Background stabilization and debris detection in launch pad video monitoring

by

Kaushik Gopalan
B.E. Rashtreeya Vidyalaya College of Engineering, 2003

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2005

Abstract

Automatic detection of moving objects in video sequences is a widely researched topic with application in surveillance operations. Methods based on background cancellation by frame differencing are extremely common. However this process becomes much more complicated when the background is not completely stable due to camera motion.

This thesis considers a space application where surveillance cameras around a shuttle launch site are used to detect any debris from the shuttle. The ground shake due to the impact of the launch causes the background to be shaky. We stabilize the background by translation of each frame, the optimum translation being determined by minimizing the energy difference between consecutive frames. This process is optimized by using a sub-image instead of the whole frame, the sub-image being chosen by taking an edge detection plot of the background and choosing the area with greatest density of edges as the sub-image of interest.

The stabilized sequence is then processed by taking the difference between consecutive frames and marking areas with high intensity as the areas where motion is taking place. The residual noise from the background stabilization part is filtered out by masking the areas where the background has edges, as these areas have the highest probability of false alarms due to background motion.

Dedicated to the beautiful minds, kind hearts, noble souls and free spirits of the world.

Acknowledgments

I thank my advisor, Dr. Takis Kasparis for his valuable guidance and support. I also thank the members of my thesis committee, Dr. Linwood Jones and Dr. Samuel Richie. I acknowledge the support that Achilleas Kourtellis, KarthikKalathi Vanumamalai and Vasud Torsekar have given me. I also thank the entire UCF community for being extremely kind to me and for tolerating my occasionally erratic driving.

I thank my parents and my brother for having played such a huge part in getting me this far. I thank my many aunts,uncles and cousins for helping me in more ways than I can count. I acknowledge the immense part Chaitanya R. R. K., Harsha G. K., Janakiraman V., Kiran Gonsalves and Nikhil Bhandari have played in my getting through the 150 odd tests that I have written as part of my undergraduate degree without any serious mishaps. I thank everybody in the Chapa group for tolerating my eccentricities for four years.

I thank Mathworks Inc. and the LaTeXcommunity for reducing greatly the labor involved in writing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The detection of movement in frame sequences is done by a variety of methods including frame differencing, determination of optical flow and by using the features of various objects in the sequence to track them. However, it is not possible to use many of these methods in a case where the background is semi-stable. This thesis deals with an application of this nature, where the movement in the background is caused by ground shake due to a satellite launch. The methods used in this thesis are simple, commonly used image processing techniques that have been combined together to give a reasonably good solution of the problem at hand.

The problem has been divided into a stabilization phase and a detection phase thereby providing an elegant way of reducing the complexity of the programming involved. It is likely that this approach adds some computational inefficiency to the algorithm, but the author believes that the resulting simplicity of the approach makes the trade-off worthwhile.

1.1 Organization of the thesis

Chapter 2 contains a review of the common methods in existing literature for motion detection in video sequences. It contains a short explanation of methods based on frame differencing. This is also a short discussion of motion detection based on optical flow, more specifically the Lucas-Kanade image alignment algorithm.

Chapter 3 contains a description of the methods used by the author for background stabilization. It contains an explanation of the process of feature extraction including a description of the Sobel method of edge detection which is used to extract information about the background. The chapter then describes the process of aligning successive frames to the initial background by shifting each frame such that the error energy between frames is minimized.

In Chapter 4 the author describes the method used to determine regions of movement by subtracting consecutive frames and also describes ways of masking out some of the false alarms caused by imperfections in the stabilization process.

Chapter 5 contains a brief description of the Graphical User Interface that is being created to help in the deployment of the software being developed for debris tracking.

1.2 Motivation for the thesis

The premier motivation for this project comes from the increased emphasis on safety in NASA's space program in the aftermath of the tragic crash of the space shuttle *Columbia*.

The space shuttle *Columbia*, NASA Orbiter Vehicle designation: OV-102, was the first shuttle of NASA's orbital fleet that went on 28 missions to space between 1981 and 2003. It was launched for the final time on the 16th of January, 2003 on a micro gravity and Earth science research mission, STS-107, that carried out a wide range of scientific experiments including biomedical research and earth observation. On the 1st of February, 2003 *Columbia* disintegrated during it's re-entry into the earth's atmosphere, just sixteen minutes before scheduled touchdown. All seven crew members were killed in the crash.

The video taken during take-off showed a piece of foam from an external fuel tank strike the shuttle's left wing. The damage caused to tiles in the thermal protection system by this debris strike is generally thought to have caused the accident.

These events have given a greater sense of importance to the process of detecting and analyzing debris in NASA's shuttle program. They have shown that what was previously considered a "turnaround issue" is vital to ensuring the safety of the space shuttle and those aboard it.

As a result NASA has greatly increased the amount of video footage taken during shuttle launches. This increased volume of footage has made human review of these video sequences significantly more time consuming and expensive. The goal of the Digital Signal Processing laboratory at the University of Central Florida is to create an automated program that reads various surveillance videos and automatically detects sequences where there is unexpected motion that might indicate the presence of debris. Such a program would greatly reduce the human labor involved in detecting potentially dangerous debris strikes and would provide useful support to engineers who are involved in ensuring the safety of the shuttle and its occupants.

1.3 Block diagram of the motion detection algorithm

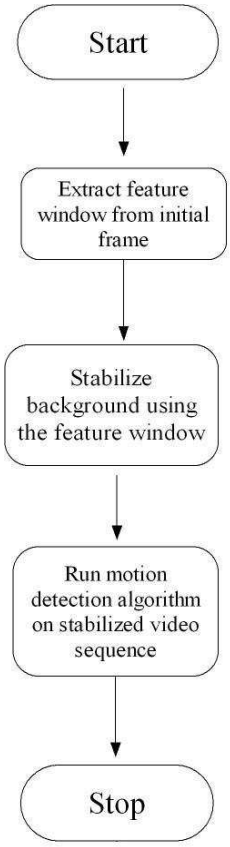

Figure 1.1: Block diagram of the motion detection algorithm

The sequence of operations performed in this thesis are shown in the preceding block diagram. The three major steps involved are feature extraction, background stabilization and motion detection. These steps are explained in detail in subsequent chapters.

CHAPTER 2

REVIEW OF EXISTING LITERATURE

We look at some of the basic ideas used in the various motion detection algorithms that are in existence. Though there are innumerable papers in this area of research, the discussion in this thesis is limited to the most commonly used methods. This chapter aims merely to provide some background about the basic concepts involved in motion detection and is by no means a comprehensive review of the current state of the art.

2.1 Methods based on frame differencing

A really common way of detecting movement in frame sequences is to subtract each frame from its previous frame. Ideally this cancels out all the pixels in which there is no change leaving non zero values only in pixels where motion has occurred. This method is central to the work done in this thesis and merits further scrutiny by means of an example.

Figure 2.1: Frame sequence shows the author playing ping-pong.

The figure 2.1 is a frame sequence taken from a hand-held camera. Let us look at the result of taking the difference of successive frames.

