Video Stabilization

Mark Mc Donnell
B.A. (Mod.) Computer Science
Final Year Project, May 2004
Supervisor: Dr Kenneth Dawson-Howe

Declaration

I hereby declare that this thesis is entirely my own work and that it has not been submitted as an exercise for a degree at any other university.

_____ April 29, 2004

Mark Mc Donnell

Abstract

Home videos are notorious for camera jitter as most are taken with hand-held camcorders. Given an un-stabilized video sequence, the objective of this project is to synthesize a new sequence as seen from a stabilized camera trajectory.

Over the years there have been many different video stabilization techniques developed to solve this problem. However the removal of the unwanted video perturbations due to undesirable camera motions has proved difficult. This report looks at the previous method used and presents a method based on feature tracking using a corner detector to find features and template matching to validate these features. Mosaicking was used in order to rebuild undefined areas that result from motion compensation applied to each video frame.

The project successfully generates a stabile sequence given an unstable video sequence. The present solution to the problem removes unwanted affine translations. Although not implemented, due to time constraints, a technique for removing affine rotation is offered which could be used to increase the stabilization. The mosaicking effect would also be improved as a result.

Acknowledgements

I would like to thank my supervisor Kenneth Dawson-Howe for his suggestions, advice and direction throughout this project's development. I must also thank my family and friends for their support and encouragement.

Contents

1 Introduction **6**
- 1.1 Purpose . . . 6
- 1.2 Aims . . . 6
- 1.3 Motivation . . . 7
- 1.4 Report Structure . . . 7

2 Background Information **9**
- 2.1 Feature Tracking . . . 9
  - 2.1.1 2D Ridge Motion . . . 10
  - 2.1.2 Unstructured Lumigraph Rendering . . . 10
  - 2.1.3 Kalman Filtering & Homography . . . 10
- 2.2 Image Intensities & Ego-motion . . . 11
  - 2.2.1 Calculating Ego-motion . . . 11
  - 2.2.2 Camera Stabilization . . . 11
- 2.3 Optical Flow . . . 13
- 2.4 Other Techniques . . . 13
- 2.5 Trinity Image Processing System . . . 13
- 2.6 Intel Libraries . . . 14

3 Main Areas **15**
- 3.1 Tracking of Features . . . 15
  - 3.1.1 Finding Features . . . 16
  - 3.1.2 Re-Find Features . . . 16
- 3.2 Validating Features . . . 16
- 3.3 Unwanted Motion . . . 17
- 3.4 Stabilize Footage . . . 17

4 Feature Tracking **18**
4.1 Finding Features 18
4.1.1 Corner Detector 18
4.1.2 Find Initial Features 19
4.2 Re-Find Features 20
4.2.1 Normalized Cross-Correlation 21

5 Feature Validation **22**
5.1 False Features 22
5.1.1 Examples 22
5.1.2 Effects 23
5.1.3 Solutions 24
5.1.4 Other Possibilities 24
5.2 Invalid Features 24

6 Jitter **26**
6.1 Original Camera Motion 26
6.2 Desired Camera Motion 26
6.3 Calculating Jitter 28

7 Stabilization **30**

8 Evaluation **31**
8.1 Results 31
8.1.1 Feature Selection 31
8.1.2 Tracking 31
8.2 Successful Aspects 33
8.3 Difficulties Encountered 34
8.3.1 Intel Manuals 34
8.3.2 TIPS 35
8.3.3 Magic Numbers 35
8.4 Future Work 35
8.4.1 Finding Features 36
8.4.2 Rotation 36
8.4.3 Interpolation 36

9 Conclusion **37**

Bibliography **39**

A Feature Selection **41**

B Least Squares **43**

List of Figures

2.1 Camera Stabilization 12

4.1 Feature tracking 20

5.1 Occlusion 23

6.1 The red line is the y displacement and the blue line is its linear interpolation, from the eleventh video 27

6.2 The solid green line segment represents the jitter of frame 30 29

8.1 Frame from first video a) and with the superimposed features b) 32

8.2 Frame from second video a) and with the superimposed features b) 32

8.3 Feature window, from top right of Figure 8.1 33

8.4 First frame from first video with 31 features superimposed a) and last frame with the remaining 20 features b) 33

8.5 First frame from second video with 19 features superimposed a) and last frame with the remaining 9 features b) 34

Chapter 1

Introduction

Computer Vision aims to allow computers to mimic the human vision system through the automated analysis of images and video sequences. This is no easy task as the images are two-dimensional and they have to be used to interpolate a three-dimensional world. Although we are still a long way off being able to fully emulate the human vision system it is possible to solve certain problems.

1.1 Purpose

Video sequences captured from hand-held camcorders can often have unwanted motion due to various reasons, such as vibrations in a moving vehicle or an unsteady hand. With today's modern cameras this unwanted motion is also amplified by high-power zoom lenses. As a result of these motions, there have been different video stabilization techniques developed.

1.2 Aims

The main aim of these stabilization algorithms is to remove the unwanted motion but to keep intact the desired motion. As it is impossible to know exactly the intended camera motion therefore the camera motion has to be calculated and an interpolation of the desired motion made. With a decision on the desired camera motion made, the jitter can be calculated and hence removed.

1.3 Motivation

Removal of the jitter in a hand held camcorder can only really have one purpose and that is to increase the aesthetic appearance of the footage. However, when considering other utilisations of a stabiliser there are more benefits than the obvious professional look.

- Detection and Tracking independently moving objects from moving platforms can be greatly simplified by removing all motion from the camera output, so that the scene background looks stationary. Important in both robot navigation and scene modelling.
- A steady video can greatly increase the clarity of the picture. For good high quality security video footage, sharp details are necessary.
- Increased compression. Modern digital compressors use a lot of bits to encode moving features of a video. If the whole image shakes, it is all moving. This wastes an enormous number of bits. As a result stabilized video has a much better compression rate.
- Human fatigue. Unstable footage can be strenuous to watch and having to examine it for hours on end can lead to difficulties. Ensuring operators are alert is a major challenge in real-life security CCTV. Stable footage can greatly help in this situation.

1.4 Report Structure

Chapter 1 Introduction. This outlines the aims and purpose of the project.

Chapter 2 Background Information. This is a study into previous work that has been carried out in the area of Video Stabilization.

Chapter 3 Main Areas. This is a break down of the project algorithm, detailing the problems and their solutions.

Chapter 4-7 These outline in detail each section of the project.

Chapter 8 Evaluation. This is devoted to the results of the project and how it performed. It also looks at the difficult aspects of that project and details possible additions that could be undertaken.

