale, and translation. A spatial transformation is needed to overlay or combine the

separate comparable images. The approaches of interest are rigid and affine transformations. The rigid transformation handles retaining the relative shape and size of objects in images that usually vary by translation and/or reflection (Skowhegan Area High School, 2005). This is typically done by recognizing the same object in two images and recognizing their coordinates. By knowing this information, cross-correlation can be applied and it is then possible through image manipulation to either overlay or expand the input images to a larger output image as demonstrated in Figure 4 and Figure 5 below.

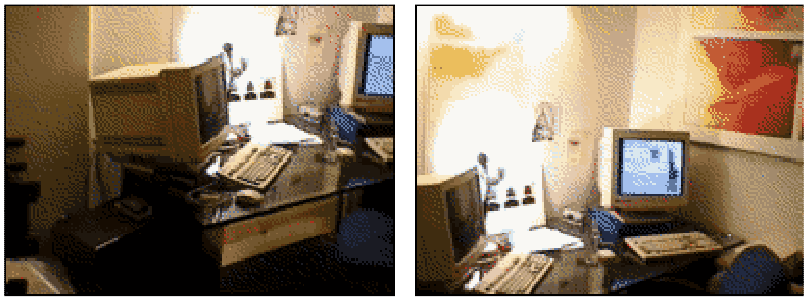

Figure 4: Two Images Taken At Different Angles (Paul Haeberli and Eyal Ofek, 2005)

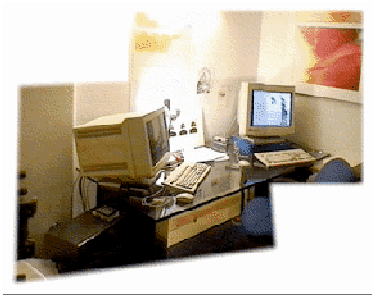

Figure 5: Merged Images (Paul Haeberli and Eyal Ofek, 2005)

The affine transform can tolerate more complicated distortions, by mapping straight lines from two separate images, and can handle differences in images including different angles, different positions, and shearing. Shearing is a deformity for which symmetric objects do not appear symmetric because of the angle and position the image was taken from. The aspect ratio varies, and this is a common and difficult problem to address. The solution is to scale each axis independently and apply rigid-body transformations to adjust for the shearing. The approach taken in the motion detection in non-stationary video research only takes into account translation while ignoring shearing and rotation. Shearing and rotation are important in accurately performing image

registration, but their significance in detecting moving objects in stationary and non-stationary video, with minor changes between frames, is minimal and computationally expensive.

Another variation between similar images is lighting and atmospheric conditions. This is of particular interest because one approach initially considered and implemented for this research was comparing intensity levels between frames of video. However, this technique was not very successful because shadows and general lighting conditions were changing and affected the correspondence results. Ordinal measures offer a solution invariant to illumination as previously mentioned.

All these variations are distortions that can be addressed by applying morphological transformations. The approach for these transformations is to make adjustments to each image independently to remove distortions and variations that will reduce the difficulty in cross-correlation between them. Only minor image enhancements were used to remove distortions in this research.

After applying image registration and eventually determining and extracting the objects in motion in stationary and non-stationary video scenes, it is desirable to segment the entire moving object and possibly even identify and classify it. It is important to understand the problems faced when attempting to create a system that can accurately recognize patterns in images. When looking at Figure 6 of the pictures of the deer, we as humans understand and recognize that these are pictures of deer. However, a computer vision system that can accurately identify such examples as deer objects is a difficult task because images can vary by position, size, orientation, illumination, and other distortions. With limited memory and processing resources, an image understanding system must be able to recognize patterns regardless of such factors. Miao Zhenjlang and Yuan Baozong (1993) propose using neural networks and Zernike moments to design an image understanding system that can recognize ten country maps and ten animals.

Figure 6: Deer Pictures (www.muskokagardens.ca/ photos/deer.jpg) & (http://www.wjsteele.com/blogimages/deer.jpg)

A neural network is a parallel distributed processing system with simple processing elements and many interconnections. Neural networks have high robustness and fault tolerance and for that reason are proposed by these authors to recognize image patterns. The input parameters to the image understanding neural network are Zernike moments. Moments are descriptions of image content that describe geometric information about images. They can describe image properties that are invariant to rotation, scale, and orientation. Zernike moments were selected as the form of inputs to the neural network because they are optimal for image analysis when it comes to sensitivity to image noise, information redundancy, and capacity for image representation. By inputting Zernike moments into an image understanding neural network, many different objects can be classified and recognized.

There are four main stages to the design explored in this article. The first stage is image acquisition, which is the process of converting a scene to an image. The second stage is preprocessing which is the manipulation of image processing techniques to remove noise and enhance the image. The third stage is feature extraction, which is the process of extracting useful information from the image that may be useful for identifying the pattern in the image. The final stage is classification, which is the process of assigning the recognized image to a database of existing classes that in this case are restricted to ten country maps and ten animals. Classification is not used in this research but by detecting deformable objects in non-stationary scenes classification can be used as the next stage after segmentation for identifying objects.

3 OpenCV

3.1 BACKGROUND ON OPENCV

OpenCV is an open source computer vision library developed by Intel. The project code is written in C++ and aims at real time performance. Originally, Matlab was used for the earlier implementation but performance became a major issue. OpenCV offers a real-time solution to the image registration problem with its highly optimized low-level functions. OpenCV provides many computer vision primitives that can be used for feature detection, shape analysis, morphological operations, and object segmentation and recognition. Some example areas of interest include object identification, segmentation and recognition, facial recognition, gesture recognition, and motion tracking. The OpenCV software used for this project runs on a personal desktop computer with a 750 MHz processor on a Windows XP platform. More can be learned about OpenCV at http://www.intel.com/technology/computing/opencv/overview.htm.

3.2 FLIPPING AVI FILES

In this research it was convenient to input AVI files into the software application instead of inputting real-time video during the implementation and development of the software. The input video files needed to have each AVI video frame flipped before being analyzed. OpenCV video frames need to be flipped because OpenCV reads in AVI files upside down by default. It is unclear why this is the case but Internet forums suggest that the OpenCV library was originally written on Linux with drivers that deliver buffers where the first byte in the buffer is equal to the upper-left most pixel in the image. In Windows/OpenGL the pixel data begins with the first component on the lower-left most pixel so the y-coordinate system needs to be flipped. If the video frames are not flipped, then the output appears upside down.

4 Detection of Deformable Objects

4.1 INTRODUCTION

A video clip called "HarryTheDog.avi" will be referenced frequently when explaining the steps involved for detecting deformable objects. This video clip shows a dog named Harry walking through a forest type setting with trees and hills. The video clip was captured with a hand held panning video camera. The dog occasionally completely disappears from view while walking behind trees or walking down a bank. There will also be references to basic geometric shapes to further help demonstrate the deformable object detection process. Assuming *n* represents a single frame in a video sequence from one to the total number of frames, image registration in this project is used to expand two input images extracted from video frames *n* and *n+1* into one larger output image that can be properly aligned and subtracted to detect motion. Then segmentation is used to analyze the areas of motion and properly segment the deformable object. The following are the major stages that were investigated and implemented to carry out the motion detection task and segmentation. A block diagram is provided below in Figure 7 showing the workflow model.

1. Read in video file in AVI format
2. For *n=1*; *n < # of video frames - 1*; *n++*
    a. Extract frame *n* and frame *n+1* from the passed in video file
    b. Calculate corners in frame *n* and *n+1*. Plot the corners of the frame and save results into a corner video file. The corner detection scheme used is the Harris Corner Detection. This is the first stage for which corners in each video frame are detected according to Chris Harris's corner detection algorithm. Save the set of coordinates for the detected corners for each image.
    c. Calculate the correspondence between frames *n* and *n+1* from the saved detected corner coordinates of frames *n* and *n+1*. The image correspondence method used is a combination of ordinal measure and orientation measure. Save the coordinate pairs that are believed to be

correspondences. The correspondence is plotted and saved into an image correspondence video file.

d. Once a correspondence is determined, the two images are aligned using translation. When properly aligned, average the overlapping pixels to create the registered image and subtract the overlapping areas to create the difference image. The difference of the aligned images should be a black region representing the background with obvious responses in white of moving foreground objects if motion exists. Save the registered images and difference images.

e. Apply segmentation to the detected foreground using histograms in the HSV color space.

3. Repeat step 2 until all frames have been analyzed in the video sequence.

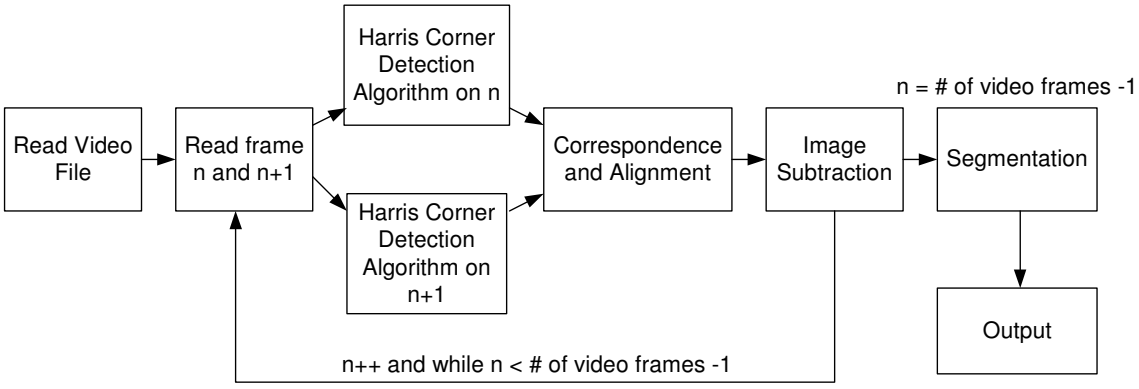

Figure 7: Workflow Model for the Detection of Deformable Objects in Non-Stationary Scenes

4.2 WHY USE IMAGE REGISTRATION?

It may be unclear at this point why image registration is necessary to detect deformable objects in non-stationary scenes. If the scenes are guaranteed to be stationary then simple subtraction can be performed between video frames, and what does not appear as black can be considered motion. However, the same rules do not apply for non-stationary video scenes because a subtraction between frames that are not aligned would produce a lot of motion responses that are not useful and correct when detecting motion. The following example demonstrates this point.

Figure 8: Simple Image Subtraction on Non-Stationary Video Scenes

4.3 DETAILED APPROACH

In this section, feature extraction, image correspondence, image differencing, and segmentation are discussed in more detail with supporting information outlining the implemented project. For the first section, there will be a presentation on edges and corners followed by the Harris Corner Detection theory and algorithm. The second section will discuss the selected image correspondence methods. The third section will discuss manipulating the data obtained from the image correspondence result to align and register two frames of video as well as demonstrate how the difference of the aligned frames will show motion detection. Finally, segmentation in the HSV color space using histograms will be discussed.

4.3.1 RGB Images

Light is composed of energy waves that compose the color spectrum shown in Figure 9 below. The dominant colors in the color spectrum are red, green, and blue. Different levels of these colors can be combined to form other colors. An image is represented as an array of pixels with a width *w* and a height *h*. RGB images are images with three overlapping planes that define the amount of red, green, and blue needed to identify each pixel in the image and gives the appearance of color images as that shown in Figure 10 of a sample video frame from the "HarryTheDog.avi" video file.