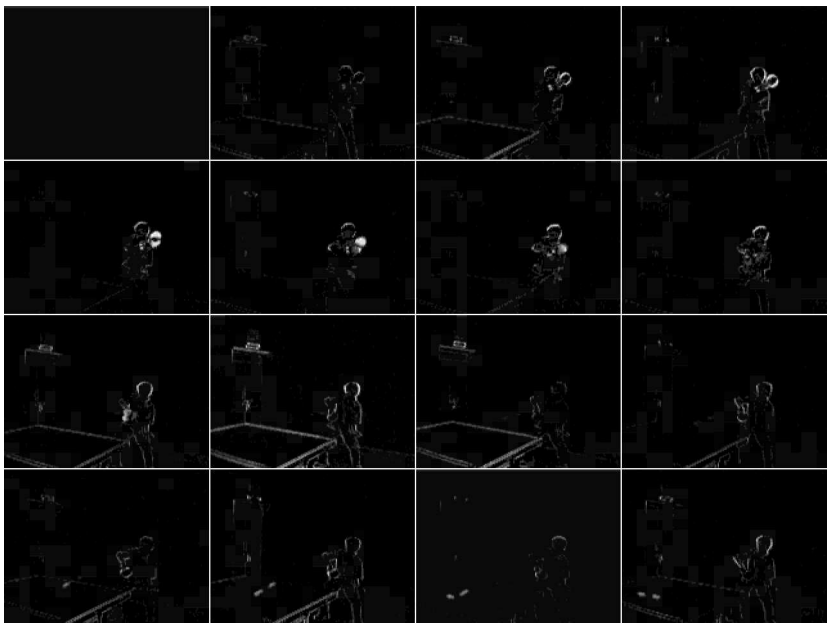

Figure 2.2: Result of subtracting successive frames

On integrating these difference frames together it is possible to determine the trajectory of motion in the sequence as is shown in figure 2.3

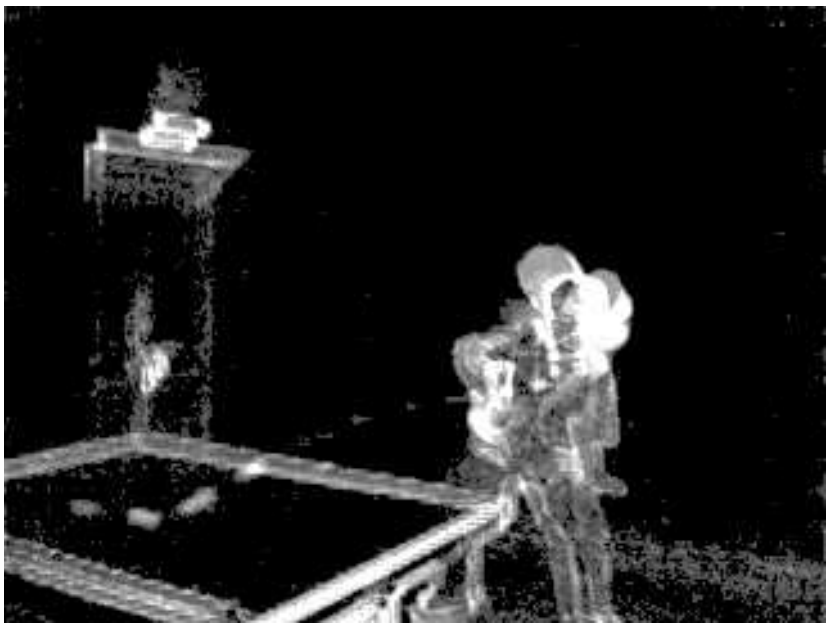

Figure 2.3: Result of integrating the difference frames

2.2 Methods based on determination of optical flow

Optical flow represents the three dimensional motion of the various objects in a video sequence as a set of vectors either originating or terminating at each pixel. There are many methods of determine the optical flow. Let us discuss some of the most common ones.

2.2.1 Lucas-Kanade method for Image registration

Lucas and Kanade have provided a widely used method for image registration that is based on minimizing the least squares deviation.

Let us consider an image $I_1(x, y)$. We wish to translate I_1 by some (h_1, h_2) such that $I(x + h_1, y + h_2)$ is as close as possible to some reference image I_2

We start by using the approximation

$$I_1(x + h_1, y + h_2) \approx I_1(x, y) + h_1 \frac{\delta I_1}{\delta x} + h_2 \frac{\delta I_1}{\delta y} \tag{2.1}$$

The error is defined as

$$\begin{aligned} E &= \sum_x \sum_y [I_1(x + h_1, y + h_2) - I_2(x, y)]^2 & (2.2) \\ &= \sum_x \sum_y [I_1(x, y) + h_1 \frac{\delta I_1}{\delta x} + h_2 \frac{\delta I_1}{\delta y} - I_2(x, y)]^2 & (2.3) \end{aligned}$$

To minimize the error we set

$$\begin{aligned} \frac{\delta E}{\delta h_1} &= 0 \\ \frac{\delta E}{\delta h_2} &= 0 \end{aligned} \tag{2.4}$$

The solution of which ends up as

$$\begin{aligned} h_{1_{opt}} &= \frac{\sum_x \sum_y \frac{\delta I_1}{\delta x} [I_2(x, y) - I_1(x, y)]}{\sum_x \sum_y [(\frac{\delta I_1}{\delta x})^2 + (\frac{\delta I_1}{\delta y})^2]} \\ h_{1_{opt}} &= \frac{\sum_x \sum_y \frac{\delta I_1}{\delta y} [I_2(x, y) - I_1(x, y)]}{\sum_x \sum_y [(\frac{\delta I_1}{\delta x})^2 + (\frac{\delta I_1}{\delta y})^2]} \end{aligned} \tag{2.5}$$

The reader is referred to [LK81] for further generalizations of this method to include rotation and zooming as well as an iterative method for solving the above equations.

2.2.2 Horn-Schunck algorithm

This algorithm provides a way of determining the optical flow in images. It is based on the assumption that the pixel intensities of the same object in different frames is constant.

Let $I(x, y, t)$ be the pixel intensity at a point (x, y) at time t. Let us now assume that this object moves to the point $(x + \mathrm{d}x, y + \mathrm{d}y)$ at time $t = \mathrm{d}t$.

Since the two intensities are equal

$$\begin{aligned} I(x,y,t) &= I(x+\mathrm{d}x, y+\mathrm{d}y, t+\mathrm{d}t) \\ I(x,y,t) &= I(x,y,t) + \frac{\delta I}{\delta x}\mathrm{d}x + \frac{\delta I}{\delta y}\mathrm{d}y + \frac{\delta I}{\delta t}\mathrm{d}t \\ I(x,y,t) &+ \frac{\delta I}{\delta x}\mathrm{d}x + \frac{\delta I}{\delta y}\mathrm{d}y + \frac{\delta I}{\delta t}\mathrm{d}t = 0 \end{aligned} \tag{2.6}$$

Dividing by $\mathrm{d}t$ and setting $\frac{\mathrm{d}x}{\mathrm{d}t} = u$ and $\frac{\mathrm{d}y}{\mathrm{d}t} = v$ we get

$$I_x u + I_y v + I_t = 0 \tag{2.7}$$

where $I_x I_y$ and I_t are $\frac{\delta I}{\delta x}\frac{\delta I}{\delta y}$ and $\frac{\delta I}{\delta t}$ respectively.