Chapter 9 Conclusion.

Chapter 2

Background Information

Advances of video stabilization techniques developed in previous years are examined in this chapter. It will also include areas that have been used by systems in the past and that have been utilized in the development of this project.

Image stabilization consists of compensating for the camera motion by applying a suitable transformation (warping) to the image. In the stabilized output footage, scene points are motionless in spite of desired camera motion.

2.1 Feature Tracking

Feature tracking algorithms (Morimoto & Chellappa, 1996), (Buehler, Bosse, & McMillan, 2001) and (Censi, Fusiello, & Roberto, 1999) have good results when it comes to video stabilization. They generally work well for different camera movement including translation, rotation and different zooming models.

In the area of feature tracking (Shi & Tomasi, 1994) is a feature selection algorithm, a tracking algorithm based on a model of affine image changes and a technique for monitoring features during tracking. However this may not always work for dissimilarities as some features may change slightly over a large number of frames.

2.1.1 2D Ridge Motion

There has been a range of algorithms developed to suit different purposes. An algorithm using feature based 2D ridge motion model (Morimoto & Chellappa, 1996) is used to deal with translational camera motion. The algorithm tracks a small set of features to estimate the motion of the camera. Stabilization is achieved by combining all motion from a reference frame and subtracting this motion from the current frame.

2.1.2 Unstructured Lumigraph Rendering

Another algorithm based on feature tracking is similar to (Buehler et al., 2001) which uses Unstructured Lumigraph Rendering (URL) to stabilize the video sequence. This is a non-metric method and the desire is to calculate the projective reconstruction, to stabilize the camera trajectory and to feature smoothing using URL. It has been suggested that this algorithm be able to perform the following tasks:

- correspond points between reference images
- interpolate between reference images
- navigate a virtual camera in the scene.

The main flaw to this algorithm is the difficulty of fitting motion models to complex motions. There is also the problem of ghosting or doubling images which may appear if there are fast moving objects.

2.1.3 Kalman Filtering & Homography

The image stabilizer (Censi et al., 1999) uses a modified version of the trackers described so far. After features are extracted from the first frame, Kalman filtering (Gelb, 1974) is used to estimate and predict the feature's position in all the subsequent frames. An outlier rejection rule (X84) (Hampel, Ronchetti, Rousseeuw, & Stahel, 1985) is used in order to estimate the homography robustly. Homographies are points that are linked by a linear projective transformation.

2.2 Image Intensities & Ego-motion

Image intensities were used in (Irani, Rousso, & Peleg, 1994) to calculate the 3D camera motion (ego-motion) of a video sequence. Once this motion is known it can be used in the imaging stabilization of the sequence.

2.2.1 Calculating Ego-motion

The first step is to get an image region corresponding to a planar surface in the scene. The regions 2D motion parameters are computed between consecutive frames. Using these values a 2D transformation is then used for image warping. This removes the rotational component of the 3D camera motion for the entire scene, leaving the problem as a pure 3D translation. The 3D translation (focus-of-expansion) is then calculated from the registered frames. Then it is just a matter of solving a small set of linear equations to find the 3D rotation. The calculation of the ego-motion can be determined in all but two situations:

- entirely planar scenes
- ego-motion with a translation in the x-y plane only

2.2.2 Camera Stabilization

A post-image stabilization can now commence with the information gathered from the ego-motion calculations. Stabilization can be preformed by warping the image back to the original position of the first frame to cancel the computed 3D rotations. The new sequence contains only 3D translations, and seems as if it was taken from a stabilized platform. An example of such stabilization is shown in Figure 2.1 on the following page. Alternatively, the rotations can be filtered by a low-pass filter so that the resulting sequence will appear to have only smooth rotations with no jitter.

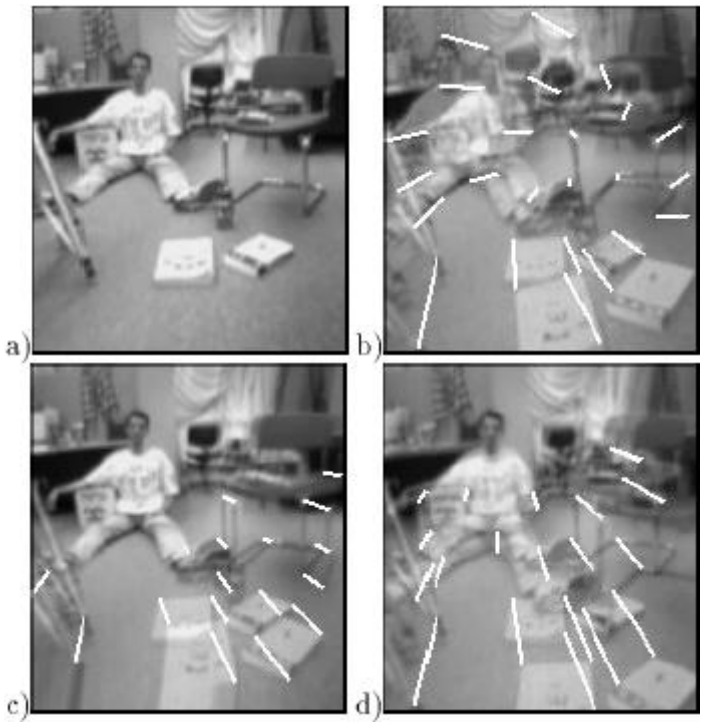

Figure 2.1: Camera Stabilization
From (Irani et al., 1994)

a One of the frames in the sequence.

b The average of two frames, having both rotation and translation. The white lines display the image motion.

c The average of the two frames after registration of the shirt. Only effects of camera translation remain.

d The average of the two frames after recovering the ego-motion, and cancelling the camera rotation. This results in a stabilized pair of images.

2.3 Optical Flow

The optical flow based technique (Srinivasan & Chellappa, 1997), regularizes the ill-conditioned gradient constraint equation by modelling optical flow as a linear combination of an overlapped set of basis functions. The solution is achieved by a numerically stable sparse matrix inversion. This results in a reliable flow field approximation over a large portion of the frame. A least squares method is applied to merge the local model parameters into a global model. The method facilitates a six parameter affine transformation.

2.4 Other Techniques

Some video stabilization algorithms like (Zhu, Xu, Yang, & Jin, 1998) use global and local motion estimations followed by filtering to remove unwanted video motions.