Figure 9: RGB Color Spectrum (RGB World Inc., 1996, p. 1)

4.3.2 Gray-Scale Images

It can be time consuming to process color images since three individual planes need to be processed. To save time and improve performance, images can be converted to gray-scale when applicable. Gray-scale images are images represented on one plane for which each pixel is assigned a value from 0 to 255 where 0 is black and 255 is white. Values in between 0 to 255 represent a shade of gray. These different shades of gray can give an image the appearance of a black and white image. To convert an RGB image to a

gray-scale image, Equation 11 can be applied to each pixel in the RGB image. Figure 10 also shows the gray-scale image of the RGB image.

Equation 11: $Y = (0.299 \times R) + (0.587 \times G) + (0.114 \times B)$

Figure 10: Frame 511 from "HarryTheDog.avi" as (L) RGB Image and as (R) Gray-Scale Image

4.3.3 Spatial Filtering and Convolution

Spatial filtering is a basic image-processing concept that involves applying a function or an operator directly on the pixels of an image. A filter mask is applied to each point in the image and the response of the image is set as the new value at that point. The response is the sum of the input pixels within the mask where the weights are the values of the filter assigned to each pixel in that mask (Quantitive Imaging Group, 1999). The process of producing a new response from blending a function with an image is called convolution. The following expression shows the response of a filtering mask on an image f of size $M \times N$ with a filter mask of size $m \times n$ (Gonzalez & Woods, 2002, p.117).

Equation 12: $$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$a = \frac{m-1}{2},\ b = \frac{n-1}{2},\ x = 0,1,2....M-1, \text{ and } y = 0,1,2....N-1$$

Different masks can be used to obtain different responses. Examples of filters used to detect isolated points and lines are shown in Figure 11.

-1	-1	-1
-1	8	-1
-1	-1	-1

-1	-1	-1
2	2	2
-1	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

Figure 11: (L) Point Detection Filter, (M) Horizontal Line Detection Filter, (R) Vertical Line Detection Filter

4.3.4 Gaussian Filter

A Gaussian filter is a convolution operator that blurs images and removes detail and noise. It is also called a smoothing filter. The output response of a Gaussian filter is the average of the pixels contained in the neighborhood of the filter mask. Initially one might think that there would be no advantage to blurring an image but in fact this operation is quite useful. The filter acts as a low pass filter and preserves edges while eliminating noise. The Gaussian filter is very useful when detecting edges and corners as will be explained later. Figure 12 demonstrates the effects of a simple smoothing filter

Figure 12: Frame 511; (L) Original Gray Image and (R) Smoothed Imaged

1	1	1
1	1	1
1	1	1

Figure 13: Smoothing Filter

4.3.5 Edges, Corners, and the Harris Corner Detector

For feature point extraction and registration, there are two important requirements. First, points need to be extracted consistently between images in order to guarantee a correspondence or otherwise the images cannot be successfully registered. Second, the neighboring pixels of the detected feature points need have enough information to describe the region so that points between images can be matched. This section discusses the first requirement.

One important task of image registration is to be able to calculate control points or points of interest that can be used to apply image correspondence in order to match up the images. One common method to determine such control points is to detect the corners in an image. There are many corner algorithms that exist today and although they usually sacrifice performance because of intense computational demands they are still widely popular because of their accuracy. Before explaining corner detection, we need to have a basic understanding of edges in images. Edges are defined as locations in images with a strong intensity contrast in one direction (Fisher, Perkins, Walker, and Wolfart, 1994). An example of an edge is demonstrated in Figure 14 below.

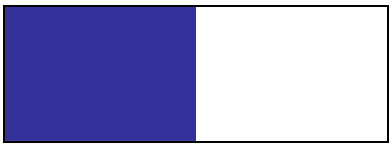

Figure 14: An Edge

Edges represent object boundaries and by representing images by their boundaries information is efficiently reduced without sacrificing data loss. A corner is very similar to an edge except a corner is a location in an image with a strong intensity contrast in more than one direction. Figure 15 below shows a basic corner in an image.

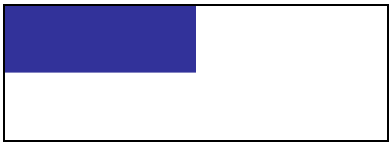

Figure 15: A Corner

An edge corresponds to strong illumination gradients, and by calculating the derivative of an image we can identify edges. Please refer to Figure 16 below for a visual of the derivative responses to an edge of a 1-D image. The initial function represents an edge in an image in a single direction. Notice the function response when taking the first and second derivatives of the function. These responses highlight detectable points of interest in images.

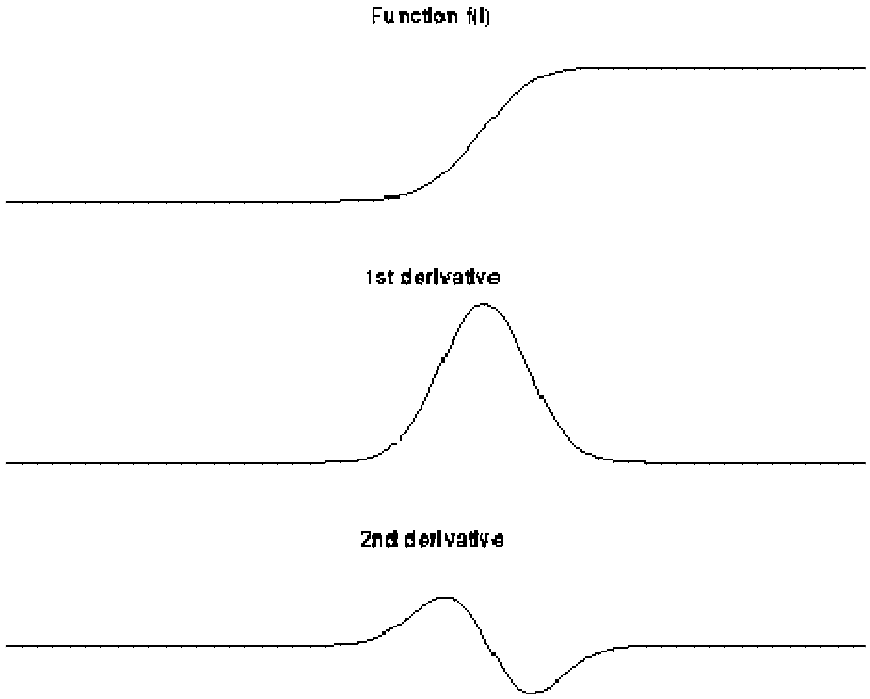

Figure 16: Derivative Responses to an Edge

(Image obtained from http://www.cee.hw.ac.uk/hipr/html/edgdetct.html)

A 2-D image is a function of two variables, X and Y (Quantitative Imaging Group, 1999). For this reason, the derivative of an image needs to be taken in the X and Y directions in order to find all horizontal and vertical edges. To obtain the first derivative of an image, the image needs to convolve with the following basic derivative filters given below in Figure 17.

$$[h_x] = [1 \quad 0 \quad -1] \qquad [h_y] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Figure 17: The Horizontal and Vertical First Derivative Filters

h_x is the horizontal derivative filter, h_y is the vertical derivative filter

Now that there is an understanding of edges and corners, the Harris Corner detector is introduced. Chris Harris and Mike Stephens developed this operator in 1988 for a system that could analyze motion and interpret an environment from a single camera (Parks, 2005). The Harris Corner Detector operates with the following steps as described by Donovan Parks and Martial Hebert. Please also refer to the provided simple geometric figures for reference.

1. Apply convolution on the input image with a Gaussian filter. The Gaussian filter is a smoothing operator that is used to blur images and remove detail and noise. The output is a weighted average of each pixel's neighborhood and this technique is generally used to provide smoothing and preserve edges (University of Edinburgh, 2005). Applying the Gaussian filter will ensure that the operator is rotationally invariant and will ensure the image noise is reduced (Pollefeys, 2000). The input *I* should be treated as a gray level intensity image convolved with a Gaussian filter:

$$\otimes = \text{Convolution Operator}$$
$$G = \text{Gaussian filter}$$
$$I = G \otimes Input$$

2. Compute the X and Y derivatives of the smoothed input image *I*. Applying convolution on the input image with the horizontal and vertical derivative filters hx and hy does this. Figure 18 shows the input test image and its X and Y derivative images.

$$I_x = h_x \otimes I, \quad I_y = h_y \otimes I$$

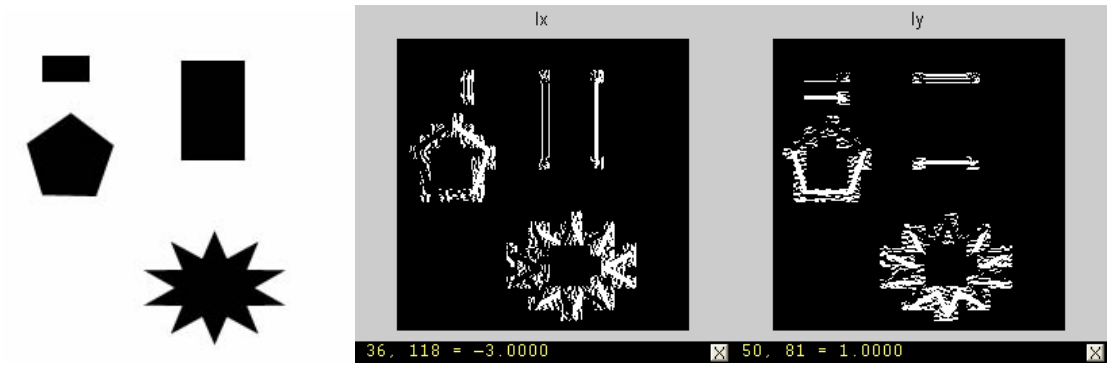

Figure 18: Input Image (L), X-Derivate Image (C), and Y-Derivate Image (R) Respectively

3. Compute the products of the derivatives at every pixel. The output is shown in Figure 19 below.

$$I_{x2} = I_x \bullet I_x, \quad I_{y2} = I_y \bullet I_y, \qquad I_{xy} = I_x \bullet I_y$$

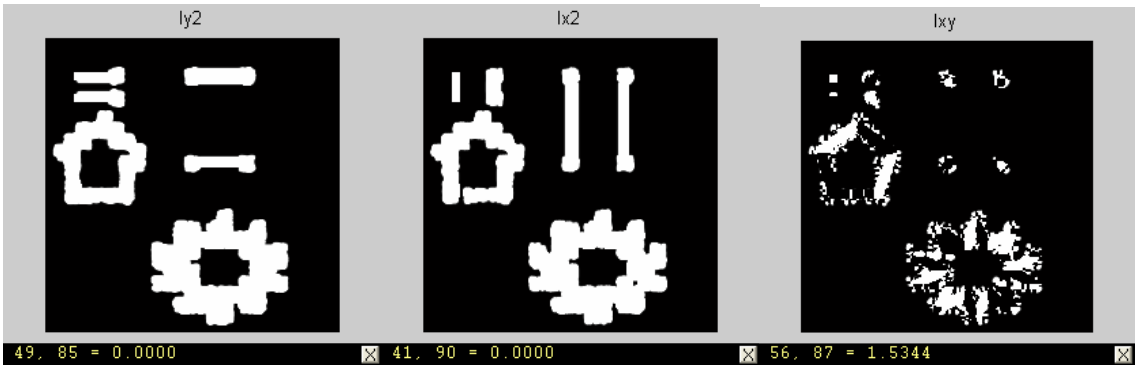

Figure 19: (L) X-Derivate Squared, (M) Y-Derivate Squared, and (R) Product of X and Y Derivate

4. For each pixel (x,y) in the smoothed image, calculate the autocorrelation matrix M. This constructs the cornerness map by calculating the cornerness measure C(x,y) for each pixel in the image. If at a certain point the two eigenvalues of the matrix M are large, then a small motion in any direction will cause an important change of grey level. This indicates that the point is a corner. The corner response function is given by C(x,y), the response of the detector at every pixel in the image (Pollefeys, 2005). Please also refer to Figure 20 for the cornerness map.

$$M = \begin{bmatrix} I_{x2} & I_{xy} \\ I_{xy} & I_{y2} \end{bmatrix}$$

$$k = Constant = 0.04 \text{ as recommended by Harris}$$

$$det = \lambda_1 \lambda_2$$

$$trace = \lambda_1 + \lambda_2$$

$$C(x, y) = \det(M) - k(trace(M))^2 = (I_{x2} \times I_{y2}) - (I_{xy})^2 - k(I_x + I_y)^2$$

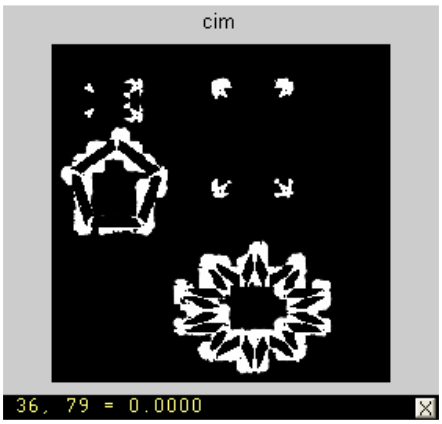

Figure 20: Cornerness Map

5. Apply thresholding on C(x,y) to eliminate weak corner responses. The output is a map indicating positions of each detected corner. Figure 21 shows the detected corners marked in red mapped over the original image.