We add another constraint that the patterns in the image move at the same velocity, i.e. we minimize

$$(\frac{\delta u}{\delta x})^2 + (\frac{\delta u}{\delta y})^2 + (\frac{\delta v}{\delta x})^2 + (\frac{\delta v}{\delta y})^2 \tag{2.8}$$

In summation the two quantities we need to minimize are

$$\begin{aligned} \epsilon_1^2 &= (\frac{\delta u}{\delta x})^2 + (\frac{\delta u}{\delta y})^2 + (\frac{\delta v}{\delta x})^2 + (\frac{\delta v}{\delta y})^2 \\ \epsilon_2 &= I_x u + I_y v + I_t \end{aligned} \tag{2.9}$$

We therefore minimize the quantity

$$\epsilon^2 = \int\int (\lambda \epsilon_1^2 + \epsilon_2^2) \mathrm{d}x\mathrm{d}y \tag{2.10}$$

If we approximate $\nabla^2 u$ and $\nabla^2 v$ as

$$\begin{aligned} \nabla^2 u &= 3(\hat{u}_{x,y} - u_{x,y}) \\ \nabla^2 v &= 3(\hat{v}_{x,y} - u_{x,y}) \end{aligned} \tag{2.11}$$

where the local averages $\hat{u}$ and $\hat{v}$ are defined as

$$\begin{aligned} \hat{u}_{x,y} &= \frac{1}{6}(u_{x-1,y} + u_{x+1,y} + u_{x,y-1} + u_{x,y+1}) + \frac{1}{12}(u_{x-1,y-1} + u_{x-1,y+1} + u_{x+1,y-1} + u_{x+1,y+1}) \\ \hat{v}_{x,y} &= \frac{1}{6}(v_{x-1,y} + v_{x+1,y} + v_{x,y-1} + v_{x,y+1}) + \frac{1}{12}(v_{x-1,y-1} + v_{x-1,y+1} + v_{x+1,y-1} + v_{x+1,y+1}) \end{aligned}$$

The solution to 2.10 then becomes

$$(\lambda + I_x^2 + I_y^2)u = -I_x I_y \hat{v} + (\lambda + I_y^2)\hat{u} - I_x I_t \tag{2.12}$$

$$(\lambda + I_x^2 + I_y^2)v = -I_x I_y \hat{u} + (\lambda + I_x^2)\hat{v} - I_y I_t \tag{2.13}$$

2.3 Probabilistic methods for motion detection

These methods attempt to determine the probability of object motion occuring at any given point by considering the statistics of the image intensities. These methods might either be pixel based or region based.

2.4 Methods based on features of objects in the video sequence

It is possible to use the properties of the various objects in the video to track them through the various frames. These properties include color, shape, texture and trajectory of motion. It is possible also to segment different objects by representing them as a mixture of two dimensional gaussians. These methods are not suitable for the problem being addressed by this thesis as the debris from the shuttle is of unknown shape and size, and could travel in any possible trajectory.

CHAPTER 3

BACKGROUND STABILIZATION

3.1 Introduction

This section deals with the problem of eliminating the background shake from the video sequence in order to simplify the task of detecting moving objects. We start by describing the need for background stabilization. The process of extracting a suitable feature from the background frame is then explained. The algorithm for minimizing the movement of the scene between frames using the extracted feature is then described in detail.

3.2 Need for stabilization

Due to the fact that this thesis uses frame differencing to detect motion, it is essential that consecutive frames are aligned with each other. The degree of mis-alignment between the frames is directly proportional to the amount of noise in the difference image.

Frame 203

Frame 204

Figure 3.1: Two consecutive frames in the unstabilized video

The figure 3.1 shows consecutive frames from the unstabilized video. The two frames are highly mis-aligned, which is easily observed from the position of the poles on the right side of the frames. The author wishes to illustrate the fact that frame differencing in such a scenario produces results that are not useful in the process of detection of debris. This is demonstrated by Figure 3.2.

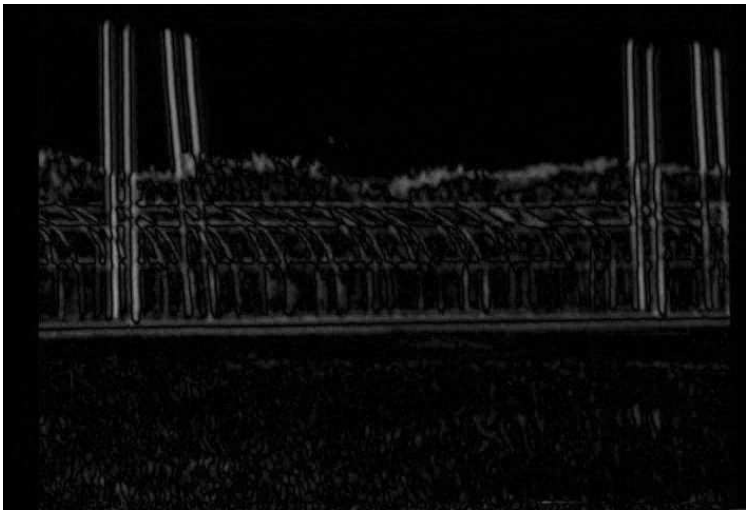

Figure 3.2: Difference of the frames in figure 3.1

3.3 Feature Extraction

3.3.1 Block Diagram

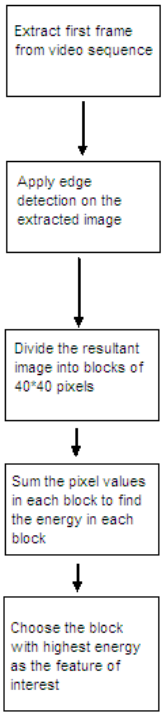

Figure 3.3: Feature extraction block diagram

3.3.2 Edge detection

Edge detection refers to the process of determining the boundaries between various objects and the background of an image. It is a subset of the more generic term "Image segmentation" which refers to the identification of sub-regions within an image. The basic idea of most edge detection algorithms is to represent the gradient of change around a pixel. This means that an edge which has a high gradient is represented with a high intensity while the interiors of objects which have fairly low changes in pixel intensities have a low gradient and are represented with a low value.

The two-dimensional gradient is expressed as

$$\nabla I(x, y) = (\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}) \tag{3.1}$$

There are many ways to define this quantity. We use the Sobel templates to approximate this quantity.

3.3.2.1 Sobel template

The Sobel method for edge detection uses two $3 * 3$ templates to determine the gradient value.The templates are

$$\nabla I_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \nabla I_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3.2}$$

The Figure 3.4 shows the result of applying the Sobel template to the first frame of the video input.

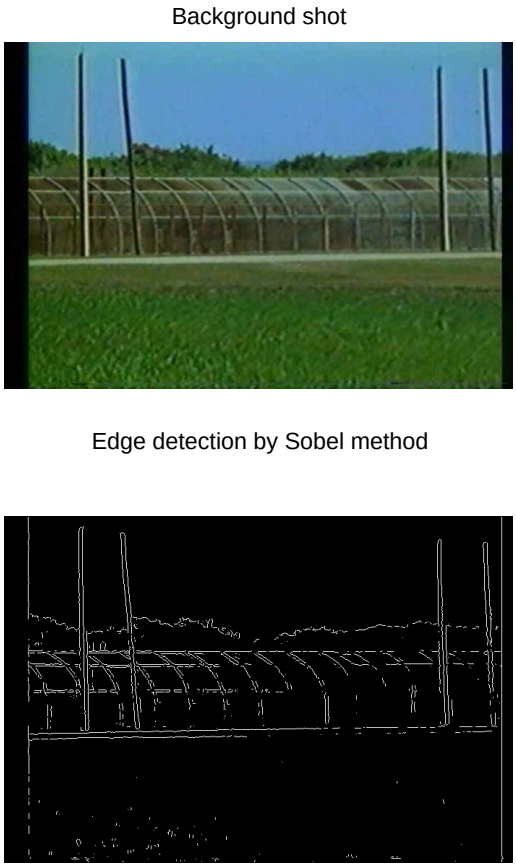

Figure 3.4: Result of edge detection of background using Sobel method

3.3.3 Choosing the feature

The most useful region for stabilization is one which has the most edges. This is so because having multiple regions with different intensity levels leads to a greater sensitivity in detecting image translation. To determine such a region we divide the edge image into blocks of $40 * 40$ pixels as shown in Figure 3.5