A probabilistic approach was taken in (Litvin, Konrad, & Karl, 1993) to model inter-frame motion parameters. A dynamic motion model reasonably depicts the inter-frame motion produced by camera movements. The results from a recursive estimation theory were used to achieve optimal evaluation of the intended image transformation parameters. The technique is flexible and allows the inclusion of existing prior information about the intentional camera motion and characteristics of unwanted camera motion. There was also a mosaicking to reconstruct undefined regions in the frames.

2.5 Trinity Image Processing System

It should be noted that this project was developed as an expansion to the Trinity Image Processing System (TIPS), developed by Kenneth Dawson-Howe (TCD). TIPS was developed to allow students to process images and video clips and to integrate their own processing functions in C++. A developer edition of TIPS was made available for this project, to allow for any changes that may be necessary. The developer

edition includes projects and source files.

2.6 Intel Libraries

The Intel® Image Processing Library (Intel, 2000a) and the Intel® Open Source Computer Vision Library (Intel, 2000b) were also utilized throughout the project. These libraries contain functions for the purpose of developing image processing applications and image processing libraries. These libraries support many functions whose performance can be significantly enhanced on the latest generations of processors.

Both libraries are already included in TIP but not all the functions are integrated. Each comes with a manual which provides a background of the image and execution architecture of the libraries, as well as detailed descriptions of the library functions.

Chapter 3

Main Areas

The purpose of this chapter is to give an impression of the project, to pinpoint the different problems that need to be tackled and to show the solutions found. A rough guide is presented to show how the main areas of this project fit together. It will provide an outline of what will be covered in more detail by the following chapters.

The key areas necessary to establish stable footage are:

- tracking the features through the video sequence
- validation of the features
- calculation of unwanted motion
- stabilization of the footage

The first three steps are performed on the first pass through a video sequence and the overall stabilization is carried out on the second pass. The reason for each of these steps is investigated in the following sections.

3.1 Tracking of Features

This task can be split into two subsections the initial finding of the features and then the subsequent re-finding of the feature windows through the video sequence.

3.1.1 Finding Features

A feature is a point in an image with a certain dissimilarity or distinction. Good features can be found when looking for areas of high texture or corners. The reason for this is that they are easy to find again.

The initial finding of features is very important because the selection result has significant influence on the performance of the tracking. Therefore it is imperative that the best feature be found. It was found that features should be well dispersed and not near the border of the image.

3.1.2 Re-Find Features

After finding a set of features in the first frame, it is necessary to track them through the video sequence. A search is done in each subsequent frame to re-find every feature. When completed the movement of every feature will be logged for future processing.

3.2 Validating Features

Unfortunately even features which are being successfully tracked may turn out to be bad. There are a few possible reasons for this:

- occlusions
- perspective distortions
- strong intensity changes
- surface discontinuities
- shadows
- specular reflectance

In any case these are features that are not representative of fixed points in the real world or have just disappeared. These features could be harmful, perhaps introducing more jitter, if not discarded.

3.3 Unwanted Motion

The unwanted motion is the difference between the original camera motion and the desired camera motion. With a secure set of features tracked it is possible to build up a good estimate of the original camera motion. This will be merely the average of all the valid features. There are a number of possible ways of interpolating the desired camera motion but for reasons that will be discussed later it was decided that linear interpolation between the first and last frame would be used.

3.4 Stabilize Footage

The stabilization of the footage is trivial once the previous sections have been completed. All that there is left to do is to warp every frame by the appropriate amount calculated in the previous section.

Chapter 4

Feature Tracking

As stated in the previous chapter for the successful tracking of features there must be methods to first find features and then to compare two features. For this reason it is easy to break this section into two subsections, in which the first finds features and the second re-finds features.

4.1 Finding Features

A feature can be described as an area or window of an image, typically 7 x 7 pixels. This window generally contains an area of high texturedness or corners. It was decided that for this project a corner detector would be used to find features. An example of a feature window, detected using corners, can be seen in Figure 8.3 on page 33.

4.1.1 Corner Detector

After an extensive period of research into corner detection, a corner detector was found in the Intel® Open Source Computer Vision Library (Int, 2000b). The testing and integration into TIPS of the methods required a significant amount of time. It was necessary to read the source code to understand the operating of the functions. Changes also had to be made to TIPS to accommodate floating-point images.

The function *cvGoodFeaturesToTrack* locates corners, with big eigenvalues, in an image. The function first calculates the minimal eigenvalue

for every pixel of an image and then performs non-maxima suppression. Only local maxima in 3 x 3 neighbourhood remain. The next step is rejecting the corners with a minimal eigenvalue less than *quality_level* time's *max_of_min_eigen_vals*.

The *quality_level* is a threshold parameter passed to the function and *max_of_min_eigen_vals* is the maximum value of all the minimum eigenvalues. It should also be noted that the *quality_level* must be less than one. During the course of this project it was found that a value of 0.6 produces the best results when looking for initial features.

Finally, the function ensures that all the corners found are distanced enough from one another by getting the two strongest features and checking that the distance between the points is satisfactory , i.e. greater than a threshold value again passed as a parameter to the function. If a point is not satisfactory it is rejected. The minimum distance threshold was found to perform the best results with a value of fifteen.

The corners that are successfully found are returned in a pointer to an array, the number of corners found is also returned. An example of features found by this method can be seen in figure 8.1 on page 32.

4.1.2 Find Initial Features

The first frame is passed during the initialization stage of the algorithm. It is in this frame that the initial features are found. Features initially found close to the frame's edges are often lost due to going out of bounds. As a result a region of interest (Int, 2000a) is set up on this image to act as a border, disallowing features to be found close to the edge of the image.

Image Regions of Interest

A very important concept in the Intel® Image Processing Library (Int, 2000a) architecture is an image's region of interest (ROI). All image processing functions can operate not only on entire images but also on image regions. The crucial aspect of ROIs is that they allow for only certain sections of the image to be processed by the Intel functions.

With the border set up it is just a matter of calling the *cvGoodFeaturesToTrack* function. The results of the function are saved in a feature structure to be used in the next stage, re-finding features. The feature structure is used to save information about the features, such as their locations in every frame and whether the feature is valid or not.

