BRIDGE MONITORING SYSTEM USING EMBEDDED COMPUTER VISION

by

KENAN MODI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2011

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to Prof. Roger Walker for his, supervision constant guidance, motivation and encouragement. His patience and invaluable suggestion helped me to successfully complete this research work.

I sincerely thank my committee members Dr. Michael Manry and Dr. Jonathan Bredow for their timely guidance during the course work.

I would also like to thanks my lab mates Akshay Joshi and Digant Shah for their help in the lab. I would like to thanks my friends for their moral support.

Very special thanks to my Mother, Father and Brother for having confidence in me, and for their unconditional love and support.

I dedicate this thesis work to my family.

July 18, 2011

ABSTRACT

BRIDGE MONITORING SYSTEM USING EMBEDDED COMPUTER VISION

Kenan Modi, M.S.

The University of Texas at Arlington, 2011

Supervising Professor: Roger Walker

The bridges and overpass are integral part of road network. These structures deteriorate the moment the construction is over. The factors of deterioration are impact loads of vehicles, natural factors such as wind, rain, earthquake, floods, etc. These deterioration leads to structural failures. In order to prevent failures these structure should be periodically monitored and maintained. In this research project a novel method for early detection of bridge failure using computer vision application and 3D road profiling system is proposed.

The objective of this research work to check the feasibility of this bridge monitoring system for any structural movement detection. Several computer vision techniques is studied and implemented for the process of bridge movement detection.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Objective

Bridges and overpasses are an integral part of the road network. But due to its extensive use over time, the surface and sides of the bridge have produced a number of different levels of deterioration in the form of deformation and cracks. The reason for these deteriorations is various external loads such as traffic, earthquakes, gusts, and possible wave loads during their lifetime. By periodically monitoring the characteristics of these structures, changes or movements in the structures that could result in their failures could be noted. [1] Monitoring structural changes can provide engineers with additional information in the maintenance of these bridges. Bridge engineers need a reliable way to assess the structural integrity of bridges to maintain the continuous operation of the road network while ensuring the safety of the public.

Presently, the main elements of bridge health monitoring are: vibration, torsion movement, strain gauges, accelerometer, etc. These sensors often provide a temporal signature of the vehicles passing over them that could be used to extract the weight and the effect of the vehicles on the structure with good accuracy. To interpret the structural movements over time a diligent statistical method needs to be used. Because of the huge amounts of data that are generated by static bridge monitoring systems, the manual inspection of such data by humans, of all the bridges and overpass, is inefficient and often impossible. Computer vision based automated data processing systems can provide a method for early detection and classification of different types of bridge and pavement deterioration. [2]

In this research project, a system based on sensors such as lasers, accelerometers, gyroscopes and cameras are integrated. There are three major components in this project.

1. Surface profiling system, for physical monitoring

   Used for road profiling of highways and roads in Texas by Texas Department of Transportation

2. Control System for Camera Platform

   Used for removing jitter in camera, in real time

3. Video capture and processing module for structural movement measurement

Computer vision and image processing algorithms are implemented to detect any changes in bridge structure.

The research done here is focused on using a video capture and post processing module for structural movement detection. For this, the following procedure is implemented

1. Compensation for any rotational and /or translation using 3-D road profile as reference
2. Image registration and matching
3. Using Least Mean Square Error (LMSE) to detect any structural changes

To get the best estimate of the bridge structures, any rotation is compensated, using reference to the road surface. By knowing the 3D of the road profile we can make the camera platform line up with the road surface in real time. Because the bridge's images are captured in different times, locations, and angles, we must register them before comparing them to measure the structures' parameters. In our project, corner point detection and shift invariant factors are used for image matching and registration. Corner point detection is an approach used within computer vision systems to extract certain kinds of features and infer the contents of an image. After matching two results there is the need to find the rotation and translation between those two images due to shift in camera position, for this homography is used to estimate camera pose. Then finally, LMSE is used to detect any changes in bridge structure.

The next chapter describes the background work done. It includes literature review, computer vision introduction and introduction to profiling algorithm and 3D road profile. Chapter 3 describes the embedded system consideration and implementation for this project. Chapter 4

describes the implementation and result of the project undertaken. Chapter 5 includes the conclusion.

CHAPTER 2

BACKGROUND

2.1 Introduction

In this chapter we will discuss some the most important computer vision and image processing fundamentals. This chapter hold key to implementation of this project

2.2 Camera Geometry

2.2.1 *Pinhole Camera Model*

A camera maps the 3D world (object space) into 2D image. Here we start with the most specialized and simplest camera model which is a basic pinhole camera (Figure 2.1). [3] As seen in figure 2.1 the object in front of the camera is projected upside-down onto the screen.

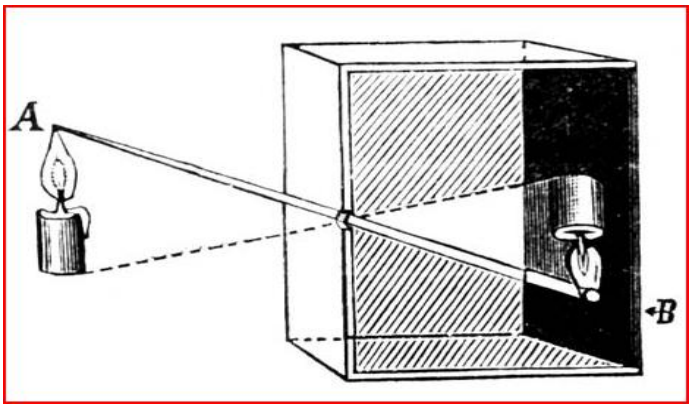

Figure 2.1 Pinhole Camera Model

Assume a projection of points in space onto a plane. Let O_c be the center of projection of the Euclidean co-ordinate system, and plane z=f, which is called *focal plane.* Let X = (x, y, z)^T be a point in space mapped on the image plane where a line joining the points X to the center of projection meets the image plane as shown in figure 2.2. Given the point in space we can get the corresponding point in image plane. By using similarities of triangle in figure 2.3 we can write

$$\frac{x}{f} = \frac{Xc}{Zc} \qquad (2.1)$$

Hence for 3D point we can write x = f $\frac{Xc}{Zc}$, and y = f$\frac{Xc}{Zc}$ (2.2)