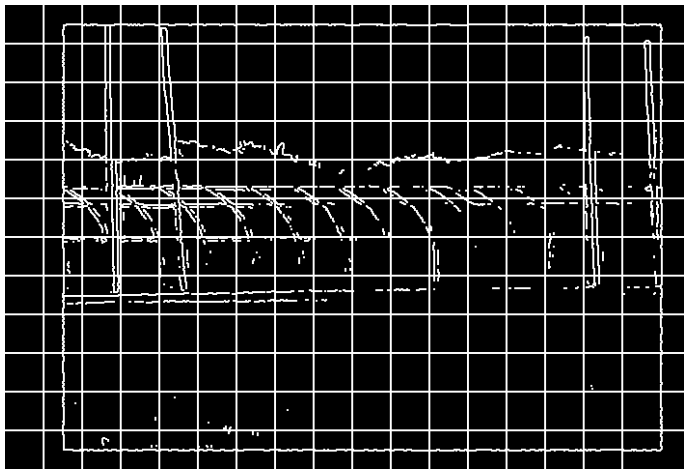

Figure 3.5: Dividing the edge image into blocks

Now we attempt to find the block with most number of edges. A good approximation for the intensity of edges in a block would be the sum of all intensities in the block.

$$E_b = \sum_x \sum_y I_b(x, y) \tag{3.3}$$

where $I_b(x, y)$ is the pixel intensity of the edge image.

It must be kept in mind here that the Sobel templates give binary intensities so I_b can only take the values 0 or 1. Thus the summation denotes the count of pixels in a block where edges have been detected by the Sobel templates. We choose the block b where E_b is maximum and choose it as the feature for the stabilization algorithm

The highlighted feature is shown in Figure 3.6

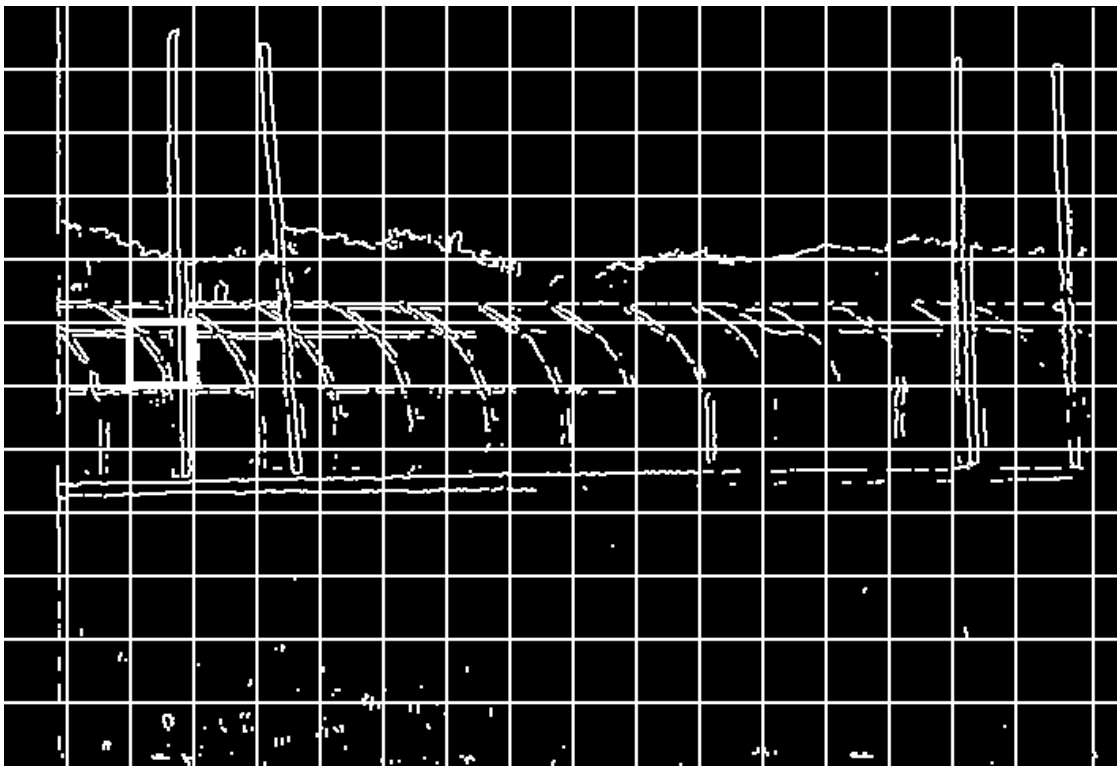

Figure 3.6: Highlighted feature among all the blocks

3.4 Stabilization algorithm

The algorithm for stabilizing the sequence is as follows.

- Extract the feature from the first frame
- Repeat the following steps for each subsequent frame
  - Given the size of the feature is $m*n$, choose a subsection of $(m+2p+1)*(n+2p+1)$ with the feature at it's center. p is the maximum number of pixels of shake in the frame sequence.
  - For all $-p \leq i \leq p$ and $-p \leq j \leq p$ compute the error signal $E(i,j) = \sum_{x=0}^{m} \sum_{y=0}^{n} (|I(x,y) - I(x-i, y-j)|)$
  - Find (i,j) for which $E(i,j)$ is minimum
  - Translate the frame by (i,j) and add to the stabilized video.

3.4.1 Effect of the size of the feature

Choosing the size of the feature presents an interesting optimization problem. It is intuitively clear that larger the size of the feature, the better the registration is likely to be. An alternative way of looking at it is that having a larger feature is equivalent to improving the resolution of the least-squares detector by averaging the least-squares over a larger area. The constraint here is that a larger feature area entails a longer execution time.

The number of multiplications N depends on the area as

$$N \propto m * n \tag{3.4}$$

The improvement of the stabilization accuracy with increase in area is illustrated by the following table

Table 3.1: Feature size vs accuracy

Size of feature	Average pixel error per frame
20*20	6.84
40*40	6.40
80*80	5.75

3.4.2 Automatic feature selection vs Manual feature selection

It is worthwhile to point out here that there was no loss of stabilization accuracy caused by the use of a generalized feature extraction algorithm instead of picking out the feature manually for the input sequences that were tested. This would be true of most cases where the size of the moving objects are significantly smaller than the features. The algorithm would not work as well if a very large object would move across the feature, thereby occluding large parts of the feature. Such a situation is however not expected in the application considered by this thesis.

CHAPTER 4

DETECTING MOVING OBJECTS IN THE STABILIZED SEQUENCE

This chapter describes step by step the method used to highlight the moving objects in the stabilized video. It starts by reiterating the effect of the now familiar step of frame differencing, moves on to various methods of removing false alarms and ends with an explanation of the method used to highlight moving objects.

4.1 Differencing consecutive frames

Figure 4.1 shows two consecutive frames from the background stabilized sequence. We look at the result of taking the difference between these two frames.