4.2 Re-Find Features

Once the initial features have been found it is possible to run through the rest of the video sequence tracking the features. To be able to perform this tasking it is imperative that the features in the previous frame be found in the present frame. Assume for the moment that a feature in the previous frame (f_0) has been found successfully and this location is saved in a feature structure. Now it is desired that the feature be found in the present frame (f_1) see Figure 4.1. A search

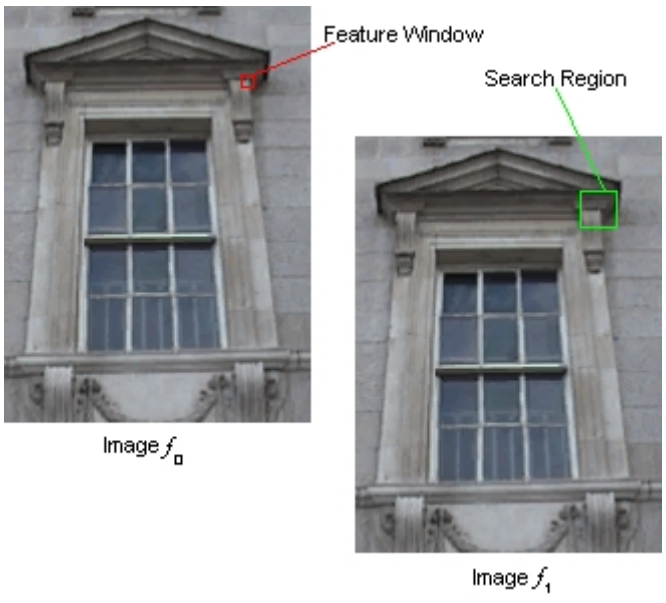

Figure 4.1: Feature tracking

region is set up in the present frame (f_1), for this project a region of

20 x 20 was chosen. The search region was arranged using a ROI and then the *cvGoodFeaturesToTrack* function was used on this area to find all the possible features. All the possible features that are found are saved to be compared with the feature to be found in frame (f_0). For this reason it was necessary to have a copy of the previous frame saved.

Template matching is used to do the comparison. The form of template matching used is a normalized cross-correlation implemented in IPL.

4.2.1 Normalized Cross-Correlation

The cross-correlation values are image similarity measures. The higher cross-correlation is at a particular pixel, the more similarity their is between the template and the image in the neighbourhood of the pixel.

For every pixel in any possible feature, the function *iplNormCross-Corr* computes the normalized cross-correlation value with the feature in (f_0), and stores the computed value in the corresponding pixel of a result image. The template matching is always computed from the image pixel at the geometric centre of the template. So for a 7 x 7 window it is just a matter of looking at the pixel value stored in location row three column three of the resultant image. This value is a percentage represented between 0 and 255, with 255 representing a perfect match.

With this percentage it is feasible to choose the most likely feature out of all the possible features. In addition, a threshold can be employed to ensure that the feature found is within certain constraints. This will be discussed further in the following Chapter.

Now that there are features found in the first frame and it is possible to track a feature from one frame to the next it is realistic to expect that the features can be tracked throughout the whole video sequence. The next step is to discard any bad features.

Chapter 5

Feature Validation

An important part of tracking the features is to ensure that the features are valid. There are a few reasons, mentioned in Chapter 3.2 on page 16, that a feature may become invalid. Some of these reasons are discussed below.

5.1 False Features

False features are virtual features that do not exist in the real world and are created by distortion or optical illusions. Their motions are not reliable and can not be used to calculate the motion of the camera.

5.1.1 Examples

Features could become occluded, as seen in Figure 5.1 on the next page. In frame *b*) an occluding object enters the scene which causing the loss of some of the features from frame *a*). The circles in frame *b*) represent the occluded features.

Perspective distortions can also cause problems. They come about when for example a horizontal edge and a vertical edge at different depths meet. They never actually meet in the real word but appear to meet when viewed from the camera. These virtual features tend to move in an unrealistic manner and have to be detected so as not to have an effect on the calculation of the camera motion.

Specular reflection and shadows are all examples of false features

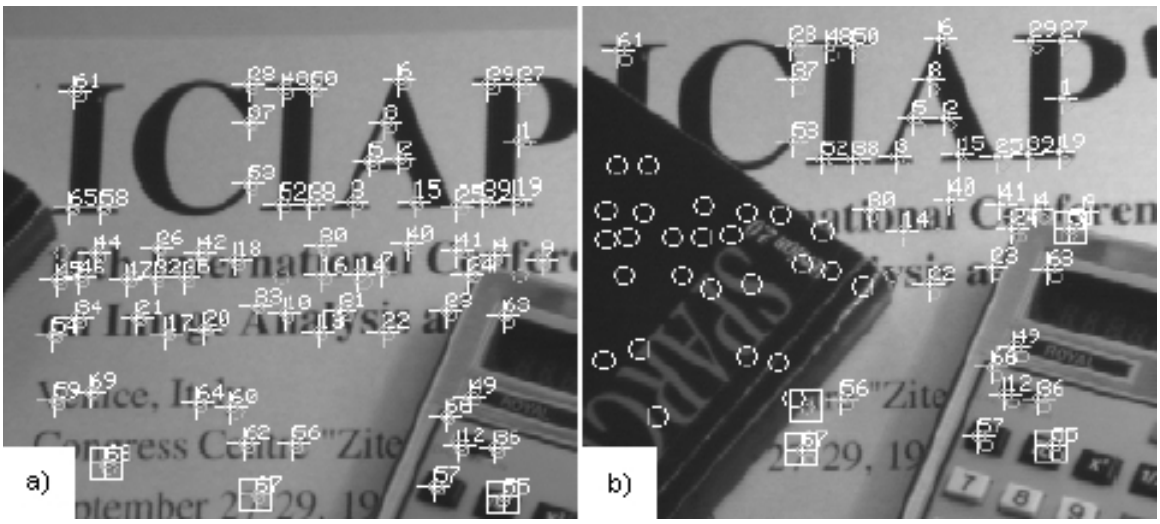

Figure 5.1: Occlusion
From (Censi et al., 1999)

which are created by lighting conditions. Due to the movement of a light source relative to the camera, these features have the same problem as the perspective distortions; they do not move in a realistic manner. Features created by shadows may even disappear when the intensity of a light source changes.

For all these reasons it is important that the false feature are identified and set invalid so that they are not to be used in future tracking and calculations.

5.1.2 Effects

Given all the examples of false features it is desired the their effects be understood, in order to identify them. The main effect that can be noticed is that false features move unlike the majority of the features. There is a possibility that the features will disappear due to occlusion or shadows fading. Another risk of fading shadows is that a feature will be unnoticeable as the original feature which was found. The specular reflection could result in features flickering or moving rapidly in a localized area.

5.1.3 Solutions

It was established that when many features are used the percentage of false features can be kept to a minimum.

If a feature fails to be re-found, i.e. if there are no possible features found by *cvGoodFeaturesToTrack*, it should be set as invalid as it is most likely occluded or has just disappeared.