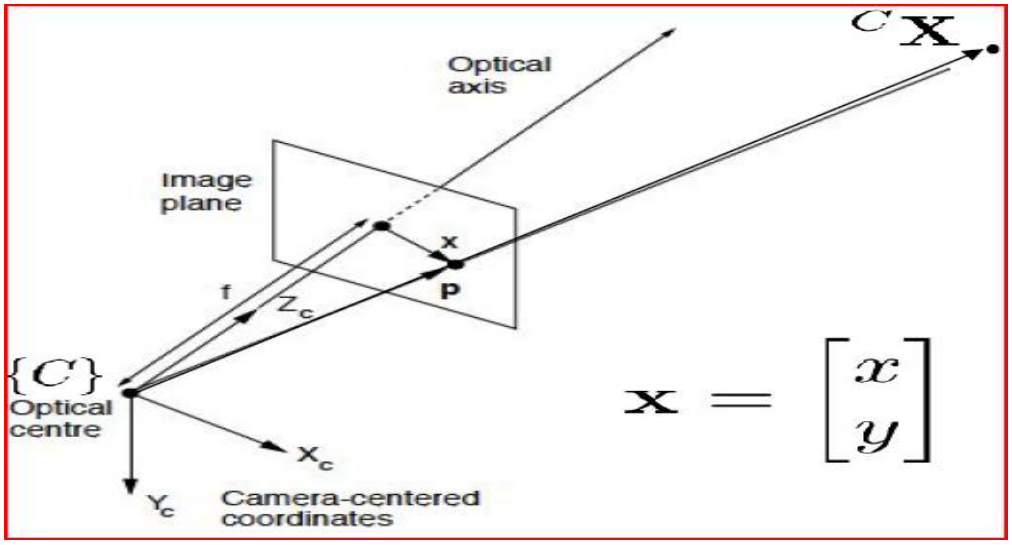

Figure 2.2 Pinhole Camera Geometry

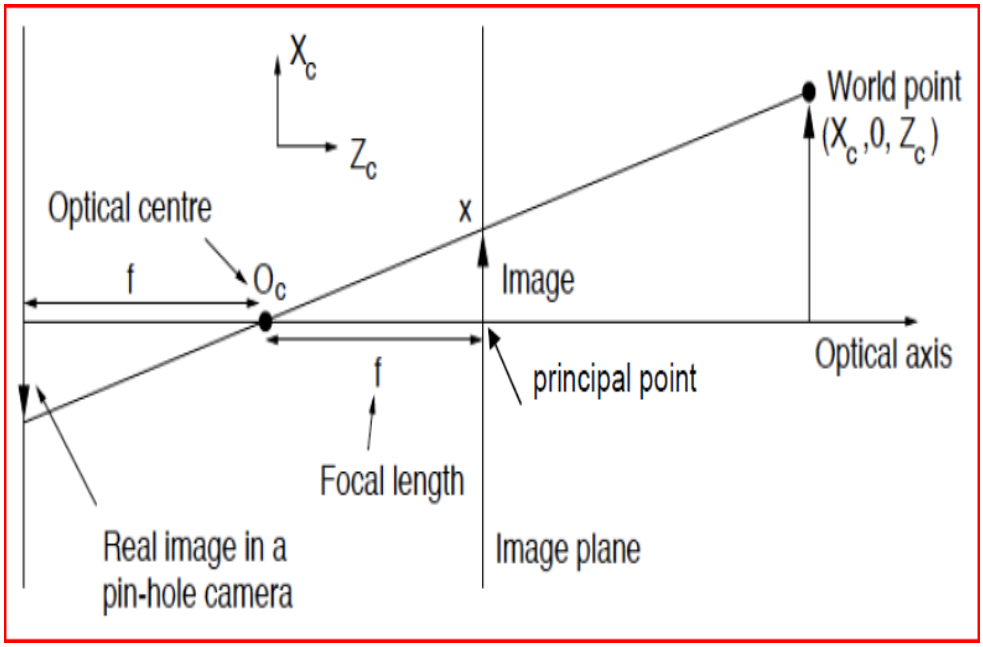

Figure 2.3 Pinhole Camera Model 2D

By expressing the above expression in vector form we can write as

$$X \triangleq \frac{x}{y} = \frac{f}{Zc} \frac{Xc}{Zc} \quad (2.3)$$

In expression (2.1), we assumed that the origin of coordinates in image plane is at principal point. But in practice, it may not be true, so in general mapping we can write

$$(X, Y, Z)^T = (f\,X/Z + p_x,\ f\,Y/Z + p_y) \qquad (2.4)$$

We define pixel coordinates $\mathbf{u} = [\,u \quad v]^T$, hence in matrix form it can be written as

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u0 \\ v0 \end{bmatrix} + \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (2.5)$$

where *k, u0, v0* are the intrinsic camera parameters.

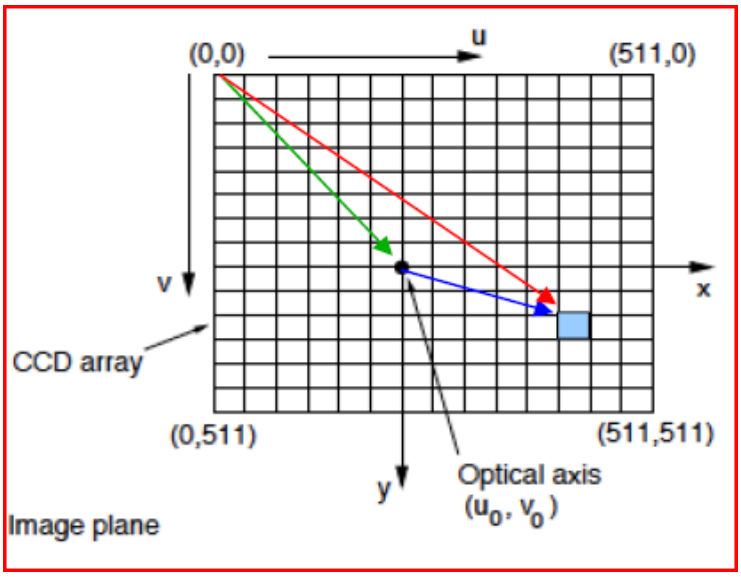

Figure 2.4 Image Plane in Pixel Form

So we can extend the equation 2.5 in Homogeneous form as given below

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k & 0 & u0 \\ 0 & k & v0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (2.6)$$

Equation 2.6 gives the homogeneous representation of the coordinate system. When applying a pinhole camera model and homogeneous coordinate system we get

$$S\tilde{\mathbf{u}} = K\pi_0\ {}^cX \qquad (2.7)$$

where K is the Intrinsic camera parameters matrix , π_0 is the ideal projection matrix.

An extensive introduction to the Pinhole camera model is supplied in [3]

2.2.2 *Epipolar Geometry*

The projective geometry between two views is called *Epipolar Geometry.* [3] It depends on cameras' intrinsic parameters and relative pose and not on scene structure. The fundamental matrix summarizes these parameters. Fundamental matrix is discussed in brief later in the chapter.

Epipolar geometry is used to extract 3D structure from a pair of images. There are two methods for this. In the first and classical method, known as the *calibrated route*, we first need to calibrate both cameras (and viewpoints) with respect to some world coordinate system, calculate the so-called *epipolar* geometry by extracting the *essential* matrix of the system, and from this compute the three-dimensional Euclidean structure of the imaged scene. This method is widely used for stereo camera system.

The second or *uncalibrated route* corresponds to the way biological systems determine three-dimensional structure from vision. In an uncalibrated system, the *fundamental* matrix is calculated from image correspondences, and this is then used to determine the projective three-dimensional structure of the imaged scene. This method is used in homography, as discussed later. [4]

Figure 2.6 shows the geometric model of a two view system. C and C' are the two optical centers of left and right cameras respectively; M is the object of interest. Here are some definitions regarding Epipolar Geometry

Baseline: It is the line joining the optical center of two cameras'

Epipole: The epipole is the point of intersection (in pixels) of the baseline with the image plane. There are two epipoles, e and e', one for each image.

Epipolar plane: An epipolar plane is the plane passing through the camera centers and each 3-D point.

Epipolar line: An epipolar line is the line of intersection of the epipolar plane with the image plane.

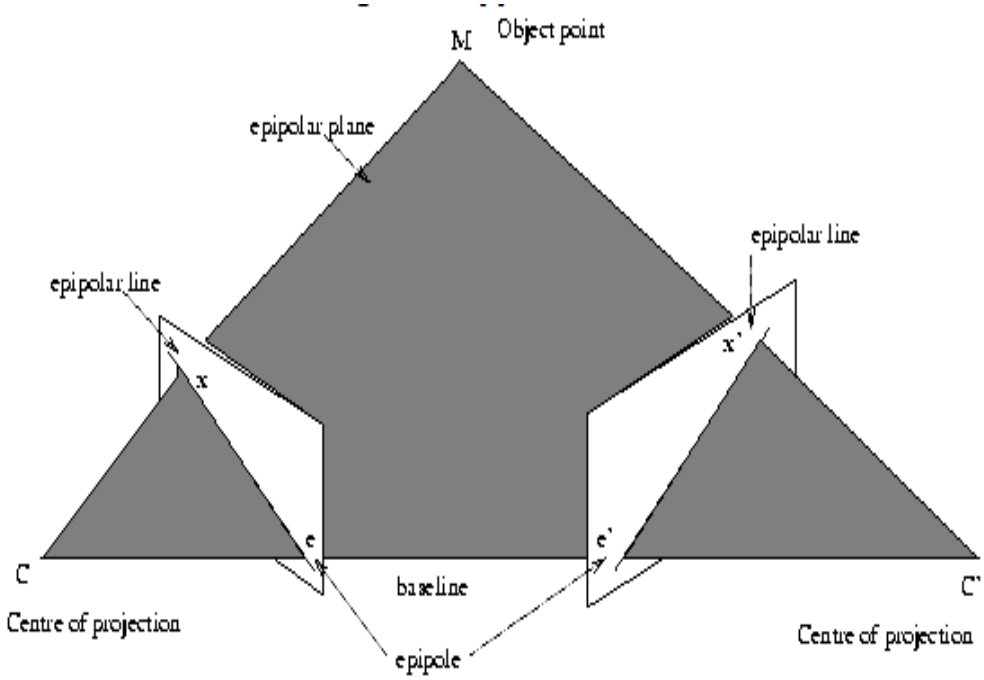

Figure 2.5 Epipolar constraints

These epipolar constraints play a fundamental role in stereo vision and motion analysis.