Frame 127

Frame 128

Figure 4.1: Two consecutive frames from stabilized video

Figure 4.2 shows the difference image. It must be noted that the areas where the motion of the birds is noticed have been highlighted manually to enhance the readability of this thesis. The highlighting of areas of motion is not done during the normal operation till a later stage. The purpose of frame differencing is the cancellation of the stationary background which has been achieved to a certain extent.

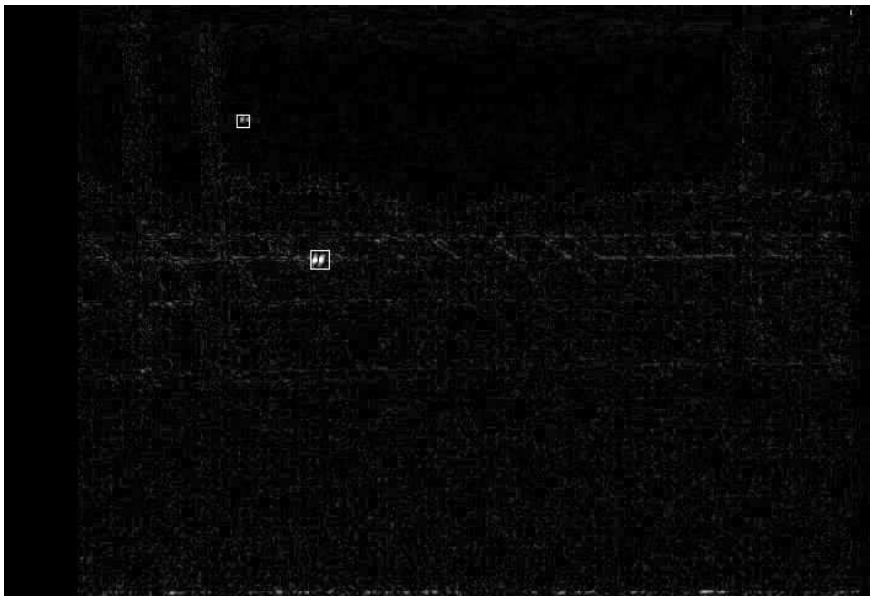

Figure 4.2: Difference of the two frames

It is easily noticeable that the fence in the background is clearly visible. This is due to the following three reasons :-

- Residual mis registration in the stabilization process
- Deformation in the fence and the poles due to the violent ground shake
- Change in pixel intensities due to motion blur

We look at ways to remove the high intensities from these areas so that they do not cause false alarms in the detection process.

4.2 Masking out false alarms

This section details three ways of removing false alarms from the difference images.

4.2.1 Mask based on location of background edges

It is apparent that there are a lot of false alarms occurring in areas where the background has edges. In the test case these areas would be along the fence and the poles. One solution would be to mask out these areas completely.

The mask is generated by dilating the edge image in Figure 3.4 by a square matrix of size $11 * 11$ as shown in Figure 4.3 .