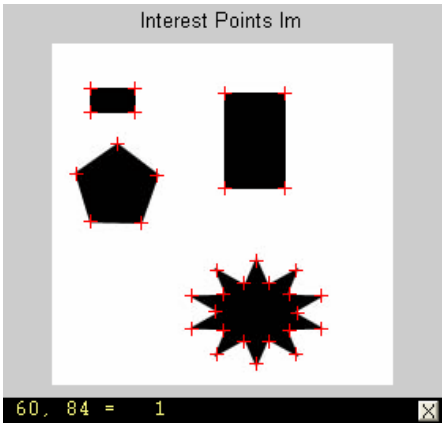

Figure 21: Detected Corners Mapped on Original Image

Figure 22 shows the detected corners from the video clip of Harry the dog using this implementation. Notice the consistency between consecutive frames of the detected

corners. The images are shown for frames 221, 222, 245, 246, 511, and 512 from the dog video clip. Because many of the detected corners appear consistent between consecutive frames, it is possible to analyze the consistent data to calculate correspondences between consecutive frames.

a *b*
c *d*
e *f*

(a) Frame 221,
(b) Frame 222,
(c) Frame 245,
(d) Frame 246,
(e) Frame 511,
and (f) Frame 512

Figure 22: Detected Corners Mapped on Original Frames

David Lowe's method of defining keypoints was not used for this paper. The reason is the cascading filter approach for defining keypoints that are invariant to scale, rotation, and illumination were not necessary when analyzing video frames. A typical video clip used for this research can run anywhere between 20 to 30 frames per second. That means that consecutive frames are analyzed that vary in time by 33ms to 50ms. Since frame images do not appear to vary much by scale, rotation, and illumination, it seems time consuming and computationally expensive to analyze points of interest that are invariant to such features. Recall the first test called the scale-space extreme for

which each image is analyzed at multiple scales using different Gaussian sizes to identify points of interest that are repeatedly recognizable. Observing the image over different Gaussian convolutions, features that are stable across all possible scales can be extracted and analyzed further since they appear invariant to scale. It made more sense for this type of application to simply extract corner points using the Harris corner detector that are not required to be invariant to scale.

4.3.6 Image Correspondence

The second requirement for image registration is that neighboring pixels of the detected feature points need to have enough information to describe the region so that points between images can be matched. Take for example the two similar images shown below in Figure 23. These two images differ slightly in horizontal and vertical translation. Once corners are detected in each image, how is it possible to find a correspondence between the two images? There may be cases where corners on one image may not be detected on the other image. It's also possible that corners on one image are not even visible on another image. There is no guarantee that the number of corners will be the same or that all corners will be detected.

During early implementation, David Lowe's paper was investigated for correspondence. He discusses keypoint descriptors. Detected points were labeled to identify the gradient magnitude and orientation for each point and this data is compared between two images to find a correspondence. The original intent in David Lowe's paper was to specify keypoint descriptors for different scales to find key points that are invariant to scale. Since there is no concern for scale invariance in video frames, a single constant Gaussian filter was applied to each frame. By ignoring multiple scales, performance is improved because less computation is required. Each detected corner point is labeled in terms of gradient magnitude and orientation. For each image at the selected scale the gradient magnitude and the orientation is calculated as a function of the neighboring pixels as demonstrated in Equation 6 and Equation 7.

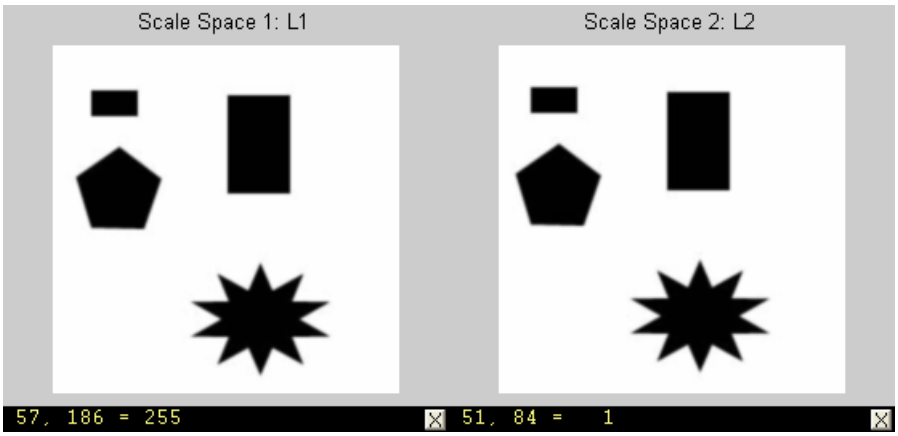

Figure 23: Two Similar Images That Differ in Horizontal and Vertical Translation

As previously mentioned, part of achieving a correspondence is making an orientation assignment for each detected corner on the image. The assigned orientation is based on local image gradient directions and this achieves an invariance to image rotation. Still and moving video data is not subject to rotation for this research, so the orientation assignments and gradient magnitudes were expected not to vary by much between consecutive frames. The orientation and the gradient magnitude assignment to each detected corner on the image is what constituted the key descriptor mentioned in David Lowe's paper. The steps to calculate the gradient magnitude and the orientation of each detected corner are listed below:

4.3.6.1 Steps to Calculate Gradient Magnitude and Orientation

1. Get the scale space of the image. The scale space is defined as a function L(x,y) that is produced from the convolution of the original image with a Gaussian filter of a specified scale.

$\otimes$ = Convolution Operator

G = Gaussian filter

I = Input Image

$$L(x, y) = G(x, y) \otimes I(x, y)$$

2. For each image sample at the scale specified by the Gaussian filter, calculate the gradient magnitude, m(x,y), and orientation θ(x,y). Notice that both the gradient magnitude and the orientation values are a function of neighboring points to the detected corner. Once each corner is labeled, corner points from two separate images can be compared.

Figure 24 shows two test images that had extracted corner points that were assigned a gradient magnitude and orientation. The values of the calculations are shown and the determined correspondences are shown in the tables below. The data for the gradient magnitude and the orientation are compared and data with minimal differences are recognized as correspondences. However, when applying this method of correspondence on larger images with more complex scenes, it was noticed that many corner points were corresponding that were not related and there was too much similar data between corner points on separate images. Furthermore, real images from video clips were having difficulty corresponding because of the restriction imposed that only one scale and one Gaussian filter size were to be used to analyzed corner points by gradient magnitude and orientation. Simple controlled scenes such as the example in Figure 24 correspond well, but that was not the case for other more complex scenes. It seemed necessary to add additional methods and restrictions to obtain better correspondence results.

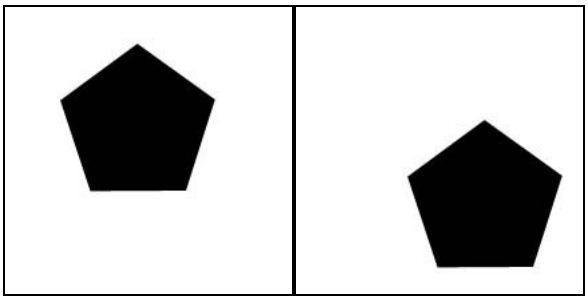

Figure 24: Test Images for Correspondence

	Image 1	Image 2
Width	216	216
Height	216	216
Depth	8	8
Channels	1	1

Table 1: Test Image Information

		Coordinate	L(x,y)	L(x+1,y)	L(x-1,y)	L(x,y+1)	L(x,y-1)	Magnitude	Orientation
Image 1	**Point 1**	(135,137)	0	204	0	204	0	288.50	45
	Point 2	(65,137)	8	0	44	0	44	62.23	45
	Point 3	(42,70)	77	0	255	0	255	360.62	45
	Point 4	(156,70)	0	128	0	128	0	181.02	45
	Point 5	(99,29)	0	34	42	34	42	11.313	45
Image 2	**Point 1**	(177,194)	30	189	0	189	0	267.29	45
	Point 2	(107,194)	9	2	71	2	71	97.58	45
	Point 3	(84,127)	91	0	235	0	235	332.34	45
	Point 4	(198,127)	14	108	0	108	0	152.74	45
	Point 5	(141,86)	0	18	61	18	61	60.81	45

Table 2: Magnitude and Orientation Data

Image 1	Image 2
Point 1	Point 1
Point 3	Point 3
Point 4	Point 4
Point 5	Point 5

Table 3: Correspondence Results

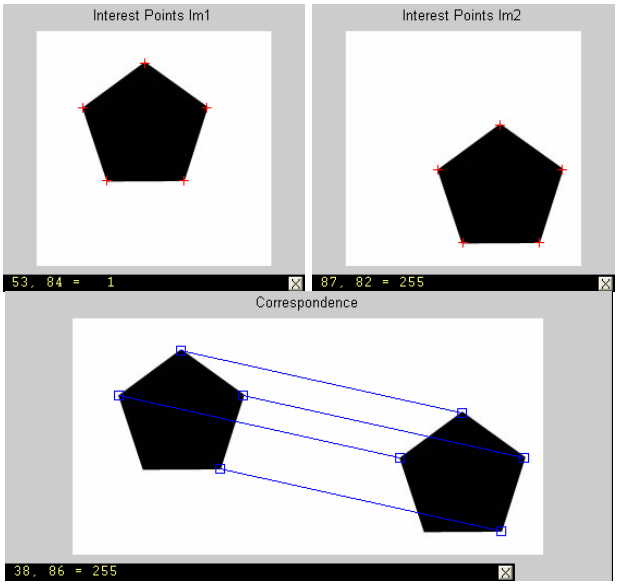

Figure 25: Correspondence Results

4.3.6.2 Steps to Calculate Ordinal Measures

To address the need for additional keypoint description information, the ordinal measures as explained by Dinkar Bhat and Shree Nayar (1998) were also used. Recall that Dinkar Bhat and Shree Nayar proposed a unique idea that involves comparing the order of intensity values between points of interest on separate images. The ordinal measure is the measure of correspondence of points based on the similarity of the order

of intensity values around a point on two separate images. It will be helpful to show an example of the manipulation of Equation 8, Equation 9, and Equation 10. Take for example the measure of correlation of the following two corner points found on two separate images.