Epipolar Constraints can be written as a function of the rays in the image as

$$^{c'}\mathbf{X}^{T}\,[^{c'}T] \times {}^{c'}R\,{}^{c}\mathbf{X} = O \qquad (2.8)$$

where T and R are translation and rotation between two cameras as shown in figure 2.7

This constraint implies that, for a given point (ray) ^{c}X in one image corresponding point (ray) $^{c'}X$ in the other camera lies on a epipolar line that has equation

$$L \sim [^{c'}T] \times {}^{c'}R\,{}^{c}X \qquad (2.9)$$

$^{c'}T \times {}^{c'}R$ is called *essential matrix* **E**

Hence we can write equation (2.8) as

$$^{c'}X^{T}\,\mathbf{E}\,^{c}X = O \qquad (2.10)$$

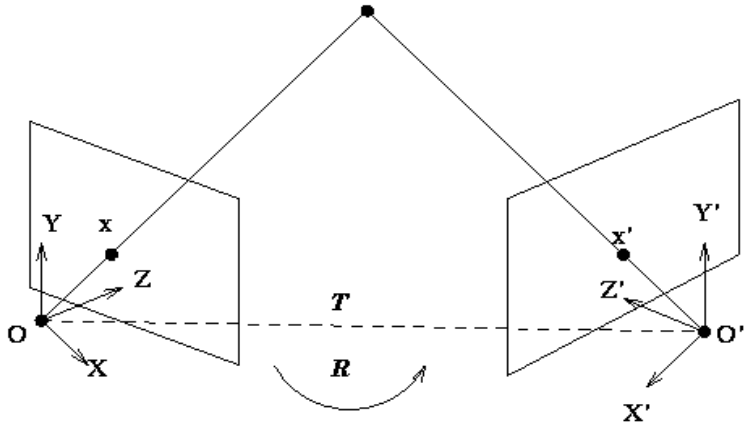

Figure 2.6 Two-view Coordinate System

In the uncalibrated case, we don't know **R** and **T**; all we have are image coordinates $[u \quad v]^{T}$ in the image plane.

We know that $^{c}X = K^{-1}\,\tilde{u}$ and $^{c'}X = K^{-1}\tilde{u}^{T}$, when rewriting the equation (2.10), we get,

$$\tilde{U}'^{T}\,K^{-T}\,E\,K^{-1}\,\tilde{u} = O \qquad (2.11)$$

where F is the *fundamental matrix,* it allows us to define the epipolar in the image plane.

For detailed derivations refer to [3]. Section 9.2.2

Two-camera geometry does not necessarily imply the use of a stereo camera system. Two frames of a single camera at different points in time can also be addressed in this way, which is used in this research to get the rotation and translation between two views.

2.3 Image Features

When analyzing camera images immense amount of information can be gathered. Several computer vision processes require the finding of features or matching points across several frames or views. As seen in the previous section, in stereo imaging we can triangulate

the two features or views to get the information about depth of object, rotation or translation between two view points, etc. In general image features consist of two parts key-point and a descriptor. This section covers these features, how to extract them and how to describe the descriptors of the feature extracted.

An image I(x, y) can be written as a function of many variables such as, object intensity, position of the camera, properties of the camera, and nature and distribution of the light source. Image features are a part of the image and are associated with interesting scene elements via the image formation process. The descriptor is a vector containing important properties of a particular image feature. Before continuing on the features and how to find them we will discuss how to deal with image noise.

2.3.1 *Filters*

Noise is always present in an image. Principal sources of noise in digital images arise during image acquisition and transmissions. This noise has to be eliminated as much as possible without altering the image. The most common technique for noise smoothing is linear filtering.

2.3.1.1 Averaging Filter

This is the most common technique used in spatial domain for noise filtering. It consists of convolving an image I(x,y) with a constant matrix K, called kernel.

$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, all the entries of K have to be non-negative for it to be average smoothing.

One drawback of averaging filter is that it intuitively takes out small variations. A bigger kernel causes more blurring. [5] A nonlinear filter such as a median filter has better performance as compared to an averaging filter as discussed later in this section.

2.3.1.2 Gaussian Kernel

This is a frequency domain filtering technique. The kernel is given as

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \qquad (2.12)$$

Where σ is the standard deviation and is measure of spread of the Gaussian curve.

To remove the noise, we use Gaussian smoothing operator. It is a 2-D convolution operator which blurs n image by removing details. The Gaussian PDF curve is bell shaped centered at its mean.

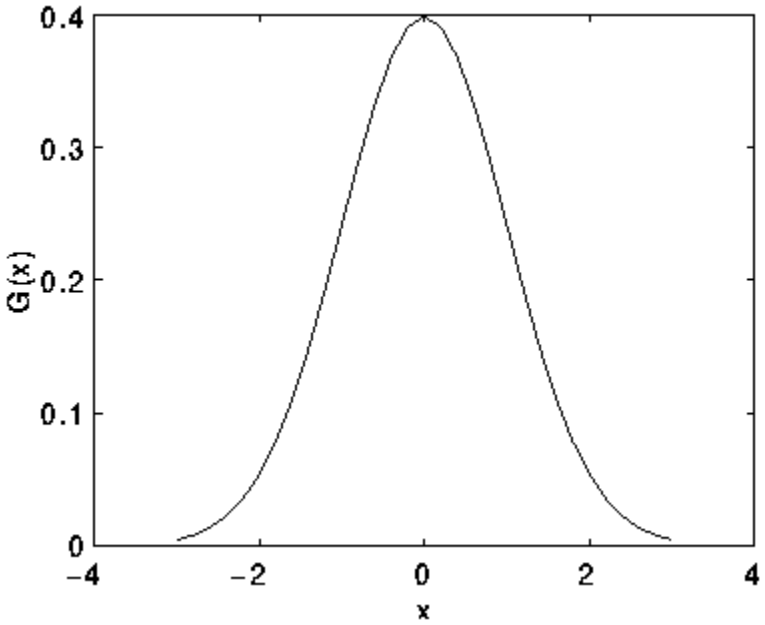

Figure 2.7 Gaussian Distribution (mean =0)

The 2-D distribution of the Gaussian smoothing filter is used as a point spread function. In other words, it is symmetrical about its mean value and it has only one maximum at mean value. The width of density function is directly proportional to the standard deviation, in effect decides the amount of blurring. The maximum value of density function is inversely proportional to the standard deviation σ. In theory, the Gaussian distribution doesn't become zero at any point. For it to go to zero, an infinitely large convolution kernel is required. As we increase the width of the kernel its computational complexity increases. Hence for practical purpose we assume three standard deviations from mean to settle to zero, and so we truncate the kernel at that point. [6]

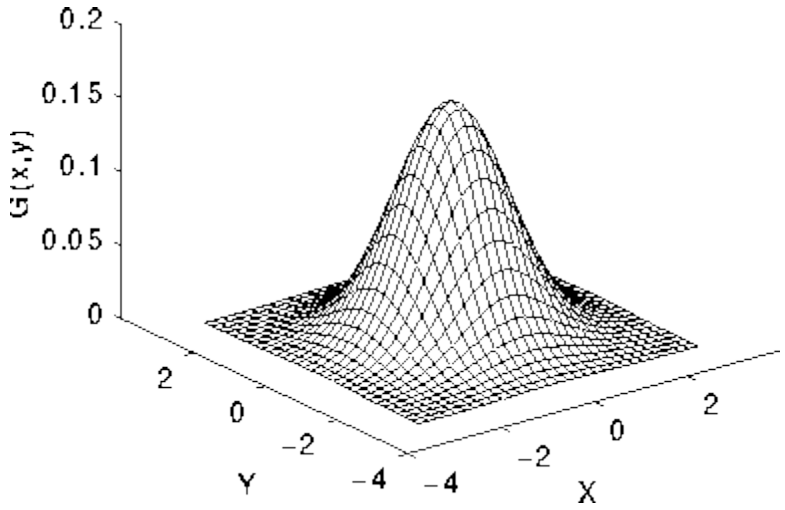

Figure 2.8 Gaussian Distribution in 2-D (mean =0, σ=1)

After selecting a suitable kernel, the Gaussian smoothing can be performed. The Gaussian kernel is circularly symmetric. Hence 2-D convolution can be break up into two parts, first performing 1-D convolution in x- direction, and then 1-D convolution in y- direction. These steps can be performed in parallel using software or using FPGA, to speed up the complex process.

The main advantage of a Gaussian filter over other filters is that there is no ringing effect no matter what the filter order is. In other words most of other filters distort the image but the Gaussian filter does not distort the original image. It has been used in several places in this project which will be discussed later.

2.3.1.3 Median Filters

As stated earlier, an averaging filter removes noise by blurring the image, during the process it also blurs the edges. Median filters have better performance because they don't distort the original image. Their response is based on the ordering or ranking of pixels contained within the kernel. The kernel of a median filter is similar to that of an averaging filter except that the kernel has no values. Below are the steps to perform median filtering:[6]

1) Assume a 3x3 empty mask
2) Place the empty mask at the left hand corner

3) Arrange the 9 pixels in ascending or descending order

4) Choose the median from these nine values

5) Place this median at the center

6) Move the mask over the image

2.3.2 *Edges*

Edges *characterize object boundaries.* They are for segmentation registration, and identification. An edge is the set of connected pixels that form a boundary between two disjointed regions. An edge is an important feature. Several methods along with different operators to find an edge will be discussed in this section. [7]

The process of edge detection is to take the derivation of the image. The basic idea of the derivative approach is to compute the local derivative operator. The first derivative at any point in an image is obtained by using the magnitude of the gradient at that point.