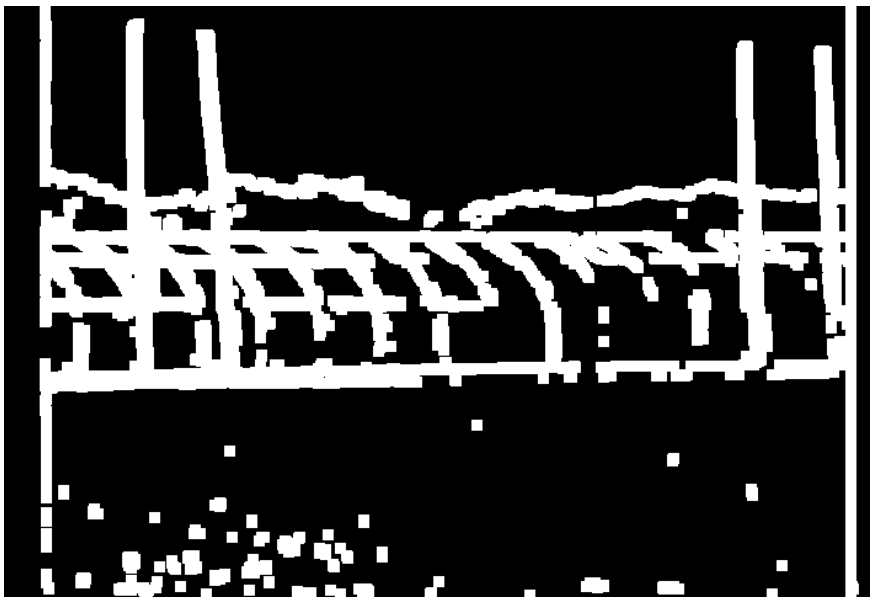

Figure 4.3: The mask used to reduce false alarms

Let us apply this mask on the difference image shown in Figure 4.4

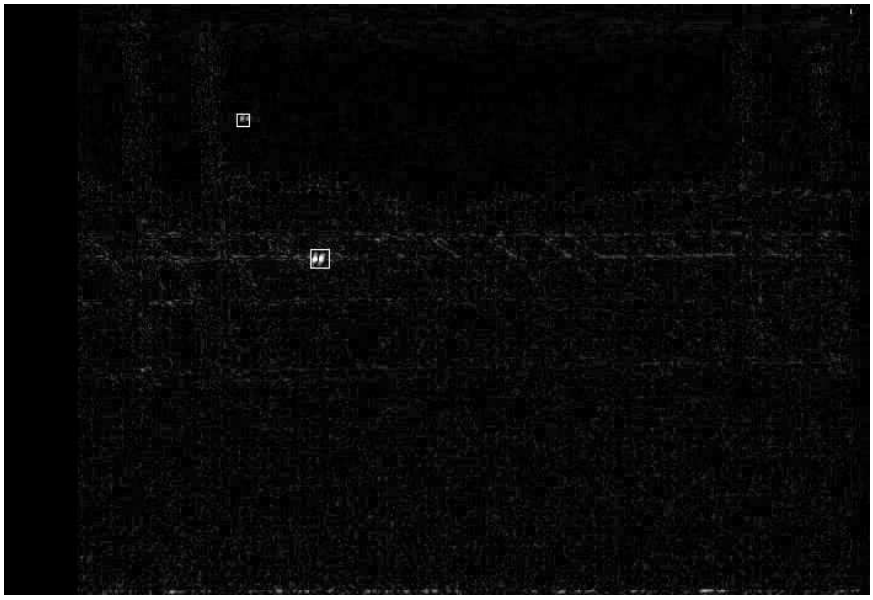

Figure 4.4: Difference image

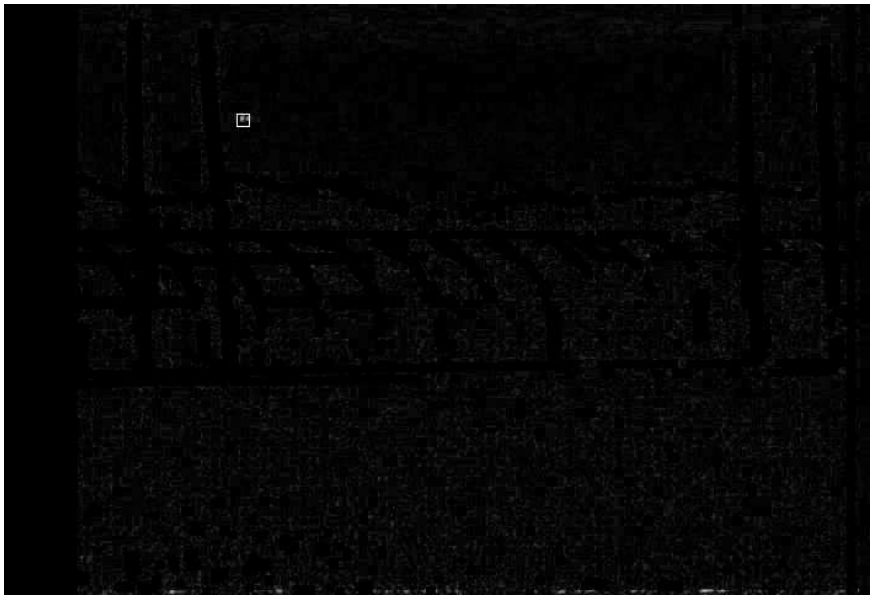

Figure 4.5: Difference image with application of background mask

Figure 4.5 shows the result of applying the mask. The noise along the fence has been eliminated and does not cause any false alarms. There is however a severe limitation that is evident in this example. The bird that was flying along the fence has also been masked out. While it seems to be highly unlikely that debris from a shuttle would follow such a path, it is desirable to find a method that does not have such a limitation.

4.2.2 Filtering based on the trajectory of motion

An alternate approach to solving the problem of false alarms would be to integrate a set of difference frames instead of considering them one at a time. It is possible then to look for areas where the high intensity pixels follow a certain trajectory. Such an approach would ideally filter out the isolated occurrences of high intensities that cause false alarms. Such areas of high intensity typically occur when there is a large amount of motion blur caused by a violent shake in the unstabilized video. The Figure 4.6 shows the result of integrating 20 consecutive difference frames from the background stabilized sequence. The mask discussed in the previous section has also been applied. The combination of these two techniques allows us to reduce greatly the number of false alarms in tracking moving objects.

Figure 4.6: Result of integrating a set of 20 difference images

The trajectory of the bird moving along the fence is evident. It is possible to filter out only that particular area by determining regions where the regions of high intensity follow a right to left horizontal path. The region where the bird is moving is therefore the only region that is highlighted.

4.2.3 Taking the average difference of p consecutive frames

Another approach towards reducing the false alarms is to average the differences of a frame with many neighboring frames to reduce the intensity of the alarms caused by the residual camera shake. The greatest advantage of this method is that the attenuation of difference intensity in areas of vibratory motion is far greater than the attenuation of the intensity where there is linear motion.

As always, the easiest way to illustrate this is with an example.

Figure 4.7: Result of taking average difference of Frame 128 with 6 neighboring frames

As Figure 4.7 shows this method gets rid of almost all of the noise in the difference image. One drawback is that the intensity at the area having moving objects is also reduced. While this does not usually cause problems in detection, it is possible that debris that is extremely small in size might be missed due to the averaging.

4.3 Highlighting regions with moving objects

In this section we detail the final step of the motion detection process, the method used to highlight the areas where moving objects are detected in the final video. This process can best be expressed as the following algorithm

- Apply the mask on the difference image
- Divide the image into blocks of $5 * 5$ pixels
- For each block b compute the average intensity $\bar{I}_b(t) = \frac{\sum_b I(x,y,t)}{25}$
- Set the alarm flag for each block by thresholding the average intensity

$$\gamma_b(t) = \begin{cases} 1 & \text{if} \bar{I}_b(t) > c_0 \\ 0 & \text{otherwise} \end{cases}$$

- Filter out some alarm flags by determining the trajectory of motion
- If the flag $\gamma_b(t)$ is still 1 highlight block b in frame t.

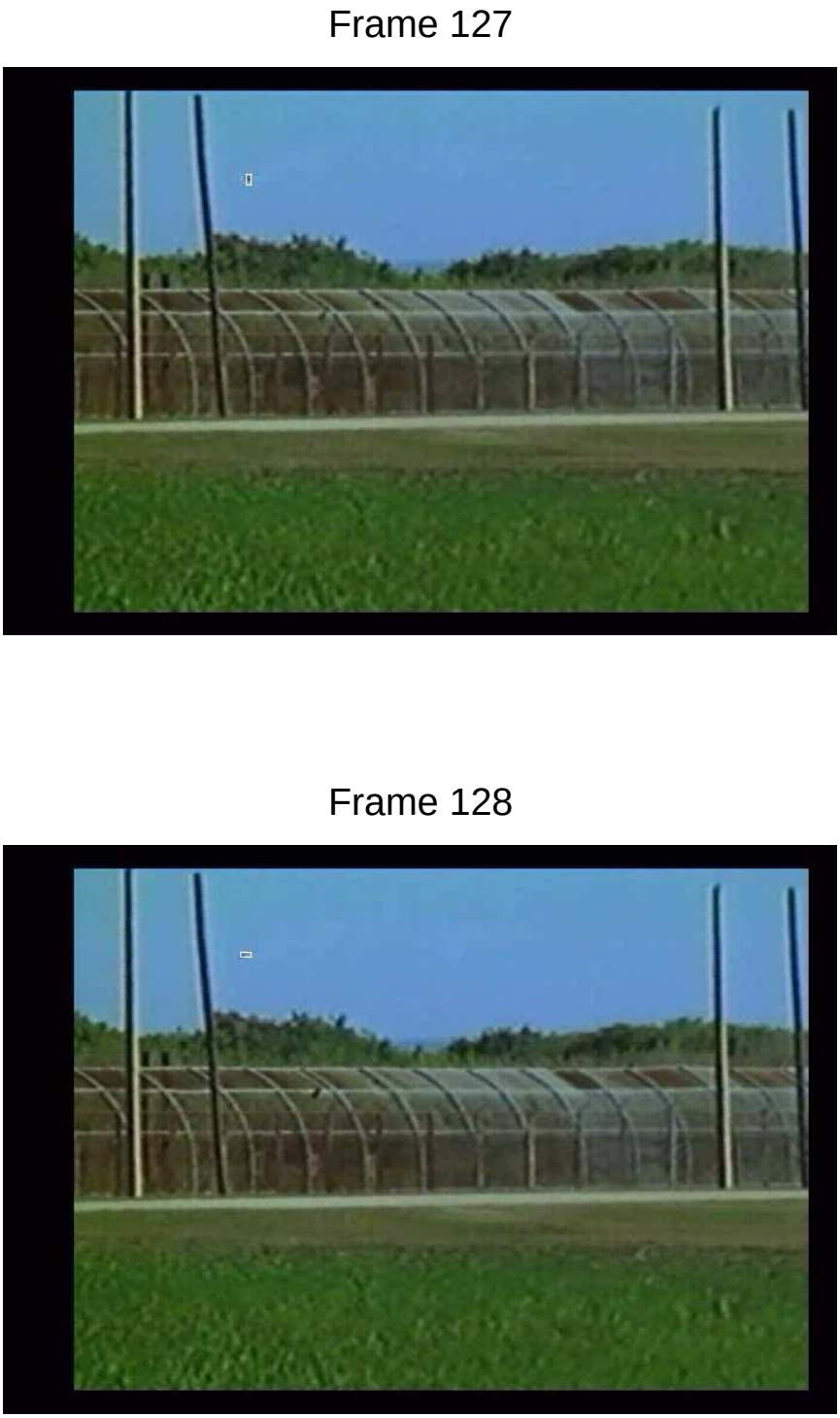

Figure 4.8: Frames in the output video with highlighted moving objects

The Figure 4.8 shows samples from the output video. The moving bird near the top of the image is highlighted by the white box around it. This emphasises the moving objects and makes it easier to see them in the image. The output may also be used to send information directly to other automated programs.