10	20	30
50	40	70
60	90	80

90	60	70
50	40	80
10	30	20

Figure 26: 3x3 Corner Point Windows on two Separate Images

The 3x3 windows show the intensity values in the window around detected corner points on two separate images. We first organize the data into two separate vectors and then assign their corresponding ranks.

I_1 = [10,20,30,50,40,70,60,90,80], I_2 = [90,60,70,50,40,80,10,30,20]

π_1 = [1,2,3,5,4,7,6,9,8] π_1 = [9,6,7,5,4,8,1,3,2]

Now we can solve for the composition permutation based on the rank information using Equation 8. Recall that as $s^i = \pi_2^k, k = \left(\pi_1^{-1}\right)^i$ where i is the element index in the window, π_1 is rank data for the first corner point, π_2 is the rank data for the second corner point, and π_1^{-1} is the inverse permutation of π_1.

$$s^1 = \pi_2^k, k = \left(\pi_1^{-1}\right)^1 \rightarrow k = 1 \rightarrow \pi_2^1 = 9$$

$$s^2 = \pi_2^k, k = \left(\pi_1^{-1}\right)^2 \rightarrow k = 2 \rightarrow \pi_2^2 = 6$$

$$s^3 = \pi_2^k, k = \left(\pi_1^{-1}\right)^3 \rightarrow k = 3 \rightarrow \pi_2^3 = 7$$

$$s^4 = \pi_2^k, k = \left(\pi_1^{-1}\right)^4 \rightarrow k = 5 \rightarrow \pi_2^5 = 4$$

$$s^5 = \pi_2^k, k = \left(\pi_1^{-1}\right)^5 \rightarrow k = 4 \rightarrow \pi_2^4 = 5$$

$$s^6 = \pi_2^k, k = \left(\pi_1^{-1}\right)^6 \rightarrow k = 7 \rightarrow \pi_2^7 = 1$$

$$s^7 = \pi_2^k, k = \left(\pi_1^{-1}\right)^7 \rightarrow k = 6 \rightarrow \pi_2^6 = 8$$

$$s^8 = \pi_2^k, k = \left(\pi_1^{-1}\right)^8 \rightarrow k = 9 \rightarrow \pi_2^9 = 9$$

$$s^9 = \pi_2^k, k = \left(\pi_1^{-1}\right)^9 \rightarrow k = 8 \rightarrow \pi_2^8 = 3$$

The composition permutation is now S=[9,6,7,4,5,1,8,2,3]. Now we can use Equation 9 to determine the distance vector used to define the distance between the ranks. Recall Equation 9 as $d_m^i = i - \sum_{j=1}^{i} J(s^j \le i)$ where i is the element index in the window, s^j is the composition permutation, j is the index of the first rank data back on the second rank data, and $J(x)$ is the indicator function returning 1 when true and 0 when false.

$$d_m^1 = 1 - \sum_{j=1}^{1} J(s^j \le 1) = 1 - \left[J(9 \le 1)\right] = 1 - \left[0\right] = 1$$

$$d_m^2 = 2 - \sum_{j=1}^{2} J(s^j \le 2) = 2 - \left[J(9 \le 2) + J(6 \le 2)\right] = 2 - \left[0 + 0\right] = 2$$

$$d_m^3 = 3 - \sum_{j=1}^{3} J(s^j \le 3) = 3 - \left[J(9 \le 3) + J(6 \le 3) + J(7 \le 3)\right] = 3 - \left[0 + 0 + 0\right] = 3$$

$$d_m^4 = 4 - \sum_{j=1}^{4} J(s^j \le 4) = 4 - \left[J(9 \le 4) + J(6 \le 4) + J(7 \le 4) + J(4 \le 4)\right] = 4 - \left[0 + 0 + 0 + 1\right] = 3$$

$$d_m^5 = 5 - \sum_{j=1}^{5} J(s^j \le 5) = 5 - \left[J(9 \le 5) + J(6 \le 5) + J(7 \le 5) + J(4 \le 5) + J(5 \le 5)\right] = 5 - \left[0 + 1 + 0 + 1 + 1\right] = 3$$

$$d_m^6 = 6 - \sum_{j=1}^{6} J(s^j \le 6) = 6 - \left[J(9 \le 6) + J(6 \le 6) + J(7 \le 6) + J(4 \le 6) + J(5 \le 6) + J(1 \le 6)\right] =$$
$$6 - \left[0 + 1 + 0 + 1 + 1 + 1\right] = 2$$

$$d_m^7 = 7 - \sum_{j=1}^{7} J(s^j \le 7) = 7 - \left[J(9 \le 7) + J(6 \le 7) + J(7 \le 7) + J(4 \le 7) + J(5 \le 7) + J(1 \le 7) + J(8 \le 7)\right] =$$
$$7 - \left[0 + 1 + 1 + 1 + 1 + 1 + 0\right] = 2$$

$$d_m^8 = 8 - \sum_{j=1}^{8} J(s^j \le 8) = 8 - [J(9 \le 8) + J(6 \le 8) + J(7 \le 8) + J(4 \le 8) + J(5 \le 8) + J(1 \le 8) +$$
$$J(8 \le 8) + J(2 \le 8)] = 8 - \left[0 + 1 + 1 + 1 + 1 + 1 + 1 + 1\right] = 1$$

$$d_m^9 = 9 - \sum_{j=1}^{9} J(s^j \le 9) = 9 - [J(9 \le 9) + J(6 \le 9) + J(7 \le 9) + J(4 \le 9) + J(5 \le 9) + J(1 \le 9) +$$
$$J(8 \le 9) + J(2 \le 9) + J(3 \le 9)] = 9 - \left[1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1\right] = 0$$

The distance vector is therefore d_m = [1,2,3,4,4,2,2,1,0]. Finally, the correlation measure can be determined using Equation 10. $\aleph(I_1, I_2) = 1 - \frac{2\max_{i=1}^{n} d_m^i}{\left\lfloor \frac{n}{2} \right\rfloor}$ where $\aleph(I_1, I_2)$ is the measure of correlation between the corner points being analyzed between the two images, $\max_{i=1}^{n}$ is the maximum value in the distance vector, and n is the number of elements in the window which in this case is 3x3=9.

$$\aleph(I_1, I_2) = 1 - \frac{2\max_{i=1}^{n} d_m^i}{\left\lfloor \frac{n}{2} \right\rfloor} = 1 - \frac{2 \times 3}{\left\lfloor \frac{9}{2} \right\rfloor} = 1 - \frac{6}{\lfloor 4.5 \rfloor} = -0.5$$

The ordinal measure returns a result of 1 for perfect correlation if the order of the intensity values of a corner on one image exactly matches the order of the intensity values of a corner on the second image. A -1 correlation measure indicates perfect inverse correlation. In this example, the correlation measure returned a –0.5 that indicates that these two corner points do not have a good correspondence.

To ensure successful correspondences between the two frames of video, a 5x5 window was analyzed around each detected corner point in both frames and a threshold of 0.6 was used to accept a correspondence. If a correlation measure of 0.6 or higher existed then it is likely that a correspondence exists. As mentioned earlier, ordinal measures are invariant to intensity because rather than comparing the intensity values around each corner point we are measuring the order of the intensity values around each corner point. This can be helpful when processing the video frames for which sunlight or other light sources can quickly change the intensity values in a scene.

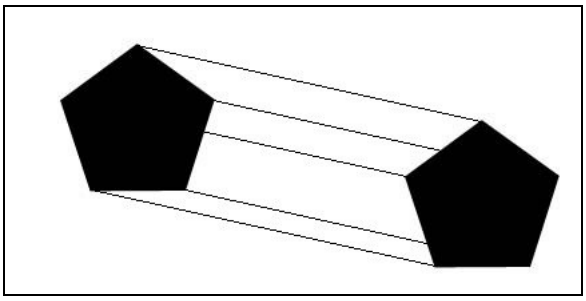

Figure 27: Correspondence by Ordinal Measure

Once the key descriptors have been defined for each corner point, the key descriptors from the first frame need to be compared against all points of the second frame while ordinal measures are calculated for each pair of corner points to find a correspondence. This is represented in Figure 28 below:

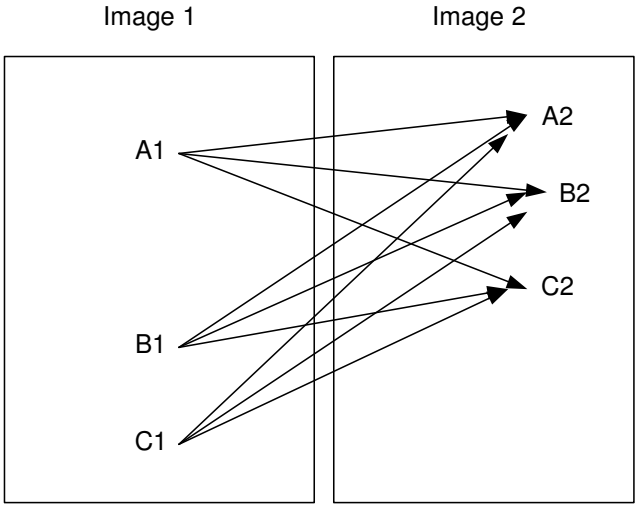

Figure 28: Correspondence Scanning Process

Each point is compared in terms of gradient magnitude, orientation, and ordinal measures. Figure 29 below shows the correspondences made between video frames 221 and 222, 245 and 246, and 511 and 512 respectively in the "HarryTheDog.avi" video clip. There are two important things happening with these correspondence results. First, multiple correspondences are being detected for individual corner point from the first frame of each pair. Secondly, sometimes the best detected correspondences are not correct. To ensure accurate correspondences, it was determined that results can be improved if the region to scan for correspondences between video frames is restricted to a

20x20 window in the same location of the detected corner points in the first frame. For example, if a corner point in frame 200 exists at coordinates (35,35), then only check for correspondences in frame 201 that exists within 20 pixels of coordinate (35,35). This makes sense since video frames are not expected to drastically change when being analyzed at 25 frames per second. This restriction proves to be accurate and helps improve system performance because unnecessary calculations for correspondences are prevented. Also, the software ensures that multiple correspondences cannot exist. Only the best corresponding pairs are selected and stored if they compare well in terms of gradient magnitude, orientation, and ordinal measures. If no reasonable correspondence can be determined for each corner point in the first frame of each pair, that corner point is ignored. Figure 30 shows the same frame correspondence results when applying the 20x20 restriction window. Notice the improved correspondence results.