2.3.2.1 Roberts Cross Edge Detector

The Roberts Cross operator computes a 2-D spatial gradient measurement on an image. Thus it highlights the discontinuities of scene intensity which often corresponds to the edges in the image.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Figure 2.9 Roberts Cross Convolution Kernels

Roberts gradient is obtained by using cross difference. The output image is estimate of gradient of the input image. The operator consists of a pair of 2X2 convolution kernels as show in figure 2.9. These two kernels are 90^0 rotated with respect to each other. Hence edges at 45^0 to pixel grid give maximum output. For quicker implementation these kernels can be applied separately on original image for each of perpendicular orientation.

Although the Roberts Edge Detection technique is very quick and simple to evaluate, there are some drawbacks of this detector. First, it is extremely sensitive to noise, since it uses difference method to calculate kernel. Second, its performance depends on width and orientation of edges in an image. Its performance also depends on the intensities in foreground and background. Hence only very sharp images can be detected. [8] [9]

2.3.2.2 Sobel Edge Detector

The Sobel operator, as Roberts's operator, computes a 2-D spatial gradient measurement on an image. The output image is an approximate absolute gradient magnitude of input image for a particular pixel. In this operator, higher weights are assigned to the pixes close to the candidate pixel.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2.10 Sobel Operators

The operator consists of pair of 3X3 convolution kernels as shown in figure 2.10. As seen in Roberts operator, these two kernels are also 90^0 apart. These kernels can be applied separately on an image. First, we convolute the original image with the Gx mask to get the gradient in x- direction, and then we convolve the original image with the Gy mask to get the gradient in y- direction. After getting these to gradients, we combine them to get absolute gradient magnitude.

The advantage of using Sobel mask for edge detection is that it provides a smoothing effect along with providing differentiation. Hence it performs better the Roberts when the image is noisy. Having said that, noise is also a high frequency component, hence derivative filters are very sensitive to noise. [8] [9]

An important property of above discussed mask is that their sums of coefficient are zero. Edges are abrupt discontinuity in grey levels, and hence are high frequency regions. Since sum of the coefficient of all these masks is zero, they eliminate all the low frequency components of the image i.e., when these mask are placed in low frequency region, output is close to zero. Hence these masks give edges without any low frequency regions in the final output image. But there disadvantage is that they are not isotopic, their performance depends on direction of discontinuity. [6]

2.3.2.3 Canny Edge Detector

Canny edge detector is one of the optimal and robust edge detector methods. Canny operator is a first derivative edge detector coupled with noise cleaning. [10]

The Canny edge detector is a multistage algorithm. The steps are as follows:

1) Pre-processing; As discussed earlier edge detectors are prone to noise. A Gaussian smoothing is used for blurring. Usually, a 5x5 Gaussian filter with σ=1.4 is used.
2) Calculating Gradient: In this step gradient magnitude and directions are calculated at every single point in the image. The magnitude of the gradient at a point determines the possibility of it lying on the edge. If gradient is high, it is on the edge. The direction of the gradient shows how that edge is oriented. A simple 2-D derivative gradient such as the Sobel edge detector is used for this purpose.
3) Non Maximum Suppression: After gradient magnitude image is obtained, a pixel which is not maxima is suppressed, i.e., set to zero. A 'thin edge' is obtained by implementing this step.
4) Hysteresis threshold: By implementing the previous step, a pixel that had a gradient magnitude greater than the upper threshold is marked. The tracking process exhibits hysteresis controlled by two thresholds: *T1* and *T2*, with *T1* > *T2*. Tracking can only begin at a point on a ridge higher than *T1*. Tracking then continues in both

directions out from that point until the height of the ridge falls below *T2*. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.

The effect of the Canny operator is determined by three parameters --- the width of the Gaussian kernel used in the smoothing phase, and the upper and lower thresholds used by the tracker. Increasing the width of the Gaussian kernel reduces the detector's sensitivity to noise, at the expense of losing some of the finer detail in the image, also the localization error in the detected edges increases slightly.

To conclude, several methods to find the edge have been discussed. The canny edge detector is implemented in this project to get rid of extra data from the image and for detection of the bridge structure. Although edge is a powerful feature, it can be misleading. For example the motion of an object using edge detection can only be viewed in one direction, i.e., normal to the edge. [7]

2.3.3 *Corners*

A corner is the point of intersection of two or more edges. An abrupt variation in the image gradient direction characterizes a corner. A corner is more reliable and stable feature compare to edge. Unlike in edge, derivative of gradient in corners changes in more than one direction. This helps to write efficient algorithms. Corner features are the most important feature for motion detection, image matching and object recognition. In this project to find the features the Harris corner detector is used.

2.3.3.1 Harris Corner Detector

Harris corner detector:

Harris corner detector can be mathematically represented by the following equation.

$$E(u, v) = \sum_{x,y} w(x,y)\,[I(x+u, y+v)-I(x, y)]^2 \qquad (2.13)$$

where E is the difference between the original window and moved window

u is the windows displacement in x direction

v is the windows displacement in y direction

w (*x, y*) is the window at position (x,y), this is the mask

I (*x, y*) is the intensity of the original image

I (*x+u, y+v*) is the intensity of the moved window

All possible small shifts can be covered by performing an analytic expansion about the shift origin. [11]

On applying the Taylor series expansion on equation 2.13 and simplifying it we get,

$$E(u, v) \approx \sum_{x,y} u^2 I_x^2 + 2uvI_x + v^2 I_y^2 \qquad (2.14)$$

The above equation can be written in matrix form as

$$E(u, v) \approx [u\ v]\ M \begin{bmatrix} u \\ v \end{bmatrix} \qquad (2.15)$$

where $M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

To conclude, the Harris Corner detector is just a mathematical way of determining which window produces large variations when moved in any given direction. The Harris corner detector is fairly simple to compute, and is a fast and robust corner detector operator. The key features of the Harris detector is that it is rotation invariant, though its performance depends on scale, i.e., it is not scale invariant. For image registration and matching it is very important that the corner detector be both scale and rotation invariant. The technique to find corners and edge irrespective of rotation and scale and matching them is discussed in the chapter 4.

CHAPTER 3

SYSTEM CONSIDERATION AND CONFIGURATION

This chapter includes the discussion on the sensors needed, their synchronization and integration, processor consideration for the project and configuration of the system.

3.1 Sensors

All the different sensors used in the system are discussed here.

3.1.1 RoLine Laser

The laser used in the system is the RoLine 11xx family of laser. It is a high speed, high density 3D profile scanning system for road profiling. The laser has the projection of 2.6" – 5.4". Its specifications are:

Input Voltage 48V DC Scanning Rate 3KHz Measurement range 200mm

Output is a 16 bit number in 1/100th mm, sent in form of Ethernet packets. A single scan of laser output gives 198 points. The laser outputs a 3K Hz sync pulse which is used to synchronize it with accelerometer. This is extremely critical for profiling algorithm.

RoLine 11xx family has two modes of operation, free mode, contains the distance value of each of the 198 points and Bridge mode, a filtered average of all the point values to single point value. Both these modes are evaluated to compute longitudinal profile of the road.

3.1.2 Accelerometer

The accelerometer used in the system is a Columbia Research Labs SA-107BHP. It is a single axis servo accelerometer. Specifications are:

Range +/- 4g Resolution 1.876 V /g, Accuracy +/- 0.2%

Input voltage +/- 15V Output voltage range +/- 7.5

A low pass filter is applied to the output of the accelerometer before connecting it to an analog to digital (A/D) converter. The accelerometer is an integral part of the system because it is used to filter out the vehicle movement from the laser reading so as to get an accurate profile of the road.

3.1.3 Gyroscope

The gyroscope taken into consideration is a Watson Industries 3 axis gyroscope. It is designed for instrumenting the drive and handling characteristics of the vehicle. The sensor provides both angular rate and acceleration outputs in analog as well as digital format. This dynamic measurement system features six accelerometer output. This allows measurement of forward and lateral acceleration which are essential for free of gravity influence. It has an RS-232 serial interface, through which the configuration and other operation can be performed. Specifications of the sensor are;

Input voltage: 10VDC to 30VDC, Rate Range: +/- 100^0 /s, Acceleration range: +/-10 g,

Pitch Range: +/- 75^0 Relative heading: 0-360^0 Accuracy: +/- 0.05 deg/ sec

Output: RS-232 serial interface (adjustable baud rate)

Using this sensor the drive characteristic of the test vehicle can be measured. When this measurement is integrated with the road profile, evaluated using a wide line laser, the 3D profile the road along with long wavelengths can be obtained. Also the data obtained from gyroscope is used to stabilize the video capture device platform. This in turn reduces the jitter in the video data caused by vehicle movement.

3.1.4 Distance Encoder and Start Sensor

A distance encoder, attached to the tire of the test vehicle, outputs the pulses per kilometer of distance traveled. The output is calibrated according to the tire size to get the translation distance traveled by the test vehicle. Along with this an infrared sensor is used to

mark the start and end of the test run. The signal from the start sensor is used to activate all the above sensors along with network camera simultaneously.