This section completes the description of the system developed by the author. The next chapter contains conclusions and a description of other methods being considered by the author for improvement of the system.

CHAPTER 5

GUI FOR BACKGROUND DETECTION AND DEBRIS TRACKING

A major parameter to measure the worth of any application software is the quality of the interface it provides to its users. Any quality software program needs an user interface that is intuitive to use, that offers its users the greatest amount of flexibility and ease of operation. Designing good interfaces often entails allowing users to give a broad range of inputs in different formats to the software system and allowing a variety of output options.

The Digital Signal Processing laboratory at the University of Central Florida aims to create an user friendly interface that allows users to stabilize input video sequences and automatically track debris under a reasonable set of constraints. We aim to build a system that accepts various video formats like AVI, MPEG and image sequences as inputs and intend to allow the user reasonable choice in choosing the output file format.

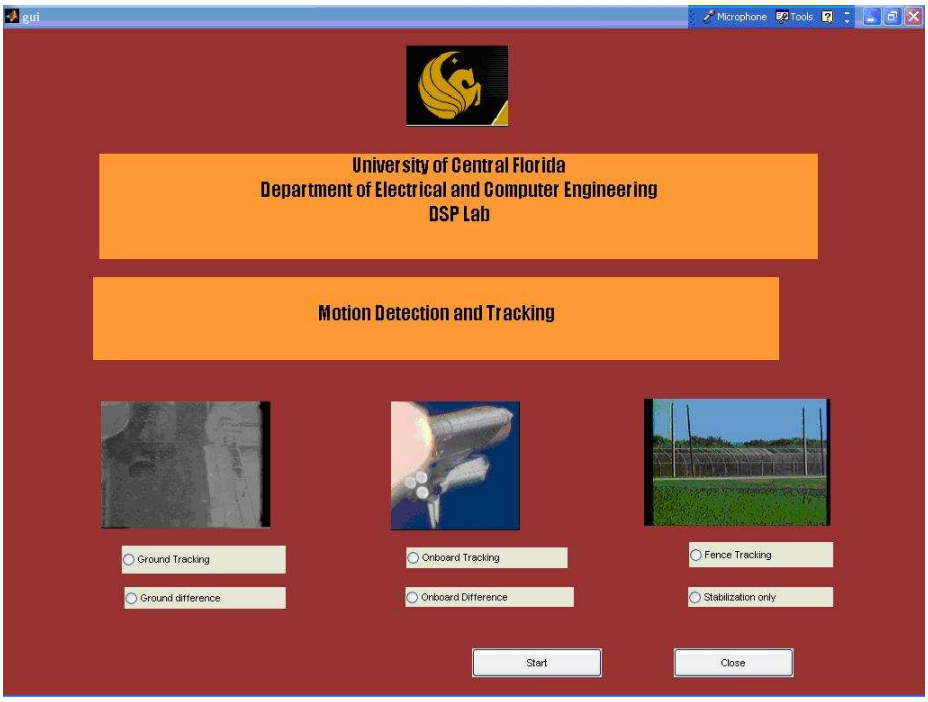

Figure 5.1: Graphical User Interface for background stabilization and debris tracking

Figure 5.1 is a screen shot of the demonstration version of the Graphical User Interface that is being developed as part of this project. It contains input videos from three different views, and allows the user to view the output videos in which moving objects that could be debris have been tracked. The current version of the GUI has the ability to read inputs as image sequences or AVI files. It is our intention to also allow the input videos to be in the MPEG file format.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The method detailed in this thesis provides reasonable results with the test video provided. The video is stabilized adequately and moving objects are detected in a majority of frames. While a serious effort has been made to avoid customizing the algorithm for the test videos, further testing with a more generic set of input sequences is required to prove the algorithm's generality.

Significant improvements need to be made before the system detailed in this thesis can be considered deployable. A serious flaw in the system's performance is that objects that move continuously across edges in the background are masked out and go undetected. A possible solution for this problem would be to compensate for motion blur in the input frames to reduce residual noise in the stabilization process. This would reduce the need for masking and also remove some false alarms.

In summation, the author considers the work completed so far a good beginning, rather than the final solution for the problem at hand.

6.2 Future work

The preceding section contains some of the shortcomings of the algorithm used in this thesis. In addition to correcting these, it is also possible to increase the scope of the algorithm to cases that have not presently been considered. One natural extension would be to extend the algorithm to non-stationary and Pan-Zoom-Tilt cameras. These cameras are used widely in surveillance and security applications. An algorithm that can track objects from video sequences fom such cameras would be useful for a far wider range of applications than an algorithm that only considers stationary cameras.

APPENDIX A

MATLAB CODE:STABILIZATION

```
% stabilizer1.m

function xxx= stabilizer1(inputFileName,outputFileName)


%GUI file inputs if arguments are not provided

if nargin < 2

    if nargin == 0

        %Get the input file

        [fileName,pathName]=uigetfile('*.avi','Select the input video');

        inputFileName = strcat(pathName,fileName);

    end

    % Get the output file name

    [fileName,pathName]=uiputfile('*.avi','Select the output file');

    outputFileName = strcat(pathName,fileName);

else

    if nargin > 2

        error('Too many arguments');

    end

end


%Determine number of frames
```

```
if inputFileName
    fileInfo =aviinfo(inputFileName);
    numberOfFrames = fileInfo.NumFrames;
end

% Output initialized
movie = avifile(outputFileName,'compression','Indeo5',...
'FPS',fileInfo.FramesPerSecond);

%Read First frame
initialFrame = aviread(inputFileName,1);
initialImage = frame2im(initialFrame);
clippedImage = initialImage;
clippedImage(1:20,:,:) = 0;
clippedImage(461:480,:,:) = 0;
clippedImage(:,1:60,:) = 0;
clippedImage(:,681:704,:) = 0;

xxx = zeros(numberOfFrames,1);
yyy = zeros(numberOfFrames,1);
```

```
backgroundImage = edge(rgb2gray(initialImage),'sobel');


for i=1:floor(480/40)
    for j=1:floor(704/40)
        edgeCount(i,j) = sum(sum(uint8(backgroundImage(...
         40*(i-1)+1:40*i,40*(j-1)+1:40*j ))));
    end
end
[blockX,blockY] = find(edgeCount==max(max(edgeCount)));
imageSubsection=rgb2gray(initialImage(
...blockX*40-80:blockX*40+40,blockY*40-80:blockY*40+40,:));
featureWindow=imageSubsection(40:80,40:80);
correlationMatrix = double(zeros(80,80));


for i=1:numberOfFrames
    tic;


    inputImage = frame2im(aviread(inputFileName,i));
    imageSubsection = rgb2gray(inputImage(...
    blockX*40-80:blockX*40+40,blockY*40-80:blockY*40+40,:));
```

```
for j=1:80
    for k=1:80
        correlationMatrix(j,k) = sum(sum(abs(double(...
        imageSubsection(j:j+40,k:k+40))-double(featureWindow))));
    end
end
newXY= find(correlationMatrix== min(min(correlationMatrix)));
shiftVector = [rem(newXY(1),80),floor(newXY(1)/80)] - [40 40];

xxx(i)=shiftVector(2);
yyy(i)=shiftVector(1);

if shiftVector(2)>0
    inputImage(:,1:704 - shiftVector(2),:)...
     =inputImage(:,1 + shiftVector(2):704,:);
    inputImage(:,690 - shiftVector(2):704,:)...
    = initialImage(:,690 - shiftVector(2):704,:);
    inputImage(:,1:34,:)= initialImage(:,1:34,:);
else
    inputImage(:,1 - shiftVector(2):704,:)...
     =inputImage(:,1:704 + shiftVector(2),:);
```

```
    inputImage(:,1:34 - shiftVector(2),:)...
    =initialImage(:,1:34 - shiftVector(2),:);
    inputImage(:,690:704,:)=initialImage(:,690:704,:);
end
if shiftVector(1)>0
    inputImage(1:480 - shiftVector(1),:,:)...
     =inputImage(1 + shiftVector(1):480,:,:);
    inputImage(481- shiftVector(1):480,:,:)...
    =initialImage(481- shiftVector(1):480,:,:);
else
    inputImage(1 - shiftVector(1):480,:,:)...
     =inputImage(1:480 + shiftVector(1),:,:);
    inputImage(1:0 - shiftVector(1),:,:)...
    =initialImage(1:0 - shiftVector(1),:,:);
end



%Try trimming the edges
inputImage(1:20,:,:) = 0;
inputImage(461:480,:,:) = 0;
inputImage(:,1:60,:) = 0;
```

```
    inputImage(:,681:704,:) = 0;

    xxx(i) = sum(sum(sum(abs(double(inputImage)-double(clippedImage) ))));


    %imwrite(inputImage(:,:,:),strcat('stab',int2str(i),'.jpg'));

    movie = addframe(movie,im2frame(inputImage));

    toc,i

end

movie = close(movie);
```