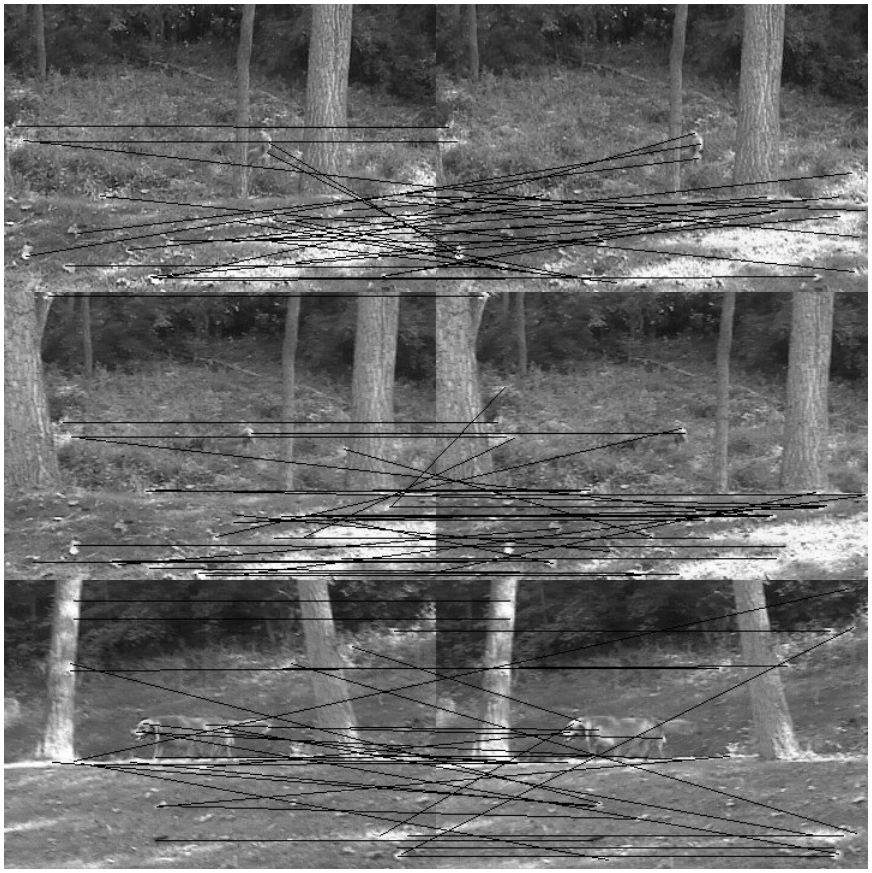

Figure 29: Correspondences between Frames 221 and 222, 245, and 246, and 511 and 512

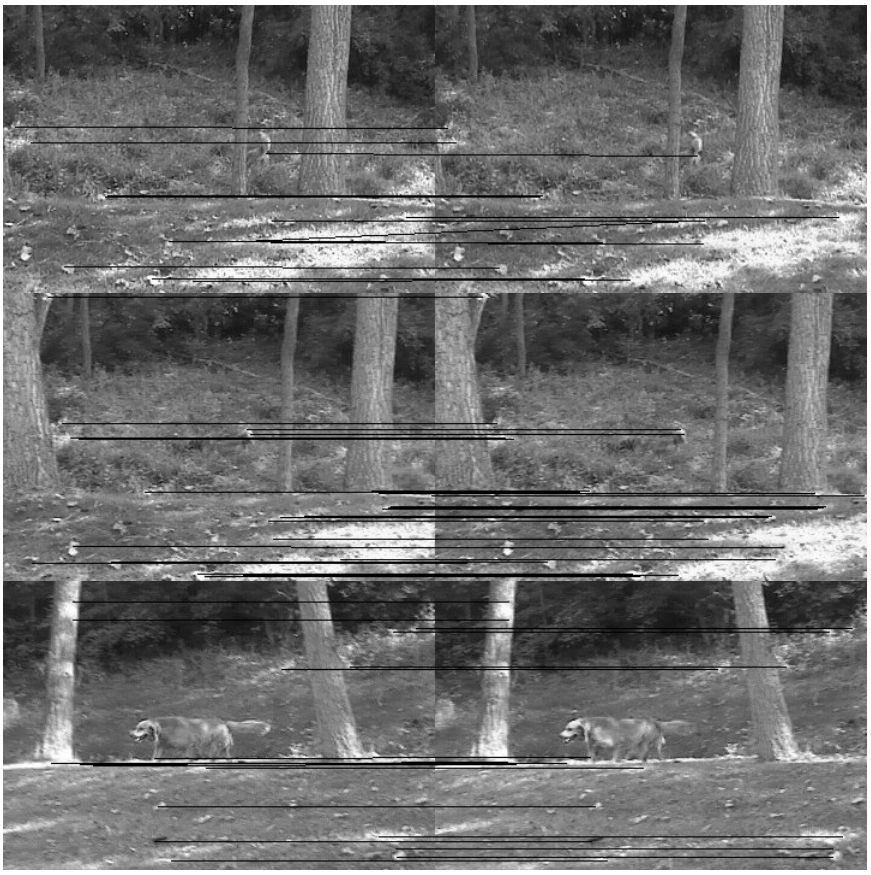

Figure 30: Correspondences between Frames 221 and 222, 245, and 246, and 511 and 512 with 20x20 Restriction Windows

4.3.7 Spatial Transformation and Registration

A spatial transformation of an image is a geometric transformation of the image coordinate system. Spatial transformations are necessary for aligning images that were taken at different times or with different sensors, correcting images for lens distortions, correcting the effects of camera orientation, and image morphing (Rhody, 2005). A special type of spatial transformation is the affine transformation.

4.3.7.1 Affine Transformation

An affine transformation is any transformation that preserves co-linearity (Rhody, 2005). It includes adjustments of rotation, translation, magnification, and shears. The following equations show the affine transformation from the (x,y) coordinate system to the (u,v) coordinate system. The coefficients c_{11} and c_{21} affect scale, c_{12} and c_{22} affect rotation and shearing, and c_{13} and c_{23} affect translation.

Equation 13: $u = c_{11}x + c_{12}y + c_{13}$

Equation 14: $v = c_{21}x + c_{22}y + c_{23}$

To register two images a transformation matrix is needed when the two images vary by translation, scale, or rotation. The transformation matrices are given below for each distortion.

$$T = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \qquad S = \begin{bmatrix} s1 & 0 & 0 \\ 0 & s2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Equation 15: Translation, Scale, and Rotation Affine Transform Matrices

To solve for the transformation matrix the values for the translational offsets, the scales, and the rotational and shearing differences between the two images would have to be known. Then the transformation matrix can be solved as the product of the previous matrices. As an example, to solve for the transformation matrix H for translation, scaling, and rotation in that order is:

Equation 16:
$$H = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s1 & 0 & 0 \\ 0 & s2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

4.3.7.2 Translation

At this point in the registration process matching coordinates have been determined. The matching coordinates need to be verified to ensure that errors do not exist. For this research, only translation is applied for image registration. Recall in the Edges, Corners, and Harris Corner Detector section it was stated that a typical video clip can run anywhere between 20 to 30 frames per second meaning the frames being analyzed vary in time by 33ms to 50ms on average. Keeping that in mind and knowing the video clips to be analyzed in this research revolve around deformable objects such as animals walking in the woods, the analysis between frames in terms of scale and rotation can be sacrificed to improve performance.

Suppose there are five matching points between two images as shown in Table 3 below. For most matching coordinates in this set, there appears to be a consistent horizontal and vertical offset of 3 pixels in the X direction and 3 pixels in the Y direction. Notice the fourth set marked in red is offset by –3 and –7 for X and Y respectively. When compared against other offsets, this correspondence is inaccurate. For this reason, the image correspondence implementation in this project determines the most common offset and considers other offsets as errors. If for example a set of 100 correspondences were determined between a pair of frames with 73 agreeing offsets in both the X and Y directions, that offset is the most likely offset to register the two frames.

Image 1		Image 2	
X	Y	X	Y
1	1	4	4
10	5	13	8
13	15	16	18
20	20	17	13
24	19	27	22

Table 3: Sample Corresponding Points between Two Images

At this point, the horizontal and vertical offset between the two frames have been determined. Therefore horizontal and vertical translation can be applied to align the images. Please refer to Figure 31 and Figure 32 below for reference. If the detected matching point on the first image has a lower column offset value, this means the image

will be placed on the right of the output image. The same idea applies for rows. If the value of the row offset on the first image is smaller than the other image row offset value, this image will be displaced downwards in the output image. For each case, the output image size is estimated depending on the coordinates obtained for the matching point. The aligned images are placed on a larger image that can hold all image information. The non-overlapping regions with no image information filled with zeros and will appear black.

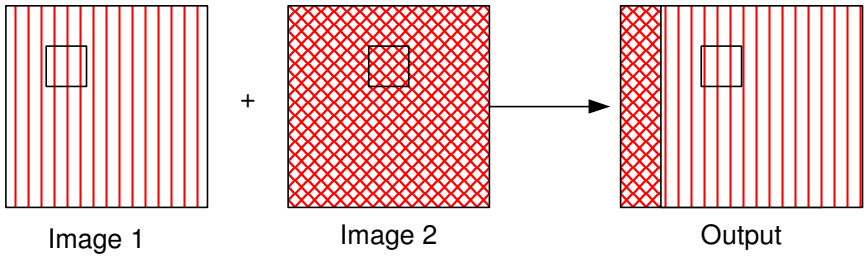

Figure 31: Horizontal Translation for Two Images with Equal Row Values

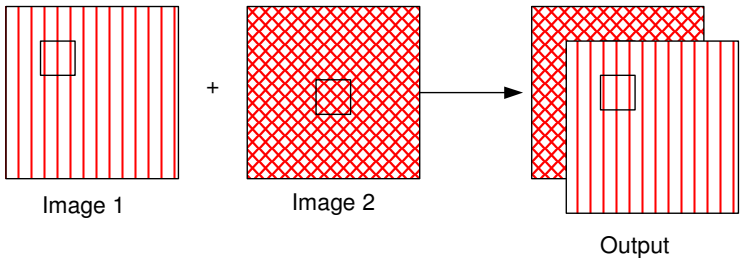

Figure 32: Horizontal and Vertical Translation for Two Images

When the images have been aligned, the combination of the two images is considered image registration. To verify the results, image averaging and image subtraction is executed. Image averaging is a simple process based on calculating the sum of the aligned pixels of each coordinate from each image and then dividing each coordinate result by two. If the images are properly aligned, the pixel values between the two images should be the same or very close. If a sample pixel element from the first image is 10 and the pixel element of the second aligned image is 10, the averaging result is $\frac{10+10}{2}=10$. Please refer to Figure 33 below for the averaged image of the input

images from Figure 23. Notice the black padding on the top, bottom, left, and right. This represents the offset padding that was needed to properly align and register the images.

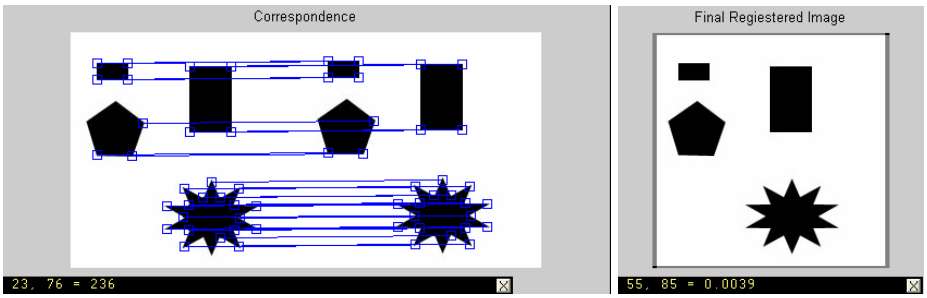

Figure 33: (L) Correspondence for Two Images and the (R) Image Registration Result

Similarly, the absolute difference of the two images if properly aligned should always be zero or close to zero in the overlapping region. The non-overlapping regions will not be affected but border elimination should be applied. Border elimination is described in the next section. If a sample pixel element from the first image is 10 and the pixel element of the second aligned image is 10, the subtraction result is 10 – 10 = 0 (black). Figure 34 shows the subtraction image of the inputs from Figure 23. Notice how the images almost appear perfectly black. Also notice the white padding on the subtraction image indicating padded areas that could not be subtracted because there is no overlapping at those regions between the two images when aligned.