3.1.5 Video capture module

A high resolution network camera, Canon VB C10, is considered for capturing the video of the structure. The camera output is through the Ethernet.

Resolution range: 640x480 to 320x240 to 160x120 Capture rate: 30 frames/ sec

Input voltage: Adapter (110-240V 50/60 Hz) or 12- 14 VDC Field View: 65^0

16x optical zoom CCD technology External Triggering Available

The Network camera can be triggered externally, which helps to synchronize the video data with other sensor data. It has a Built-in Web Server and FTP Server, compatible with different protocols to integrate easily into any network. It also includes an Ethernet Terminal (10M/100M, auto negotiation), compatible with all the following protocols: TCP/IP, HTTP, BOOTP, FTP, NTP. This solves the portability issue for integrating the camera to the embedded system. Also the frame rate of this camera is programmable. Since the frame rate has a direct relation to the amount of photons captured, it is highly desirable to have a variable frame rate. For example, if the frame rate is 100 Hz, exposure time reduces to 10ms, which in turn limits the amount of photons acquired by the camera. Dark scene requires higher gain at high frame rates, which introduces the noise level. In order to minimize the noise, the frame rate of the camera is adjusted according to ambient light. Also, this camera has auto gain adjust calibration built in.

3.2 Data Synchronization and Integration

Each sensor has a different sampling rate and outputs data in different formats. Hence synchronization of this data is very important. Laser and camera has an Ethernet output, the accelerometer gives an analog output, while the gyroscope gives serial output. The distance encoder and start signal, and sync signal from the laser gives digital pulses. Data translator

board (part no. 9816A) is used to capture data from the accelerometer, distance encoder, start signal and sync signal. Synchronization of these sensors is done in software.

While measuring the road profile it is of utmost importance that the accelerometer and the laser reading are synchronized. The accelerometer data is used to filter out the test vehicle movement from the laser reading to measure the accurate road profile. To measure the 3D road profile gyroscope data is integrated with the surface data. Through the gyroscope we are able to get the long wavelength as well as short wavelength transverse profile. Camera data is synchronized with the data from distance encoder. By doing this, the position of the camera for a particular frame can be noted. This helps in image registration and matching process. Chapter 4 discusses this integration in detail.

3.3 Processor Consideration

The selection of the microprocessor for a given application is probably the most difficult task, at the same time the most critical task in designing any Embedded System. The key to selection of a microprocessor is the knowledge of different processors, suitable for a specific application. For example, if the application is to develop a stand-alone system performing a specific and discrete task then generally a microcontroller or Digital Signal processor is selected, and for general or wide variety of applications a General Purpose Processor is used. So, before discussing the various processor hardware available, we need to make sure that it is suitable for application of this project, which is to generate a 3D road profile system as well as implementing various computer vision algorithms. In this section we describe in brief various processors.

3.3.1 General Purpose Processor

The General Purpose Processor (GPP) is the generic processor, designed for a wide variety of applications. Most of the GPP uses the *von Neumann* architecture; it uses one bus for both data and program memory access. This can become a bottleneck in case of cache coherency. Some of the important features of GPP are; its flexibility and generality, wide

address bus allowing the management of large memory spaces, integrated hardware memory management unit, wide data formats, and integrated co-processor supporting complex numerical operations, such as floating point multiplications.

3.3.2 Digital Signal Processor

The programmable Digital Signal Processor (DSP) is general purpose processor, designed specifically for digital signal processing purposes. They contain a special architecture and instruction set so as to execute computation intensive DSP algorithms more efficiently. In other words, a DSP's instruction set is optimized for matrix operations, particularly multiplication and accumulation (MAC). Most DSP's uses modified Harvard architecture; it uses the DSP's multiport memory that has separate bus system for program memory and data memory. Important features of DSP are: Fixed-point processor or floating point processor, architecture optimized for intensive computation, narrow address bus supporting a only limited amounts of memory, specialized addressing modes to efficiently support signal processing operations.

3.4 Memory Management Consideration

For vision based algorithms, fairly large amount of memory with high bandwidth and small access time is desired. Hence selection of memory in an Embedded Computer Vision application is driven by the space, capacity and access time. For example, multiport SDRAM would be ideal for creating an easy to use shared memory located between DSP and FPGA or GPP and FPGA.[12] Though SDRAM is widely used in an embedded system, SDRAM controller has to be integrated in DSP or system on chip (SoC). Since DSP generally does not have a memory management unit, it can become a bottleneck in memory read/ write operation with large memory size. The solution to this problem can be to avoid the use of DSP in low level read/ write operations.

As stated earlier, the system configuration depends on the application and not just the speed or performance of the processor. Though performance of DSP processor is much superior to GPP for computer vision application, this project is more than just computer vision. It

includes data processing of various sensors from laser to gyroscope. Also generating the road profile is sequential process, DSP might not necessarily increase the performance. Hence for the evaluation purpose GPP processor is used to implement the computer vision algorithms. The possible use of the FPGA along with the GPP, for this particular project is discussed in Chapter 5.

The processor used for this project is an INTEL's CORE 2 Quad. It has two core processors in a single package. Each of cores has 4GB of cache memory integrated with it.

INTEL's OpenCV and MATHWORK's Matlab's (image processing toolbox) were used to implement the algorithms. For the further information on these software tools refer [13] and [14]

CHAPTER 4

IMPLEMENTATION AND RESULTS

The previous chapters have provided details on the essential components of this project and also reviewed several computer vision algorithms. In this chapter we will investigate how those components are integrated and the implementation of the computer vision algorithms. We will also analyze the results of the experiments performed.

The complete process is divided in three parts; 1) Compensation of rotation and /or translation due to camera jitter, 2) Feature extraction, detection and image matching, and 3) Detecting any structural movement.

4.1 Compensating Camera Jitter (Post Processing)

To detect any structural movement in the bridge structure, we compare the two frames of the video taken at different time. The factors that should be considered before comparing two frames are; jitter in camera, separating object motion from camera motion, camera position, scale of image, and illumination. Also there has to be a reference structure from which the movement can be compared. In our case the road profile is the reference. But, the bridge structure deterioration might also change the profile of the road. This problem is solved by comparing the 3D road profile generated in the current run with that of previous runs. If any changes are detected; the bridge engineers are notified for further careful and diligent inspection using static methods which give more accurate measurement.

After comparing the 3D road profile, there is the need for stabilization of the camera platform. One of the modules of this research project is the implementation of real time camera jitter removal. The fundamental aim of this module is to make the camera platform follow the road profile. For this a control system was simulated. The parameters of this control system are gyroscope data, from the road profiling system and other from the camera platform, which has

to be stabilized. The accelerometer data is used to remove the test vehicle movement. The accelerometer data is double integrated to get the displacement using the Trapezoidal rule to find the area under the curve. A Proportional integral- derivative (PID) controller is simulated for real time jitter reduction. Though the time response of in simulation is around 10ms, the fact that PID controller takes few extra milliseconds' to settle down, there is the need for post-processing to remove error due to jitter.

4.1.1 Jitter Removal Using Post Processing.

Since all the sensors are synchronized in software using a start sensor and sync signal, we can easily integrate the data from camera, two gyroscopes (one mounted on the camera platform and the other with road profiling system). We first calculate the difference in the reading of two gyroscopes, which will give the amount of jitter present in camera. If there is difference in roll of camera position, then that error with 3D world is projected as rotation error in the 2D camera world. This error is compensated using 'rotate' in image processing. Similarly there can be error due to pitch error. Pitch error in the 3D world projects as vertical translation error in 2D camera world. We can detect this angle using homographic technique but to correct the angle it is very difficult, even using post processing. But since we will be using scale invariance transform for image registration and matching, this error if not more than 10^0, can be incorporated.

In this section we propose a novel methodology of integrating different sensors to accurately detect any bridge surface movement and also how the camera platform is tied with this bridge surface and correcting any error due to camera jitter. The above mentioned step is critical since now we can take the bridge surface as reference and detect any structural movement.

4.2 Local Feature Extraction and Image Matching

In this section we will cover the implementation of local feature detection and image matching algorithm. Image Matching features across different images in a common problem in

computer vision. The important characteristics that feature detection and matching algorithm must possess are scale invariance, rotation invariance, robust to illumination, blur invariance, and computational complexity. [15]

Here we have considered the Scale Invariant Feature Transform (SIFT) algorithm, because it isn't just scale invariant but also invariant to rotation, illumination and viewpoint, though not the fastest. [16] [17]

The algorithm

SIFT is complex algorithm involving number of steps.

1. **Constructing a scale space.**

   For the algorithm we create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".

Scale space:

Since we want to detect edges and corners as features, we first need to filter out the noise present in the image, i.e., we need to get rid of some details. While getting rid of these details, we must ensure that we do not introduce new false details. The only way to do that is with the Gaussian Blur (it was proved mathematically, under several reasonable assumptions). So to create a scale space, we take the original image and generate progressively blurred out images. Then, we resize the original image to half size. We then generate blurred out images again. Images of the same size (vertical) form an octave. The figure 4.1 below is four octaves of an image. Each octave has 5 images. The individual images are formed because of the increasing "scale" (the amount of blur).