It is also important that there be a threshold on the comparison between the features in successive frames. This is to ensure that a feature is reasonably like it was in the previous frame, i.e. that it is the same feature. A threshold was set at eighty-five percent. This threshold will take care of features that change as to be unnoticeable.

To remove features which move unlike the majority of the features it is first essential to determine the mean movement of the all the features. This is accomplished by summing the displacement of all the valid features and dividing the result by the number of valid features. The displacement of a feature is the difference between the present frame location and the first frame location. With the mean known, the standard deviation can be determined and with these two values a feature that moves outside a certain times the standard deviation of the mean can be regarded as invalid. A value of 3.5 times the standard deviation was used for all the test AVIs on the accompanying CD.

5.1.4 Other Possibilities

It must also be taken into account that features may go out of bounds, i.e. that they go off the side of a frame. These features should also be set as invalid to stop other possible features from being mistaken as them and subsequently incorrectly tracked.

5.2 Invalid Features

It is important to note that a feature that becomes invalid, for whatever reason, may not be totally redundant. Up until the frame at which it became invalid, it was obviously successfully tracked. For this reason

the feature structure mentioned in Chapter 4 saves at what frame the feature becomes invalid. This allows the features valid information to be used in the calculation of the jitter, discussed in the next chapter.

Chapter 6

Jitter

The unwanted motion or jitter can be described as the difference between the original camera motion and the desired camera motion. Therefore an estimation of these motions will have to be made.

6.1 Original Camera Motion

With the ability to track features successfully through a video sequence and an effective method of disregarding unacceptable features it is feasible to estimate the original camera motion. This is achieved by calculating the mean movement of the valid features, similar to calculating the mean movement of all features in Chapter 5.

6.2 Desired Camera Motion

To estimate the desired camera motion there are a few more possibilities. A smoothing of the original camera motion could be undertaken or (as decided for this project) an interpolation can be made. There is still the question of what sort of interpolation to use. The data estimated for the original camera motion has an x and y displacement value for every frame. Knowing this it would be practical to use a least squared method to do the interpolation, as seen in Appendix B.

However it is desired that a panning functionality be incorporated. This would require dividing the video sequence into several overlapping sub-sequences and then applying the algorithm outlined so far on each.

And to ensure that this appears seamless the first and last frame of every section would have to remain in the same location. This being the case a linear interpolation was chosen to interpolate between the first and last frame. Figure 6.1 shows a graphical representation of the sort of interpolation made on video sequence eleven of the accompanying CD. The red line represents the displacement in the y-direction and the blue line the interpolation of the desired motion.

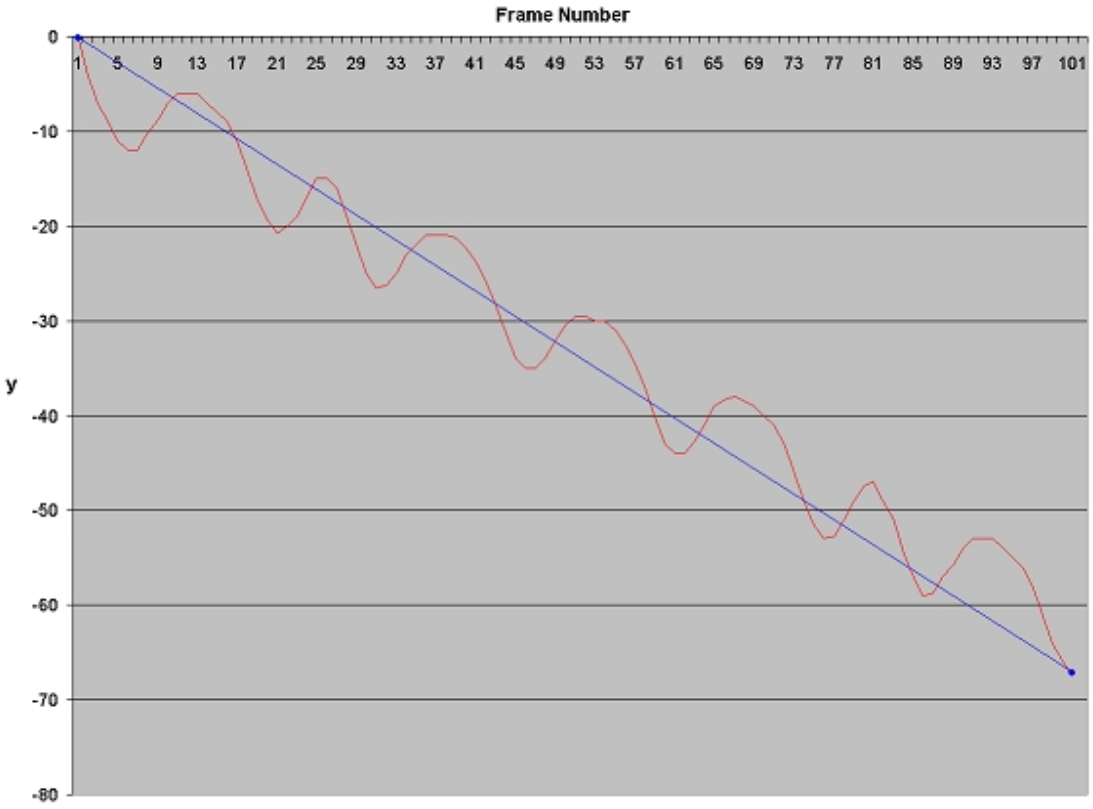

Figure 6.1: The red line is the y displacement and the blue line is its linear interpolation, from the eleventh video

With a decision on the interpolation made it is just a matter calculating the value of the interpolation for every frame. The desired camera movement in x-direction and y-direction should satisfy the following equations:

(6.1) For x-direction: $x = m_1 f + c_1$

$$\text{For y-direction: } y = m_2 f + c_2 \tag{6.2}$$

where m_1, m_2, c_1, c_2 are constants and f is the frame number.

Now it is just a matter of finding the values of m_1, m_2, c_1 and c_2. m_1 and m_2 are the slopes of the respective linear interpolations. The slope can be calculated by finding the average overall displacement of all the valid features and dividing it by the number of frames. c_1 and c_2 are the locations where the linear interpolations cross the y-axis and are always zero, note Figure 6.1. With all the constants known, it is simply a matter of plugging into equations 6.1 and 6.2 to find the desired camera displacement for a given frame.

6.3 Calculating Jitter

Now with the ability to calculate the original and desired camera displacements it is possible to make an estimation of the jitter for any frame. The estimate is merely the difference of the two displacements, given a certain frame number. An example of the jitter calculated for frame 30 of the eleventh video can be seen in Figure 6.2 on the next page.