The difference image will indicate motion detection. If an object is moving in a still video scene the subtraction will show regions of motion because the subtraction will not return zero for areas of moving objects. If the video is moving, two consecutive frames will be aligned and the differencing should show black for the background area and white for moving foreground areas. The resulting difference images act as masks that can be thresholded and can then be anded with the original image to show the areas of motion in the original image. Thresholding of the masks is discussed after the border elimination section. Figure 34 demonstrates detected motion in non-stationary video scenes from the "HarryTheDog.avi" video file. Table 4 shows sample data for the calculated horizontal and vertical offsets for those example video frames.

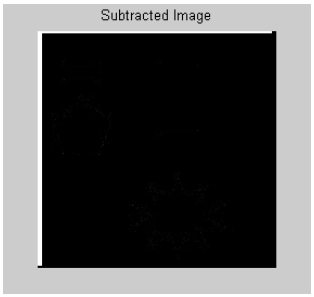

Figure 34: Subtraction Image from Two Input Images with No Motion

Frame	X-Offset	Y-Offset
221	-1	0
245	-2	0
511	-3	0

Table 4: X and Y Offsets for Frames 221, 245, and 511 in "HarryTheDog.avi"

Figure 35: Frames 221, 245, and 511 image results for (L) Registration, (M) Difference Image, and (R) the Difference Mask Anded With the Original Image

4.3.7.3 Border Elimination

After applying translation to align the images, motion is found in areas where the absolute difference returns a reasonable response. However, borders around the newly aligned difference images will not be affected as shown in the figure below of frames 221, 245, and 511 of the "HarryTheDog.avi" video clip.

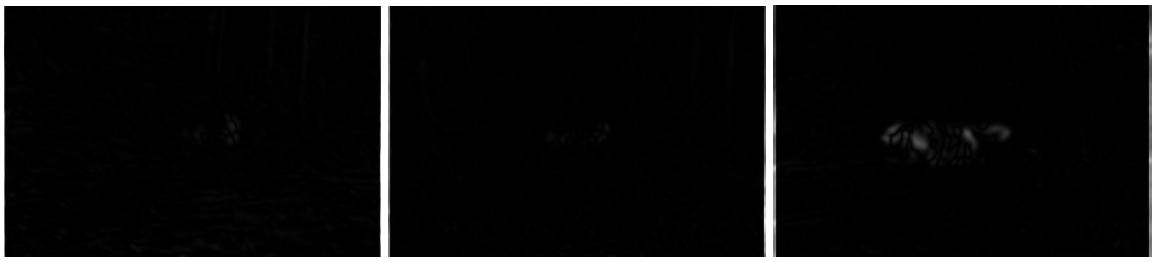

Figure 36: Absolute Difference Images for Frames 221, 245, and 511 without Border Elimination

To eliminate the border regions where there is no overlapping, the translational registration offset in the horizontal and vertical direction are used to determine how much clipping will be made around the entire absolute difference image. This is done by calculating the registration offset when aligning the two frames and removing a border width equal to that offset from each side of the difference image. The results are shown below but it is admittedly difficult to see the affects here.

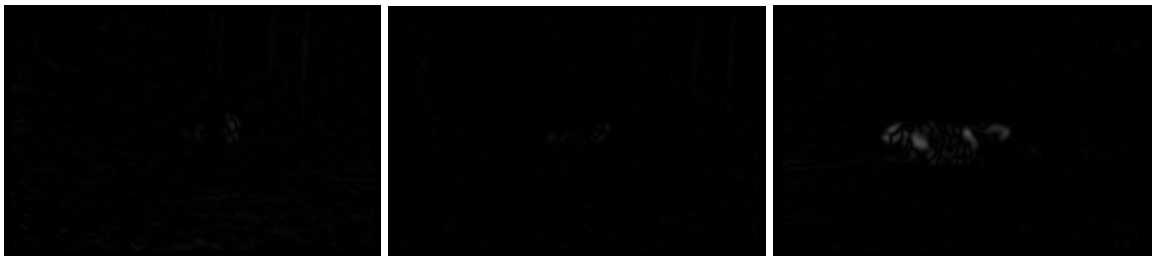

Figure 37: Absolute Difference Imaged for Frames 221, 245, and 511 with Border Elimination

4.3.7.4 Thresholding and Dynamic Noise Reduction

Thresholding plays a big role in image segmentation and noise reduction. Recall in the previous section that the motion masks generated as those shown in Figure 35 needed to be thresholded. Observe frame 511 of the "HarryTheDog.avi" video file in the following figure without thresholding. The following image was enlarged for convenient viewing. The brightest areas are the areas of motion. However, if you look very carefully you will see that background noise also exists in the difference mask.

Figure 38: Difference Mask without Thresholding

This background noise can cause problems in the segmentation stage so it is necessary to eliminate the background noise and to emphasize the foreground motion in the image mask. To do so, thresholding can be used to zero out pixels that are not within a certain range that would be classified as motion. The problem is a constant threshold cannot be applied to every frame in a video sequence. The reason is background noise is dependant on the translational differences between frames. If two consecutive frames vary by a big translational difference more background noise is apparent versus a slight or no translational difference that results in minimal background noise. For this reason, a dynamic noise reduction method needs to be applied. After testing by trial and error, Equation 17 was determined for the best results of segmenting out motion while eliminating background noise. A threshold of 9 is a reasonable threshold value to use

when the video frames are stationary. The threshold should increase with the increase in the offset by small increments.

Equation 17: $Threshold = \sqrt{xOffset^2 + yOffset^2} + 9$

Figure 39: Frame 511 Motion Mask after Thresholding

4.3.8 Segmentation

At this point we have obtained the motion masks for each video frame but there exists a problem with the masks. The masks fail to capture the entire moving object because the foreground object is not always entirely moving between video frames. Recall that motion is detected by subtracting consecutive aligned video frames and any pixels that do not subtract to zero are considered pixels that identify motion. When the dog is walking between two video frames not much has really changed, especially since the dog is mostly a single solid color. The areas of motion that will be most obvious are the areas where the dog is walking from and the areas where the dog is walking to because there is an obvious change between foreground and background. The areas of least motion are the regions in the middle of the dog because those regions can still subtract to zero since the dog is a solid color. Segmentation does not work properly

because the dog is translating over itself and therefore appears motionless. This is especially obvious in frame 511 of Figure 39.

To isolate the foreground from the background, segmentation needs to be applied. Segmentation is a process to partition an image into regions (Gonzalez & Woods, 2002, p.331). In this case, it is desired to partition the deformable object in the foreground from the background. Different directions were investigated to apply successful segmentation in the test videos that were used. They are dilation and erosion, connected component analysis and region growing, and segmentation by histograms.

4.3.8.1 Dilation and Erosion

Initially, dilation and erosion were used to segment out deformable objects in non-stationary scenes. Dilation and erosion are basic morphological processing operations that use a structuring element to enhance an image. Dilation allows objects to expand by filling in small holes and connecting disjoint objects while erosion shrinks objects by reducing object boundaries (Umbaugh, 2005). The following steps are taken as described by Umbaugh to perform dilation and erosion on binary images.

4.3.8.1.1 Dilation Procedure

1. If the origin of the structuring element coincides with a '0' in the image, there is no change; move to the next pixel.
2. If the origin of the structuring element coincides with a 1 in the image, perform the OR logic operation on all pixels within the structuring element.

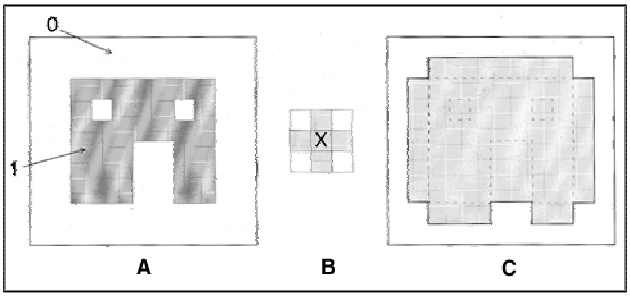

Figure 40: Dilation: (A) Original Image; (B) Structuring Element; (C) Image after Dilation (Umbaugh, 2005)

4.3.8.1.2 Erosion Procedure

1. If the origin of the structuring element coincides with a '0' in the image, there is no change; move to the next pixel.
2. If the origin of the structuring element coincides with a '1' in the image, and any of the '1' pixels in the structuring element extend beyond the object ('1' pixels) in the image, then change the '1' pixel in the image to a '0'.

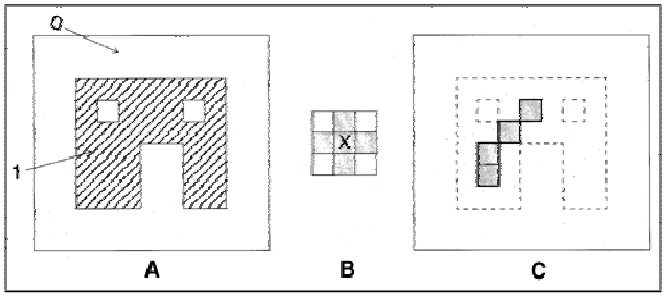

Figure 41: Erosion: (A) Original Image; (B) Structuring Element; (C) Image after Erosion (Umbaugh, 2005)

Unfortunately, as demonstrated in the results below, the use of the combination of erosion and dilation fail to properly segment out the moving objects because these techniques are primitive and uncontrolled. Notice how frame 221 has had background noise expanded while frame 511 came out fairly good. In some cases too much background is considered foreground and too much foreground is considered background. Because segmentation is not guaranteed to be of acceptable quality or to segment out just the foreground in most frames, other techniques had to be investigated including region growing and histogram segmentation.

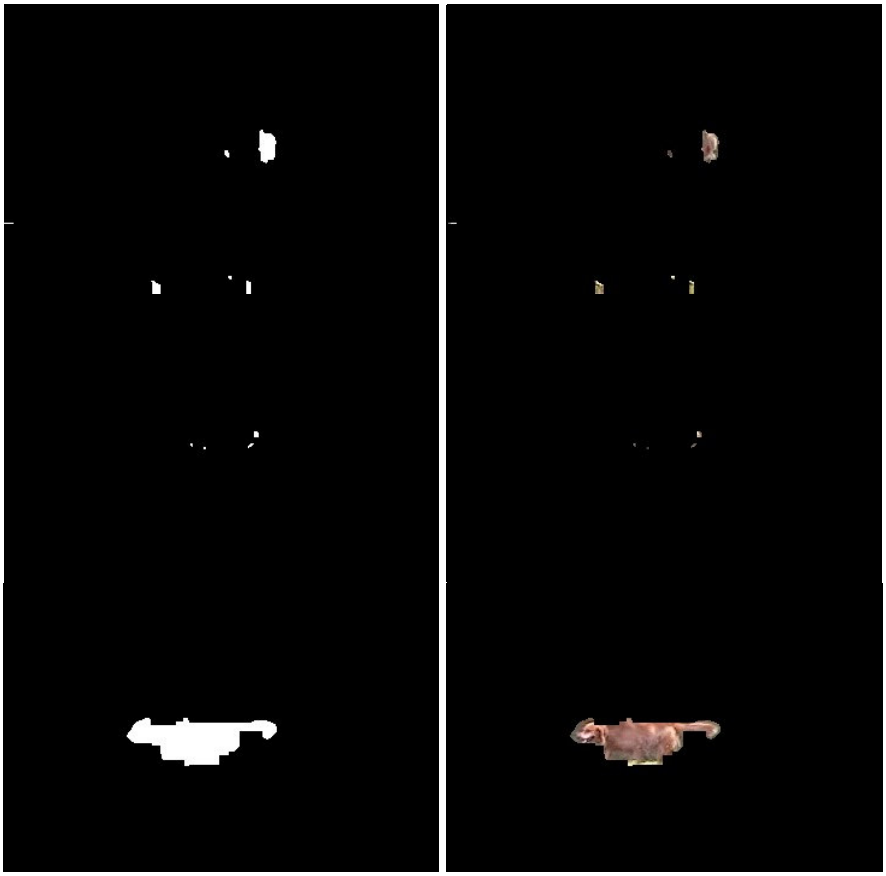

Figure 42: Frames 221, 245, and 511 after Applying Basic Dilation and Erosion

4.3.8.2 Region Growing

Region growing is a procedure that groups pixels into larger regions based on predefined criteria (Gonzalez & Woods, 2002, p. 613). A starting seed is selected in an image and pixels are grown by connecting neighboring pixels that have similar properties to the seed. In Figure 43, a seed was selected within the larger blob. Then pixels were grouped and marked as green from the starting seed until no more neighboring pixels can be grouped together.