Octaves and Scales

The number of octaves and scale depends on the size of the original image. While programming the SIFT algorithm, one can have a variable number of octaves and scales, depending on the application. However, the creator of SIFT suggests that 4 octaves and 5 blur levels are ideal for the algorithm, which is what we have used in our experiment.

The first octave: For the first octave we have doubled the original size of the image and anti-aliased it (by blurring) since the algorithm produces more four times more key-points. The more the key-points, better the performance of the algorithm.

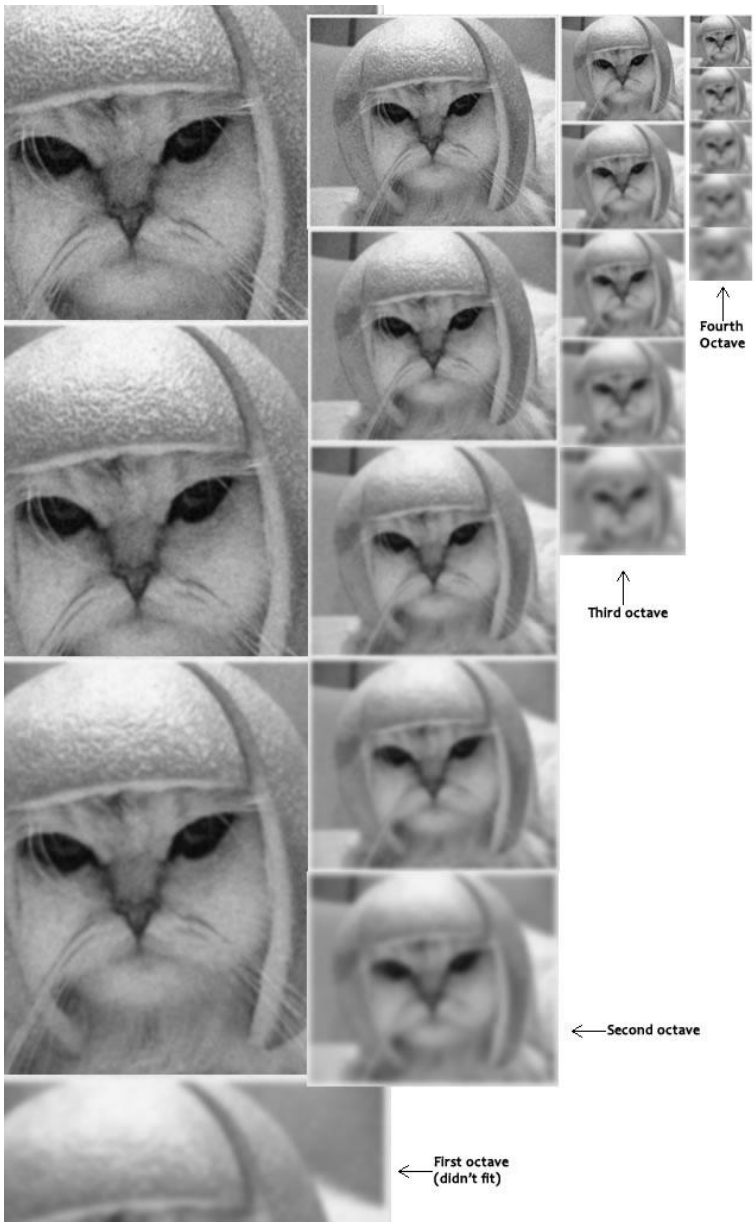

Figure 4.1 Example of Scale Spaced and Octave of Image.

Amount of blurring: The amount of blurring in each image is important. For example, assume the amount of blur in a particular image is σ. Then, the amount of blur in the next image will be k*σ. Here k is constant.

In the first step of SIFT, we generate several octaves of the original image. Each octave's image size is half the previous one. Within an octave, images are progressively blurred using the Gaussian Blur operator.

2. **LoG Approximation**

The Laplacian of Gaussian is useful for finding interesting key points in an image. But it's computationally expensive. So we approximate it using the representation created earlier.

In the previous step, we created the scale space of the image. The idea was to blur an image progressively, shrink it, and blur the small image progressively and so on. Now we use those blurred images to generate another set of images, the Difference of Gaussians (DoG). These DoG images are a useful for finding out key points in the image.

Laplacian of Gaussian: For the Laplacian of Gaussian (LoG) operation, we fist blur an image. Then, we calculate second order derivatives to get (or, the "Laplacian"). This locates edges and corners on the image. These edges and corners are good for finding keypoints. However the second order derivative is extremely sensitive to noise. The blur smooth's out the noise and stabilizes the second order derivative. The problem is, calculating all second order derivatives is computationally intensive. So we calculate the difference between two consecutive scales, or, the Difference of Gaussian (DoG). Figure 4.3 below shows the DoG graphically.

These Difference of Gaussian images are approximately equivalent to the Laplacian of Gaussian. Thus we have replaced a computationally intensive process with a simple subtraction

(fast and efficient). These DoG approximations are also “scale invariant”, and produced a much better track able points.[3]

We know the DoG result is multiplied with σ^2, but it’s also multiplied by another number. That number is (k-1). This is the k we discussed in the previous step. We will be only looking for the *location* of the maximums and minimums in the images. We’ll never check the actual values at those locations. So, this additional factor won’t be a problem. (Even if we multiply throughout by some constant, the maxima and minima stay at the same location)

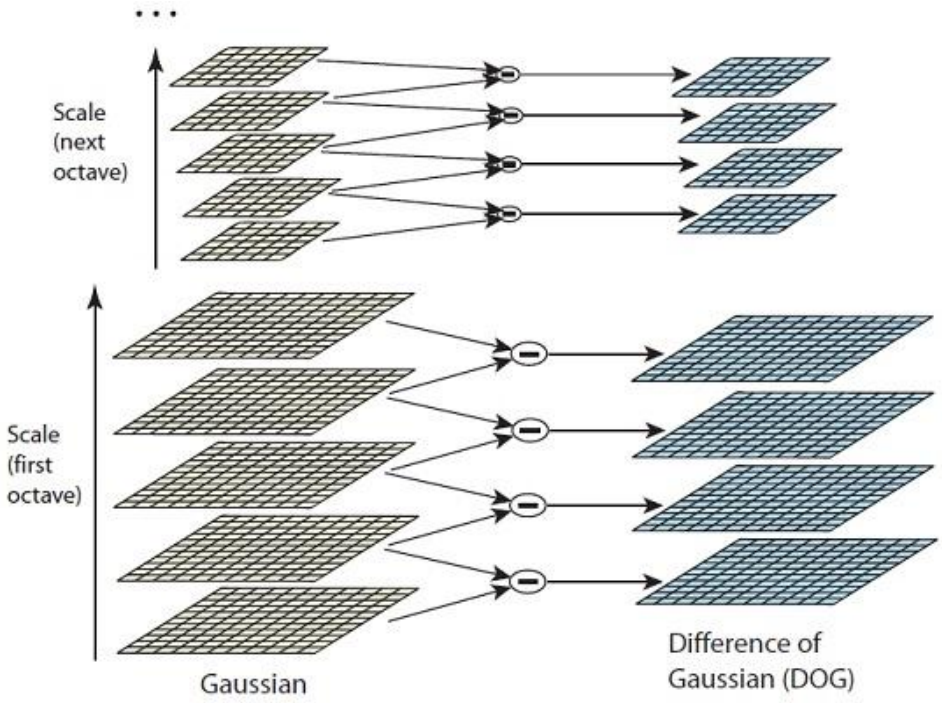

Figure 4.2 Difference of Gaussian

The figure 4.4 shows the subtraction for just one octave. The same thing is done for all octaves. This generates DoG images of multiple sizes.

To summarize this step, two consecutive images in an octave are picked and one is subtracted from the other. Then the next consecutive pair is taken, and the process repeats. This is done for all octaves. The resulting images are an approximation of scale invariant LoG

(which is good for detecting eypoints). There are a few “drawbacks” due to the approximation, but they won’t affect the algorithm.