APPENDIX B

MATLAB CODE:DETECTION

```
function averageDifferencer(inputFileName,outputFileName)



fileInfo =aviinfo(inputFileName);

numberOfFrames = fileInfo.NumFrames;


% Output initialized

movie=avifile(outputFileName,'compression','Indeo5','FPS',fileInfo.FramesPerSecond);

movie1=avifile('foobar1.avi','compression','Indeo5','FPS',fileInfo.FramesPerSecond);

%Read First frame

initialFrame = aviread(inputFileName,1);

initialImage = rgb2gray(frame2im(initialFrame));


foo = zeros(480,704);

for i=1:213,

x = rgb2gray(frame2im(aviread('afternoon.avi',i)));

y = double(edge(x,'sobel'));

foo = foo + y;

end

fooo = foo/(13);

fooo(fooo<0.2)=0;
```

```
fooo(fooo>0.2)=1;
%background = edge(initialImage,'sobel');
mask = ones(480,704)-double(imdilate(fooo,strel('rectangle',[3 5])));
imshow(mask);
previousWindowIntensity = zeros(floor(size(initialImage)/5));
for i = 4:numberOfFrames-3
    tic;

    %Iniatializations
    differenceImage = zeros(size(initialImage));

    currentFrame = aviread(inputFileName,i);
    currentImage = double(rgb2gray(frame2im(currentFrame)));

    for j = -3:3
        differenceImage = differenceImage + abs(currentImage -...
         double(rgb2gray(frame2im(aviread(inputFileName,i+j)))));
    end
    differenceImage = differenceImage/6;
    if i==128
        imshow(differenceImage);
```

```
end

differenceImage(differenceImage<30)=0;

differenceImage(1:20,:)=0;

differenceImage(251:480,:)=0;

differenceImage(:,1:65)=0;

differenceImage(:,671:704)=0;

differenceImage = differenceImage.*mask;

outputImage = frame2im(currentFrame);

windowIntensity = zeros(floor(size(differenceImage)/5));


for j = 1:floor(size(differenceImage,1)/5)-1

    for k = 1:floor(size(differenceImage,2)/5)-1

        windowIntensity(j,k)=sum(sum( differenceImage...

        (5*(j-1) +1 :5 *j, 5*(k-1)+1 : 5*k) ))/25;

        windowIntensity(j+1,k)=sum(sum( differenceImage...

        (5*(j) +1 :5 *(j+1), 5*(k-1)+1 : 5*k) ))/25;

        windowIntensity(j,k+1)=sum(sum( differenceImage...

        (5*(j-1) +1 :5 *j, 5*(k)+1 : 5*(k+1)) ))/25;

        if windowIntensity(j,k)>10

            if k==1 || windowIntensity(j,k-1)<=10

                outputImage(5*(j-1) +1 :5 *j, 5*(k-1)+1,:) = 255;
```

```
            end
            if k==floor(size(differenceImage,2)/5) || windowIntensity(j,k+1)<=10
                outputImage(5*(j-1) +1 :5 *j, 5*k,:) = 255;
            end
            if j==1 || windowIntensity(j-1,k)<=10
                outputImage(5*(j-1) +1 , 5*(k-1)+1:5*k,:) = 255;
            end
            if j==floor(size(differenceImage,1)/5) || windowIntensity(j+1,k)<=10
                outputImage(5*j , 5*(k-1)+1:5*k,:) = 255;
            end
        end
    end
    previousWindowIntensity = windowIntensity;
end
```

```
differenceImage = reshape([differenceImage differenceImage...
 differenceImage],[size(differenceImage) 3]);
differenceFrame = im2frame(uint8(differenceImage));
```

```
    movie1=addframe(movie1,differenceFrame);

    movie = addframe(movie,im2frame(outputImage));

i,toc;

end


movie = close(movie);

movie1 = close(movie1);
```

LIST OF REFERENCES

[AP96] Mitiche A. and Bouthemy P. "Computation and analysis of image motion: a synopsis of current problems and methods." *International Journal of Computer Vision*, **19**(1):29–55, 1996.

[BFB92] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. "Performance Of Optical Flow Techniques." *Computer Vision and Pattern Recognition*, pp. 236–242, 1992.

[Boa03] Columbia Accident Investigation Board. "Report Volume 1." Technical report, August 2003.

[CV99] Fusiello A. Censi A. and Roberto V. "Image stabilization by features tracking." In *Proceedings of the International Conference on Image analysis and processing*, pp. 665–667, September 1999.

[HS81] Berthold K.P. Horn and Brian G. Schunck. "Determining Optical Flow." *Artificial Intelligence*, **17**:185–203, 1981.

[Jav05] Omar Javed. *Scene monitoring with a forest of cooperative sensors*. PhD thesis, University of Central Florida, 2005.

[JG03] Cohen I. Jinman Kang and Medioni G. "Continuous tracking within and across camera streams." In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pp. I–267– I–272, June 2003.

[Lim90] Jae S. Lim. *Two-dimensional Signal and Image Processing*. Prentice Hall Signal Processing series. 1990.

[LK81] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision." In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pp. 674–679, April 1981.

[LM99] Wixson L. and Hansen M. "Detecting salient motion by accumulating directionally-consistent flow." In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pp. 797–804, 1999.

[MC01] G. Marcenaro L., Vernazza and Regazzoni C.S. "Image stabilization algorithms for video-surveillance applications." In *Proceedings of the International Conference on Image processing*, volume 1, pp. 349–352, October 2001.

[R97] Parker J. R. *Algorithms for Image processing and Computer vision.* Wiley Computer Publishing, 1997.

[Sha] Mubarak Shah. "Fundamentals of computer vision." Course Notes for CAP 5415: Computer Vision Fall 2004.

[Yil04] Alper Yilmaz. *Object tracking and Activity recognition in video acquired using mobile cameras.* PhD thesis, University of Central Florida, 2004.