The unwanted motion for every frame of a video sequence can be calculated and saved for further process in the next stage, which is stabilization.

Figure 6.2: The solid green line segment represents the jitter of frame 30

Chapter 7

Stabilization

Now that there is an estimate for the unwanted displacement of every frame it is just a matter of warping each back to result in a stable video sequence.

As each frame translation has been calculated from the first pass of the video sequence now it is necessary to have a second pass to apply the warping of the individual frames.

To perform the warping *iplWarpAffine* was used. The affine transformation is made according to the following formulas:

$$x' = \text{coeffs}[0][0] \cdot x + \text{coeffs}[0][1] \cdot y + \text{coeffs}[0][2]$$

$$y' = \text{coeffs}[1][0] \cdot x + \text{coeffs}[1][1] \cdot y + \text{coeffs}[1][2]$$

where x and y denote the original pixel coordinates; x' and y' denote the pixel coordinates in the transformed image.

The important insertions into the formula are coeffs[0][2] which represents the vertical unwanted displacement and coeffs[1][2] which represents the horizontal.

Due to the movement of the frames, there will be sections of each frame which do not have any information, (pixel values). Video seven on the accompanying CD is an example of this. The undefined areas are filled with black. For all the remaining video sequences a mosaicking effect was used. This is achieved by filling the undefined areas with pixel values from previous frames which have similar content.

Chapter 8

Evaluation

An evaluation of the project is undertaken in this chapter. The results of the algorithm to stabilize a video sequence are initially presented. This is followed by the areas of the project that were successful and difficulties that were encountered are subsequently discussed. As a final point there is a section on the possible further development of the project.

8.1 Results

This section presents experimental results obtained from both synthesised and real image sequences. The accompanying CD to this report includes the video sequences that were used to test the algorithm. There is also a supplement stabilized video for each unstable sequence.

8.1.1 Feature Selection

Figure 8.1 on the following page and Figure 8.2 on the next page show how the corner detector finds features. Figure 8.3 on page 33 is a feature window taken from the top right-hand side of Figure 8.4 on page 33. It shows the type of features that are found by the corner detector.

8.1.2 Tracking

Figure 8.4 on page 33 and Figure 8.5 on page 34 show how the features are discarded through an AVI. In Figure 8.5 it is obvious to see that

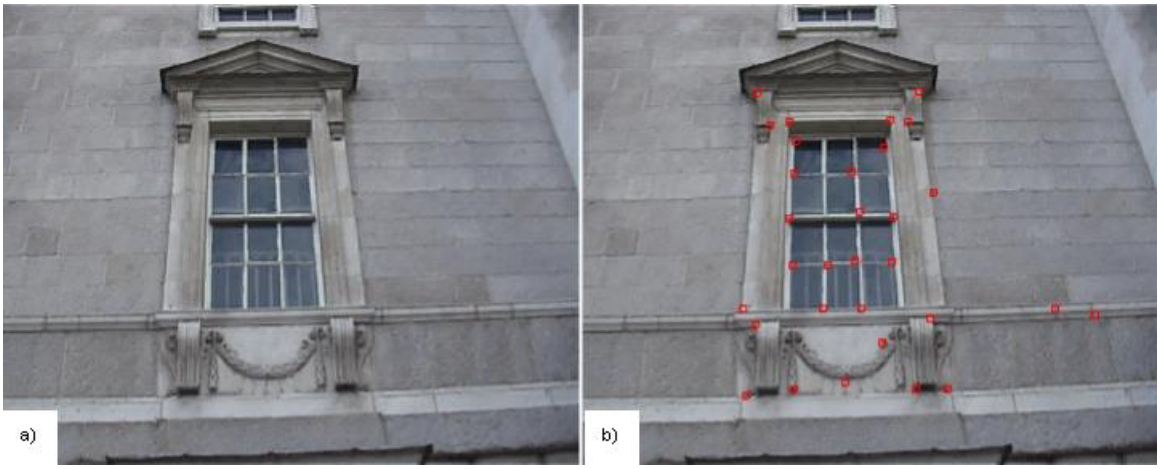

Figure 8.1: Frame from first video a) and with the superimposed features b)

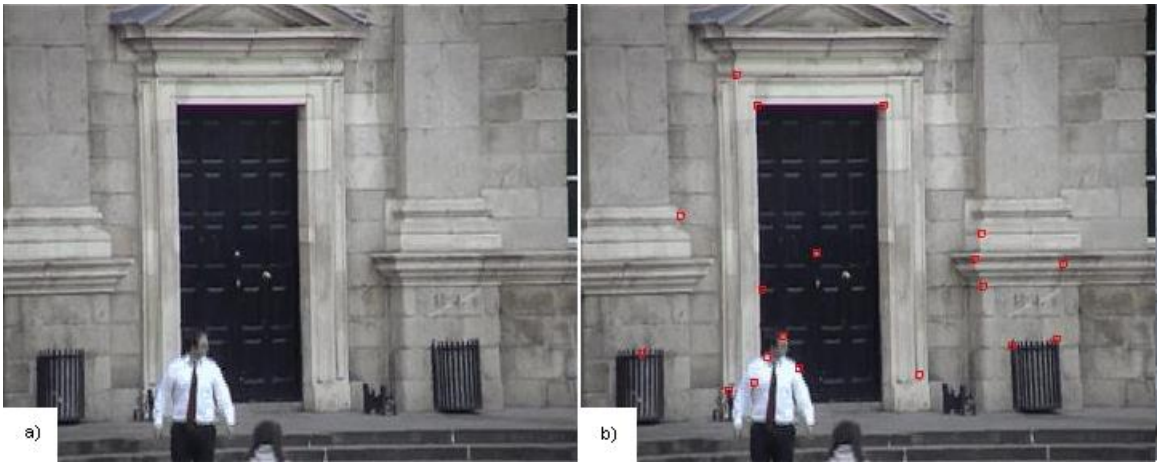

Figure 8.2: Frame from second video a) and with the superimposed features b)

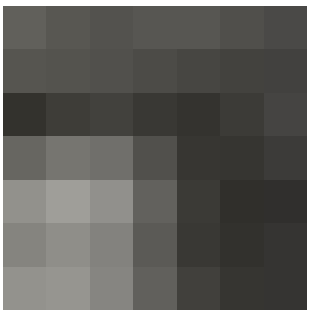

Figure 8.3: Feature window, from top right of Figure 8.1

movement of the man, out of the scene, is responsible for five of the features being lost.