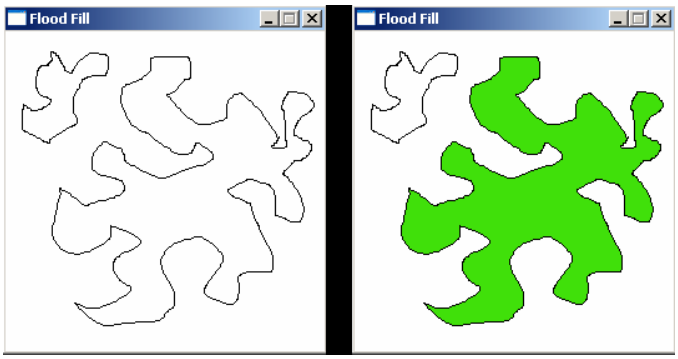

Figure 43: Region Growing Example
(http://www.student.kuleuven.ac.be/~m0216922/CG/floodfill.html)

Selecting a starting seed is not difficult because our current motion mask show us portions of foreground deformable object. The steps taken for region growing in this research are explained in the next section.

4.3.8.2.1 Steps for Region Growing

1. Create a map of the current difference image. Each pixel is assigned a value in the mask. A "0" indicates that the current pixel is not within the difference image response region. A "1" indicates a pixel is within the difference image response region but has not yet been analyzed. A "2" indicates a pixel is within the difference image response region that has been analyzed. To do this, make a copy of the difference image into a region map object. Normalize the region map so that any motion detected regions will be defined as 1 rather than 255, the original values after thresholding. Now we have a region map object with "0" for areas of undetected motion and "1" for areas of detected motion. As each pixel in the mask image is analyzed, they can be assigned any of the following values:
    a. 0 = Not in Mask and not analyzed
    b. 1 = In original mask but not analyzed
    c. 2 = Region grown but not analyzed
    d. 3 = In original mask and analyzed
    e. 4 = Region grown and analyzed
    f. 5 = Not in mask and analyzed

2. Loop through each pixel coordinate I(x,y) in the current unmodified video frame. For each pixel, lookup the corresponding region map. If the corresponding value is equal to 1 or 2, analyze the neighborhood of that pixel on the current video frame. If any of these pixels have a color value similar to the current pixel being analyzed, update the region map with the appropriate value.

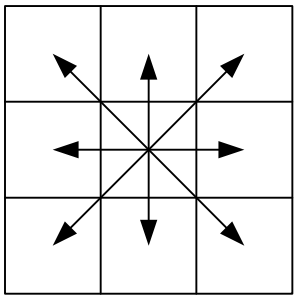

Figure 44: Region Growing Analysis of Neighboring Pixels

3. Repeat step 2 until all pixels in the region map are analyzed.
4. The result at this point is an updated mask with region growing that can be applied on the current frame to segment out the deformable objects.

The following results show that the region growing implementation with a manual seed selection works as expected on basic geometric objects with no noise. However, the results do not work as expected when applying this technique on video frames from the "HarryTheDog.avi" video clip.

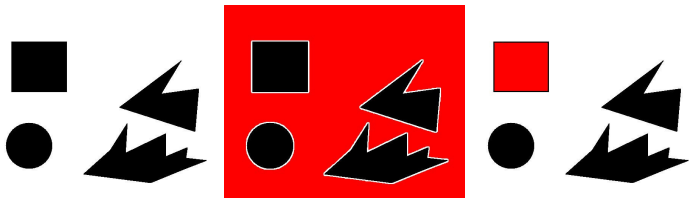

Figure 45: Region Growing; (L) The Original Image, (M) Region Growing at Seed (0,0), (R) Region Growing at Seed (100,100)

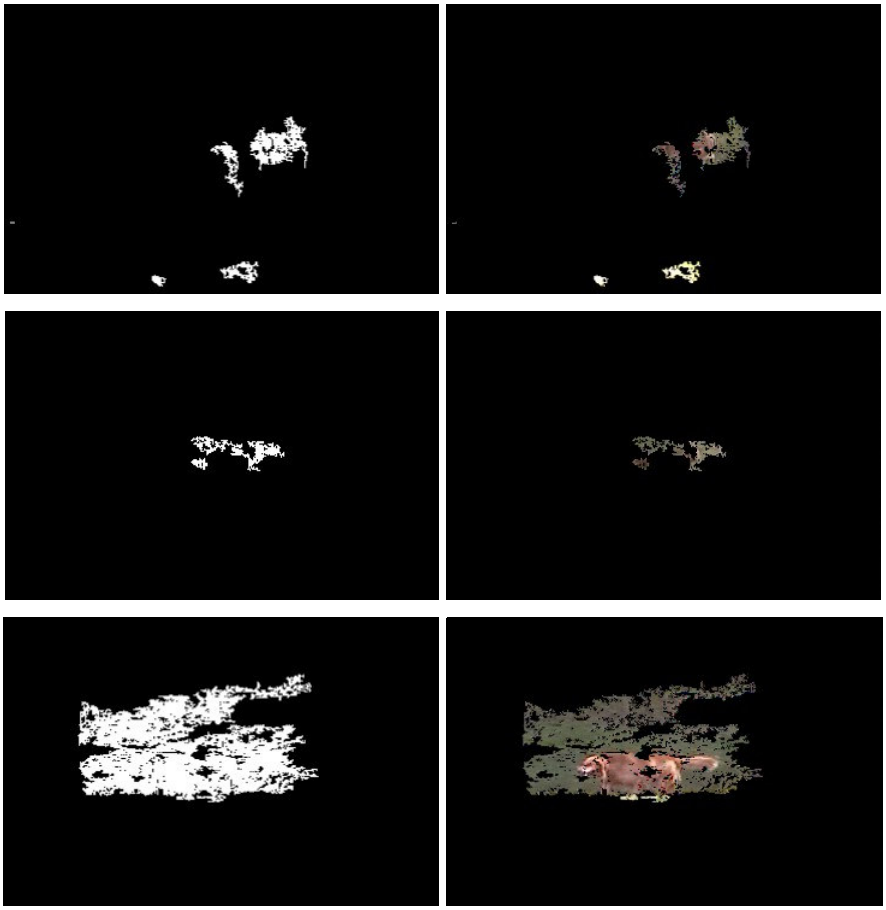

Figure 46: Frames 211, 245, and 511 Segmentation with Region Growing

The reason poor results were obtained was because the motion masks include regions with both foreground and background pixels. Recall that motion includes responses where the deformable object moves into and regions where the deformable object leaves. This means that seeds can be selected that are actually part of the background and can therefore be grown and grouped together to provide masks as shown for frame 511 in Figure 46. It was obvious that another solution was needed.

4.3.8.3 Histogram Segmentation

A histogram is a graphical means of displaying the number of items in a collection. For images, histograms can display the distribution of pixels, which is useful for enhancing images and determining optimal thresholds. The goal is to segment out the deformable object and identify that object as a foreground while ignoring the background. To do so, an idea of segmenting by histograms was investigated for which a

histogram is analyzed on the current motion mask of each video frame. The motion mask anded with the original frame, although identifying some areas of both foreground and background, displays a majority of the deformable object. By applying a histogram on the mask and ignoring all black pixels (background pixels), the histogram should show peaks of the most common pixel values in the mask that identify the deformable moving object. However, rather than analyzing histograms in the RGB color space, it was found that analyzing the frames in the HSV color space was more useful.

4.3.8.3.1 HSV Color Space

HSV stands for Hue, Saturation, and Value and a model of the color space is shown in Figure 47 below. We classify colors as a simple color type such as yellow, red, green, and blue. These basic color types are identified by the hue. The saturation indicates the purity of a color. A low saturation value more grayness and fade while a high saturation color indicates less impurities. The value indicates the brightness of the color.

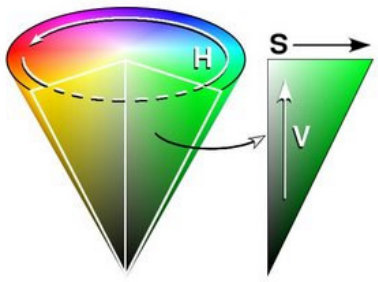

Figure 47: HSV Color Space Model (Wikipedia, 2005)

To convert from the RGB color space to the HSV color space, the following algorithm is used as explained in the OpenCV reference manual.

$$V = \max(r, g, b),$$

$$S = \begin{cases} \dfrac{(V - \min(r, g, b)) \times 255}{V}; V != 0 \\ 0 \end{cases},$$

$$H = \begin{cases} \dfrac{(G - B) \times 60}{S} \xrightarrow{for} V = R \\ 180 + \dfrac{(B - R) \times 60}{S} \xrightarrow{for} V = G \\ 240 + \dfrac{(R - G) \times 60}{S} \xrightarrow{for} V = B \end{cases}$$

HSV images and their corresponding masks on the original HSV image are shown of frames 221, 245, and 511 in the "HarryTheDog.avi" video file. Now histograms can be analyzed for each HSV mask to determine the peak hue and saturation for segmentation.