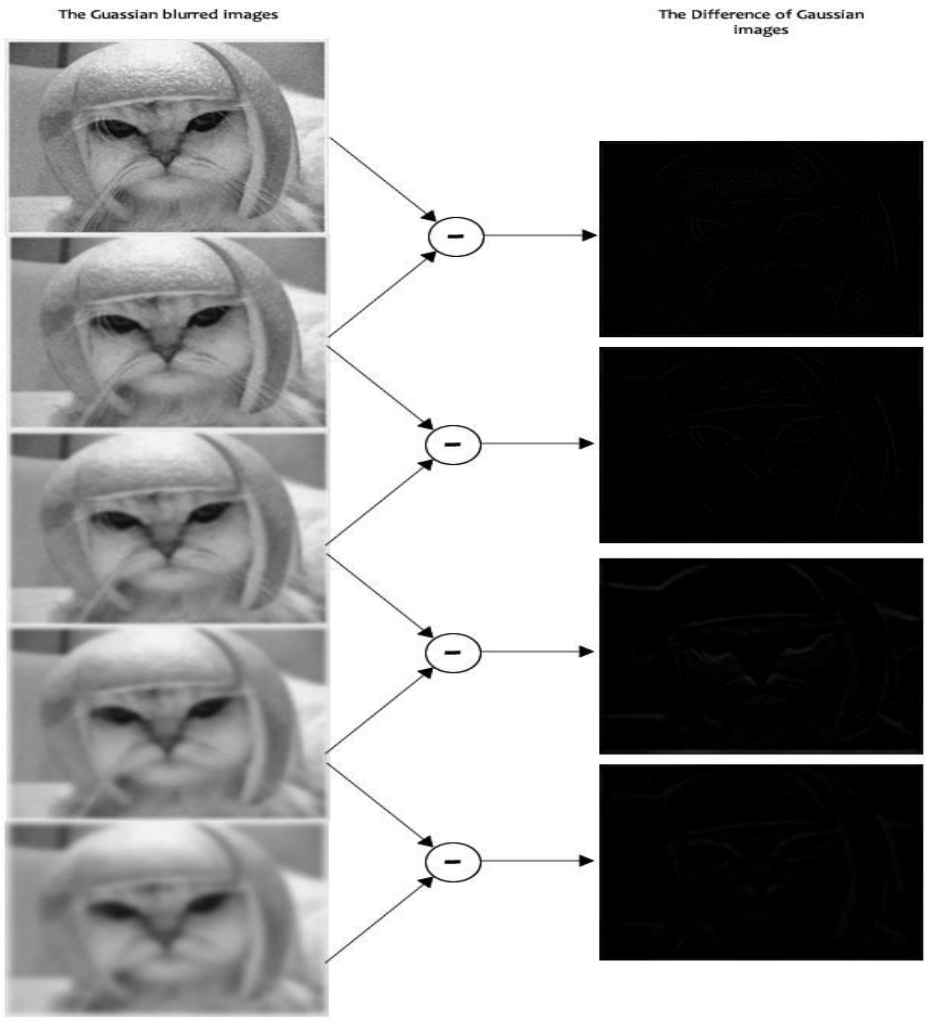

Figure 4.3 Example of Difference of Gaussian

3) **Finding keypoints.**
We now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2

Up till now, we have generated a scale space and used the scale space to calculate the Difference of Gaussians, which is scale invariant. Finding key points is a two part process:

1. Locate maxima/minima in DoG images

2. Find sub pixel maxima/minima

Locate maxima/minima in DoG images

The first step is to coarsely locate the maxima and minima. We iterate through each pixel and check all its neighbors. Then check is done within the current image, and also the one above and below it.

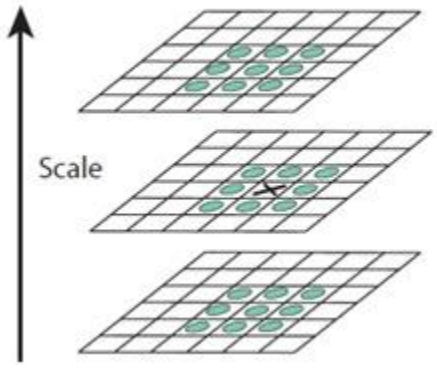

Figure 4.4 Locating Maxima Minima in Scale Space

X marks the current pixel. The green circles mark the neighbors. This way, a total of 26 checks are made. X is marked as a “key point” if it is the greatest or least of all 26 neighbors**.** Usually, a non-maxima or non-minima position won’t have to go through all 26 checks. A few initial checks is usually sufficient to discard it. Note that keypoints are not detected in the lowermost and topmost scales. There are simply enough neighbors to do the comparison. So we skip them.

Once this is done, the marked points are the approximate maxima and minima. They are “approximate” because the maxima/minima never lie exactly on a pixel. It lies somewhere between the pixel. But we simply cannot access data “between” pixels. So, we must mathematically locate the sub pixel location.

Find sub pixel maxima/minima:

Using the available pixel data, sub pixel values are generated. This is done by the Taylor expansion of the image around the approximate key point.

Mathematically, it's like this:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{4.1}$$

We can easily find the extreme points of this equation (differentiate and equate to zero). On solving, we'll get sub pixel key point locations. These sub pixel values increase chances of matching and stability of the algorithm.

The figure 4.6 shows just one octave. This is done for all octaves. Also, this figure shows only the first part of keypoint detection. The Taylor series part has been skipped.

To summarize this step, we detected the maxima and minima in the DoG images generated in the previous step. This is done by comparing neighboring pixels in the current scale, the scale "above" and the scale "below".

4) Get rid of bad key points.

Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here.

Key points generated in the previous step produce a lot of key points. Some of them lie along an edge, or they don't have enough contrast. In both cases, they are not as features. So we get rid of them. The approach is similar to the one used in the Harris Corner Detector for removing edge features. For low contrast features, we simply check their intensities. [11]

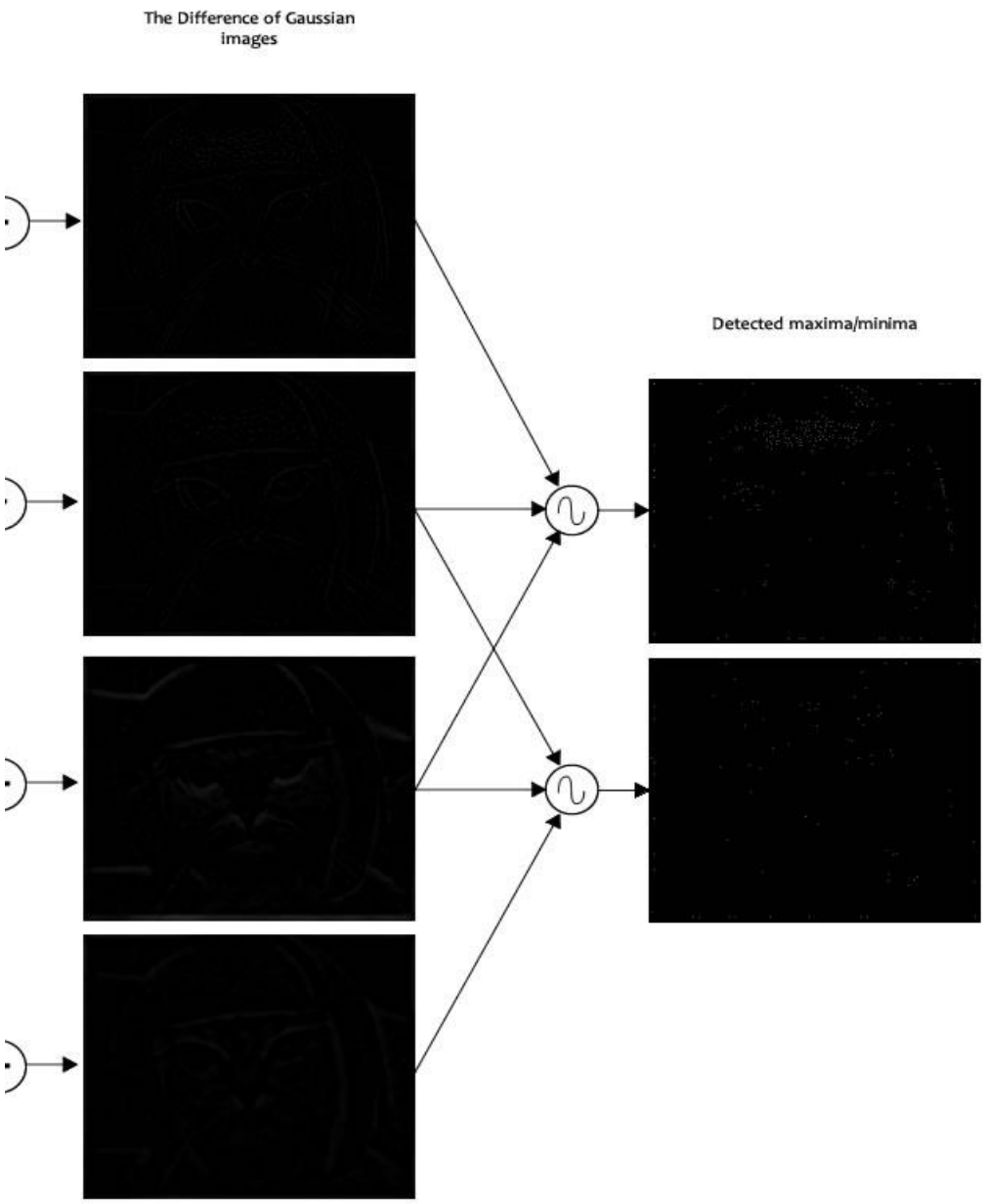

Figure 4.5 Example of Maxima Minima

Removing low contrast features

If the magnitude of the intensity (i.e., without sign) at the current pixel in the DoG image (that is being checked for minima/maxima) is less than a certain value, it is rejected. Because we have subpixel keypoints (we used the Taylor expansion to refine keypoints), we again need to use the Taylor expansion to get the intensity value at subpixel locations. If its magnitude is less than a certain value, we reject the keypoint.