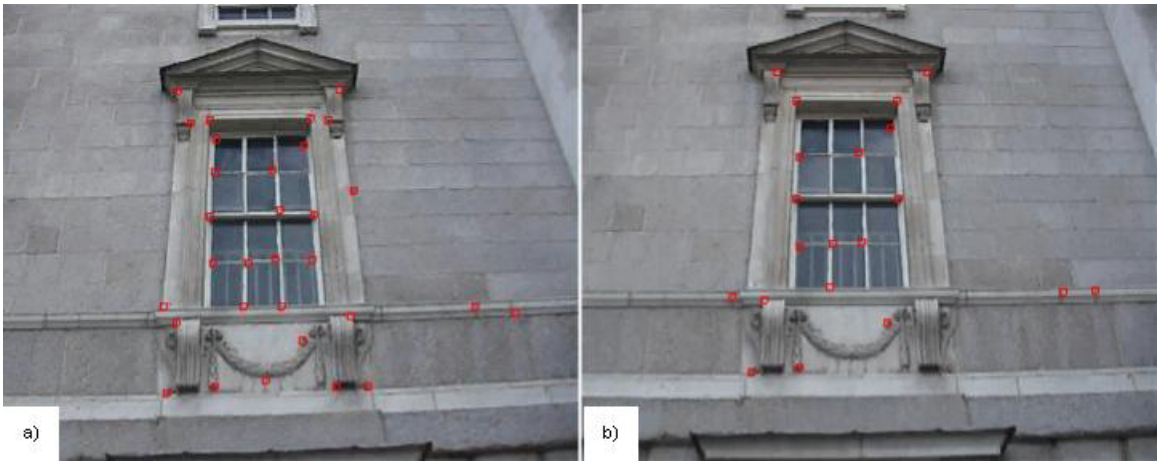

Figure 8.4: First frame from first video with 31 features superimposed a) and last frame with the remaining 20 features b)

8.2 Successful Aspects

Firstly it should be noted that the goal of the project, to stabilize unsteady video sequences from a hand held camcorder, has been accomplished. This is as result of the achievement of each section of this project. Each of the four main areas of this project worked indepen-

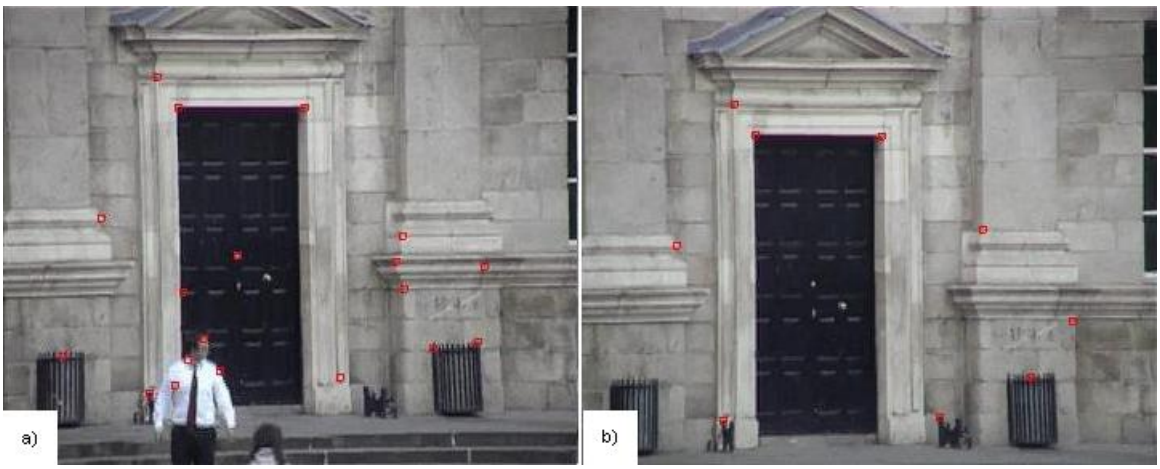

Figure 8.5: First frame from second video with 19 features superimposed a) and last frame with the remaining 9 features b)

dently, operating and calculating results which would be used by the next stage. All stages were effectively tied together to perform the stabilization of any video sequence.

8.3 Difficulties Encountered

Throughout the development of this project various problems have been encountered. However, almost all have been successfully tackled.

8.3.1 Intel Manuals

Methods from both the Intel® Image Processing Library (IPL) Manual and the Intel® OpenCV Reference Manual were utilized. The executing of these methods proved tricky as there was often very little description of how they should be operated. It was necessary to read the source code, which is extensive, to get a better understanding of the functions and their requirements.

8.3.2 TIPS

It was necessary during the course of this project to make changes to the TIPS. As in the case of the Intel® manuals there was a substantial amount of code and to make the required changes to it a lot of this had to be read and understood.

One such example is as follows: It was noticed that when an AVI was finished running that the last frame was continually passed to the stabilizing algorithm. This was found to be undesirable so alterations had to be made to the code to prevent this from occurring. The solution of this problem was assisted by *Darren Caulfield*, a postgraduate at TCD.

8.3.3 Magic Numbers

There were certain sections of the project that required magic numbers. These are numbers that have to be specified, for use as a threshold. Two such magic numbers were required in the *cvGoodFeaturesToTrack* function. The first was used to ensure a minimum distance between two possible features and the second to specify the quality required of the features to be found. Substantial testing was required to find values that would work best across all the possible movie sequences. Other types of these numbers can be seen used in the threshold of the template matching and the number of times the standard deviation used in feature validation.

8.4 Future Work

There are certain sections of this project that could be improved by the addition of new methods. There is also the possibility for changes to be made to existing methods. This section discusses some of the possibilities.

8.4.1 Finding Features

This project used a corner detector to find features although this was adequate and worked well it may be desirable to find additional type of feature to make the tracking more robust. For this reason it is suggested that an additional method of finding highly textured areas could be implemented. Areas which have an altering texture pattern are predominantly unique in an image. It should also be noted that areas of uniform or linear intensity are often common and not unique. It is advised that an implementation of (Tomasi & Kanade, 1991) feature selection method be used. An outline of this can be seen in Appendix A.

8.4.2 Rotation

In this project the removal of unwanted translations was successfully accomplished. Due to time constraints it was not feasible to implement the removal of the unwanted rotational effects. Although there was substantial investigation into the mathematics and it was discovered that the best approach would be an implementation of the Kanade-Lucas-Tomasi Tracking equation (Lucas & Kanade, 1981) and (Shi & Tomasi, 1994).