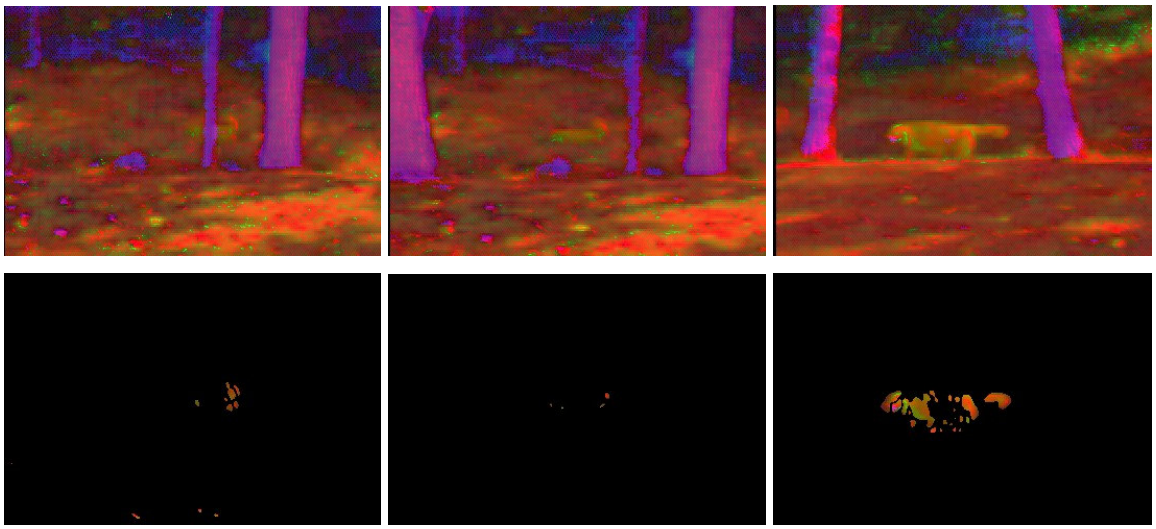

Figure 48: HSV Images and their Corresponding Masks for Frames 221, 245, and 511

4.3.8.3.2 Segmenting in the HSV Color Space

The HSV color space can be divided into three planes to represent the hue, saturation, and value individually. Hue and saturation contain enough information that is needed to segment out the deformable object in the video scene. Figure 49 and Figure 50 show the masked hue and saturation planes and their corresponding histograms for

frames 221, 245, and 511. For demonstration purposes these histograms have been normalized. Notice the peaks in the hue and the saturation histograms. They all appear consistent and now the original frame can be segmented in the HSV color space to obtain an improved mask that will segment out the foreground deformable object. The mask is obtained by checking the peak in the hue plane and saturation plane and comparing each pixel in the original non-masked hue and saturation frame to the predetermined peaks. If the hue and saturation pixel values agree within a certain absolute difference, then that pixel is activated as a foreground object identify the deformable object in the video scene. Otherwise, the pixel is deactivated and assumed to be background.

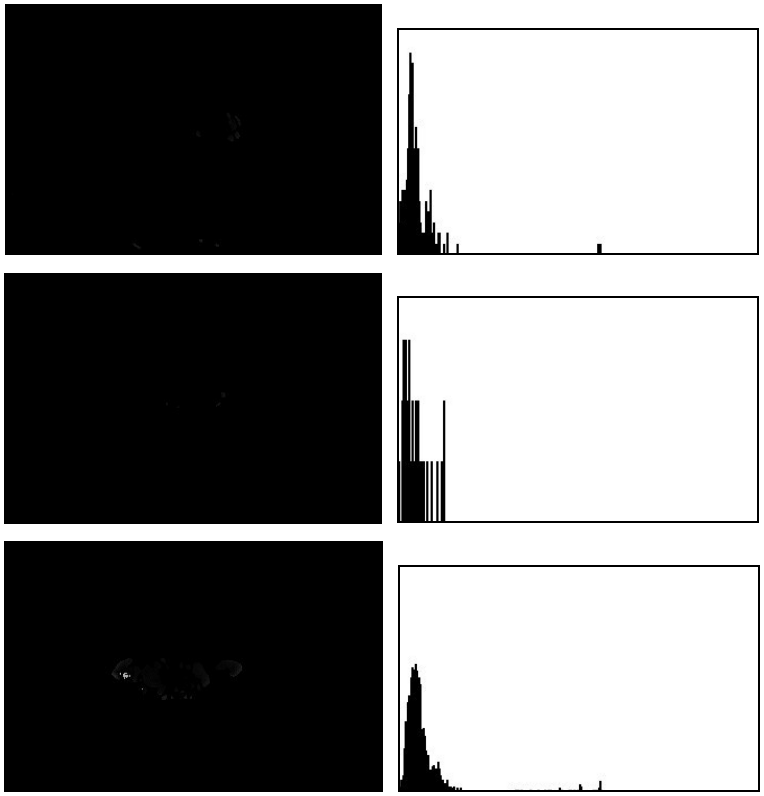

Figure 49: Hue Masks and their Corresponding Histograms for Frames 221, 245, and 511

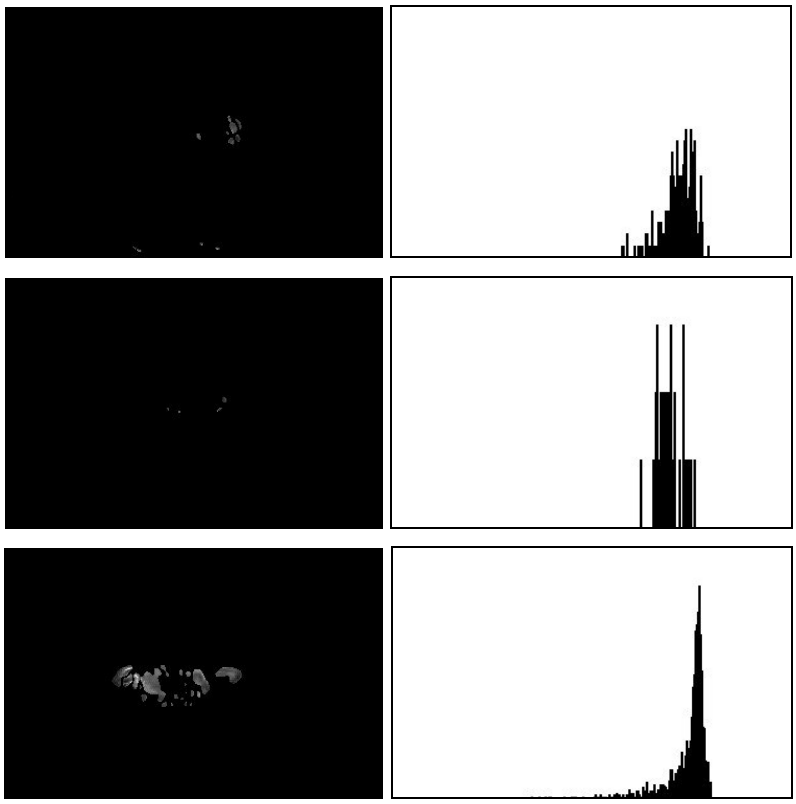

Figure 50: Saturation Masks and their Corresponding Histograms for Frames 221, 245, and 511

The hue and saturation masks can then be anded together to obtain a mask that will identify the deformable object with similar hue and saturation peaks to that identified by the mask. The results of segmentation are shown below. Notice that there still exists background noise but the foreground object is segmented fairly well.

Figure 51: Segmentation Results with HSV Histograms on Frames 221, 245, and 511

5 Future Work and Improvements

5.1 PERFORMANCE IMPROVEMENTS

In all, the results came out fairly well indicating that the approach taken for segmenting deformable objects in non-stationary scenes has potential. One area of concern is real-time performance. Throughout this paper there has been a defense on not executing certain steps in order to improve performance. A sacrifice was made between quality and performance. At this point on a system running on a Windows Platform with a 750Mhz processor the software implemented for this research can process between 7 to 10 FPS. Of course this means this software cannot process real-time video on this machine. Improvements can be made. First, the Harris Corner Detector although effective requires a lot of resources to process each frame and detect corners. Other methods of feature point extraction need to be investigated. The correspondence stage is also time consuming. Processing between 100 to 150 corners per frame and finding a correspondence between two separate frames requires determining the most common horizontal and vertical offsets. Faster methods need to be investigated for determining the common offsets for translation.

5.2 SCALE INVARIANCE

After significant testing it was determined that scaling video frames affects the correspondence stage. The ordinal measure is invariant to intensity around each detected corner but that's the only true invariance involved. As a result, significant scaling in video scenes causes two consecutive frames to not register properly and as a result deformable objects are not segmented correctly.

5.3 SEGMENTATION

There are advantages and disadvantages of segmenting in the HSV color space. The dog in this video example is segmented well, even when walking behind the tree in frame 221. However, there are limitations to this process. Because peaks in the hue and saturation histograms identify the thresholding values to be used for segmentation, multiple deformable objects of varying hue and saturation cannot be detected simultaneously. Instead, the deformable object with the strongest motion response mask

will be segmented. Another limitation to this design is that the deformable objects must always be moving to be detected. In the "HarryTheDog.avi" video clip the dog is constantly moving and wagging its tail. If there is no motion the hue and saturation histograms will not contain data and therefore segmentation will not be performed. Finally, there is too much background noise resulting from the current segmentation method. Basic morphological techniques can be used to improve the qualities of the mask but none have been implemented at this time. It has been difficult to address these various problems. In an ideal world the motion masks would perfectly segment out the deformable objects and segmenting in the HSV color space via histograms would not have been necessary. Other means of segmentation should be investigated that will allow for multiple deformable object recognition

6 REFERENCES

1) Bhat, Dinark; Nayar, Shree. "Ordinal Measures for Image Correspondence". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 4, April 1998. Columbia University.
2) Brown, Lisa. "A Survey of Image Registration Techniques". ACM Computing Surveys. Vol. 24, No. 4, December 1992. Department of Computer Science, Columbia University.
3) "Convolution Based Operations". Quantitive Imaging Group. 1999. http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Convolut-2.html. October 27, 2005.
4) "Derivative Based Operations". Quantitative Imaging Group. 1999. http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip-Derivati.html. May 10, 2005.
5) Fisher, Bob; Perkins, Simon; Walker, Ashley; and Wolfart, Erik. "Edge Detectors". Hyper Media Image Processing Reference. 1994. http://www.cee.hw.ac.uk/hipr/html/edgdetct.html. May 10, 2005.
6) "HSV Color Space". Wikipedia. October 18, 2005. Retrieved October 30, 2005 from http://en.wikipedia.org/wiki/HSV_color_space
7) Rafael C. Gonzales, Richard E. Woods, Digital Image Processing, second edition, Prentice Hall, 2002.
8) Haeberli, Paul and Ofek, Eyal. "Automatic Panoramic Image Merging". http://www.sgi.com/misc/grafica/merge/. May 10, 2005.
9) Hebert, Martial. "Interest Points and Scale Invariance". Carnegie Mellon. May 10, 2005. <www.andrew.cmu.edu/course/16-720/lectures/**interest**.pdf>
10) Kudo, Yuma; Kajiyama, Takeshi; and Mitsahushi, Wataru. "Extraction of Feature Points Suitable for Matching Image Correspondence". IEEE. 2003. University of Electro-Communications.
11) Lowe, David. "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision. Jan. 5, 2004. Computer Science Department, University of British Columbia.
12) Open Source Computer Vision Library. Retrieved June 17, from http://www.intel.com/technology/computing/opencv/index.htm/
13) Parks, Donovan. "Harris/Plessy Operator". http://www.cim.mcgill.ca/~dparks/mainHarris.htm. May 10, 2005.
14) Pollefeys, Marc. "Feature Point Extraction". July 12, 2000. http://www.esat.kuleuven.ac.be/~pollefey/tutorial/node51.html. May 10, 2005.
15) RGB World Inc (1996). "RGB World – Understanding Color". Retrieved October 26, 2005 from http://www.rgbworld.com/color.html.
16) Rhody, Harvey. "Geometric Image Transformations". Chester Carlson Center for Image Science. 2005. Rochester Institute of Technology.
17) Skowhegan Area High School, "Geometric Transformations". Retrieved September 12, 2005 from <http://www.msad54.k12.me.us/MSAD54Pages/skow/MathDept/CPMP/M2U2/m2u2page3.html>

18) Umbaugh, Scott. "Morphology Overview". Retrieved October 30, 2005 from http://zone.ni.com/devzone/conceptd.nsf/webmain/8CA8DE2E8881C1AB8625682E0079CE74
19) University of Edinburgh, "Gaussian Smoothing". Retrieved August 17, 2005 from http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
20) Zhenjiang, Miao; Baozong, Yuan. "A NN Image Understanding System for Maps and Animals Recognition". IEEE. 1993. Institute of Information Science, Northern Jiaotong University. Beijing, China.