8.4.3 Interpolation

To make an estimation of the desired camera motion an interpolation was made between the fist and last frame. A linear interpolation was used for the evaluation but if panning was not an issue it is recommended that a least squares method be utilized (see Appendix B).

Chapter 9

Conclusion

Handheld camcorders can suffer from image jitter, as a result of an unsteady hand, moving vibrations and more. And with modern high powered zoom cameras this can be greatly amplified. This unwanted motion (jitter) affects the aesthetic look and feel of home videos and could be greatly improved by stabilizing the video sequence. As a result there is a need for image stabilization. This project has successfully managed to remove the unwanted motions that were caused by translational instability.

In order to stabilize a video sequence the camera motion first had to be calculated. This was achieved with the use of feature tracking. From the first frame of a video sequence features were found, they were then tracked through the remaining sequence. After dropping invalid features the location information of remaining features was used to estimate the average movement of the camera.

The unwanted motion or jitter was determined by subtracting the calculated camera motion from an interpolated version of the desired camera motion. This interpolation was achieved using a linear translation between the first and last frame. Alternative methods for the interpolation could have been used but for the purpose of panning it was desired that the first and last frame remain in the same location as the original. (see Appendix B)

With the unwanted motion known, it was just a matter of warping the each frame of the video sequence by the corresponding identified jitter.

As a consequence of warping there were possible undefined areas in every frame. Mosaicking was used in order to reconstruct these areas by means of adding previous frames which have similar content when accurately aligned.

Bibliography

Buehler, C., Bosse, M., & McMillan, L. (2001). NON-METRIC IMAGE-BASED RENDERING FOR VIDEO STABILIZATION. In *Computer Vision and Pattern Recognition*.

Censi, A., Fusiello, A., & Roberto, V. (1999). IMAGE STABILIZATION BY FEATURES TRACKING. In *the 10th International Conference on Image Analysis and Processing*, pp. 665–667.

Gelb, A. (1974). *Applied Optimal Estimation*. The MIT Press, Cambridge, Massachusetts.

Hampel, F., Ronchetti, E., Rousseeuw, P., & Stahel, W. (1985). *Robust Statistics, The Approach Based on Influence Functions* (1. edition). Wiley Series in Probability and Mathematical Statistic. John Wiley and Sons, New York.

Intel (2000a). *Image Processing Library*.

Intel (2000b). *Open Source Computer Vision Library*.

Irani, M., Rousso, B., & Peleg, S. (1994). RECOVERY OF EGO-MOTION USING IMAGE STABILIZATION. In *Computer Vision and Pattern Recognition*, pp. 454–460.

Litvin, A., Konrad, J., & Karl, W. (1993). PROBABILISTIC VIDEO STABILIZATION USING KALMAN FILTERING AND MOSAICKING. In *SPIE Conference on Electronic Imaging*.

Lucas, B. & Kanade, T. (1981). AN ITERATIVE IMAGE REGISTRATION TECHNIQUE WITH AN APPLICATION TO STEREO VISION. In *IJCAI81*, pp. 674–679.

Morimoto, C. & Chellappa, R. (1996). FAST ELECTRONIC DIGITAL IMAGE STABILIZATION. *Proceedings of the 13th International Conference on Pattern Recognition*, *3*, 284–288.

Shi, J. & Tomasi, C. (1994). GOOD FEATURES TO TRACK. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Srinivasan, S. & Chellappa, R. (1997). IMAGE STABILIZATION AND MOSAICKING USING THE OVERLAPPED BASIS OPTICAL FLOW FIELD. *ICIP*, *3*, 356–359.

Tomasi, C. & Kanade, T. (1991). SHAPE AND MOTION FROM IMAGE STREAMS: A FACTORIZATION METHOD PART 3. DETECTION AND TRACKING OF POINT FEATURES. CMU-CS 91-132, School of CS – CMU.

Zhu, Z., Xu, G., Yang, Y., & Jin, J. (1998). CAMERA STABILIZATION BASED ON 2.5D MOTION ESTIMATION AND INERTIAL MOTION FILTERING. In *IVS98*, pp. 329–334.

Appendix A

Feature Selection

This is an algorithm for detecting areas of high texturedness as seen in (Shi & Tomasi, 1994). The algorithm finds good features by examining the minimum eigenvalue of each 2 by 2 gradient matrix.

Image gradient is defined as:

$$g = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \bigtriangledown I \tag{A.1}$$

Now consider the following matrix:

$$gg^T = \begin{bmatrix} \dfrac{\partial I}{\partial x} \\ \dfrac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \dfrac{\partial I}{\partial x} & \dfrac{\partial I}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \tag{A.2}$$

Define the feature window W (typically 7 x 7 pixels). Integrating the matrix (A.2) over the area W and get:

$$Z = \int \int_W \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \Phi(x, y) dx dy \tag{A.3}$$

where Φ(x,y) is a weighting function. Φ(x,y) set to be 1, because it is desired that every point in the feature window W is equally important.

Z is a 2 x 2 matrix and contains texture information on the region W. The image can be divided into different regions with m x n pixels. By

investigating the eigenvalues of Z, it is possible to classify the texture information in the region W. The noise requirement implies that both eigenvalues of Z must be large. Two small eigenvalues mean a roughly constant intensity profile within a window. A large and a small eigenvalue correspond to an unidirectional pattern. Two large eigenvalues can represent corners, salt-and-pepper textures, or any other pattern that can be tracked reliably.

As a consequence, if the two eigenvalues of Z are λ_1 and λ_2, a window is accepted if:

$$min(\lambda_1, \lambda_2) > \lambda \tag{A.4}$$

where λ is a predefined threshold. The value of λ depends on the type of feature, such as corners and highly textured regions, to be tracked.

Appendix B

Least Squares

The method of least squares assumes that the best-fit curve of a given type is the curve that has the minimal sum of the deviations squared (least square error) from a given set of data. Suppose that the data points are

$$(x_1, y_2), (x_2, y_2), ..., (x_n, y_n)$$

where x is the independent variable and y is the dependent variable. The fitting curve f(x) has the deviation (error) d from each data point, i.e.,

$$d_1 = y_1 - f(x_1), d_2 = y_2 - f(x_2), ..., d_n = y_n - f(x_n).$$

According to the method of least squares, the best fitting curve has the property that:

$$\prod = d_1^2 + d_2^2 + ... + d_n^2 = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = min \tag{B.1}$$

It would be advised that a 2nd degree polynomial be used

$$y = ax^2 + bx + c$$

To obtain the least square error in B.1, the unknown coefficients a, b and c must yield zero first derivatives.