Removing edges

The idea is to calculate two gradients at the keypoint, both perpendiculars to each other. Based on the image around the keypoint, three possibilities exist. The image around the keypoint can be:

A flat region: If this is the case, both gradients will be small.

An edge: Here, one gradient will be big (perpendicular to the edge) and the other will be small (along the edge)

A "corner": Here, both gradients will be big. Corners are great keypoints. So we want just corners. If both gradients are big enough, we let it pass as a key point. Otherwise, it is rejected.

Mathematically, this is achieved by the Hessian Matrix. Using this matrix, we can easily check if a point is a corner or not.

Here's a visual example of what happens in this step:

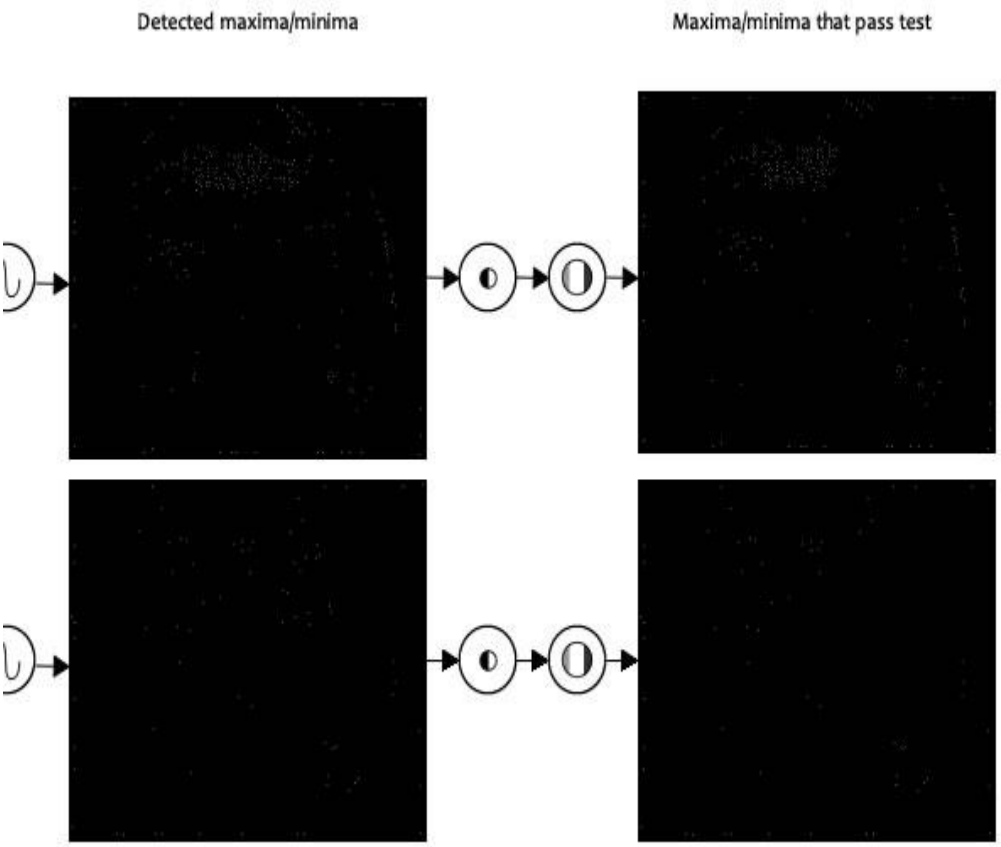

Figure 4.6 Selecting Stable Keypoints

Both extreme images go through the two tests: the contrast test and the edge test. They reject a few keypoints (sometimes a lot) and thus, we're left with a lower number of keypoints to deal with.

In this step, the number of keypoints was reduced. This helps increase the efficiency and also the robustness of the algorithm. Keypoints are rejected if they had a low contrast or if they were located on an edge.

5) Assigning an orientation to the keypoints.

An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.

After step 4, we have genuine key points. They've been tested to be stable. We already know the scale at which the keypoint was detected (it's the same as the scale of the blurred image). So we have scale invariance. The next thing is to assign an orientation to each keypoint. This orientation provides rotation invariance. The more invariance we have the better it is.

The idea is to collect gradient directions and magnitudes around each keypoint. Then we figure out the most prominent orientation(s) in that region. And we assign this orientation(s) to the keypoint. Any later calculations are done relative to this orientation. This ensures rotation invariance.

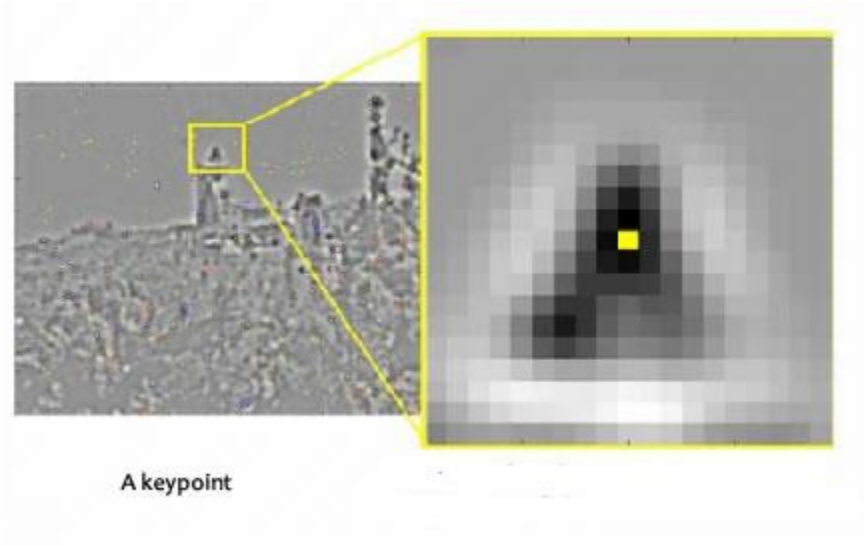

Figure 4.7 Example of keypoint

The size of the "orientation collection region" around the keypoint depends on its scale, bigger the scale, the bigger the collection region.

Gradient magnitudes and orientations are calculated using these formulae:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad (4.2)$$

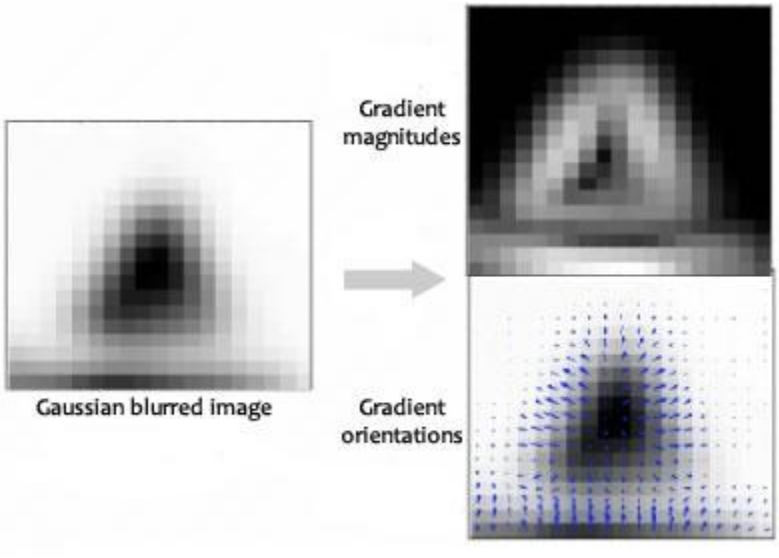

Figure 4.8 Gradient calculation

The magnitude and orientation is calculated for all pixels around the keypoint. Then, a histogram is created for this. In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees). Let's say the gradient direction at a certain point (in the "orientation collection region") is 18.759 degrees, and then it will go into the 10-19 degree bin. And the "amount" that is added to the bin is proportional to the magnitude of gradient at that point.

Once we've done this for all pixels around the keypoint, the histogram will have a peak at some point. Below, we see the histogram peaks at 20-29 degrees. So, the keypoint is assigned orientation 3 (the third bin) Also, any peaks above 80% of the highest peak are converted into a new keypoint. This new keypoint has the same location and scale as the original. But its orientation is equal to the other peak. So, orientation can split up one keypoint into multiple keypoints.

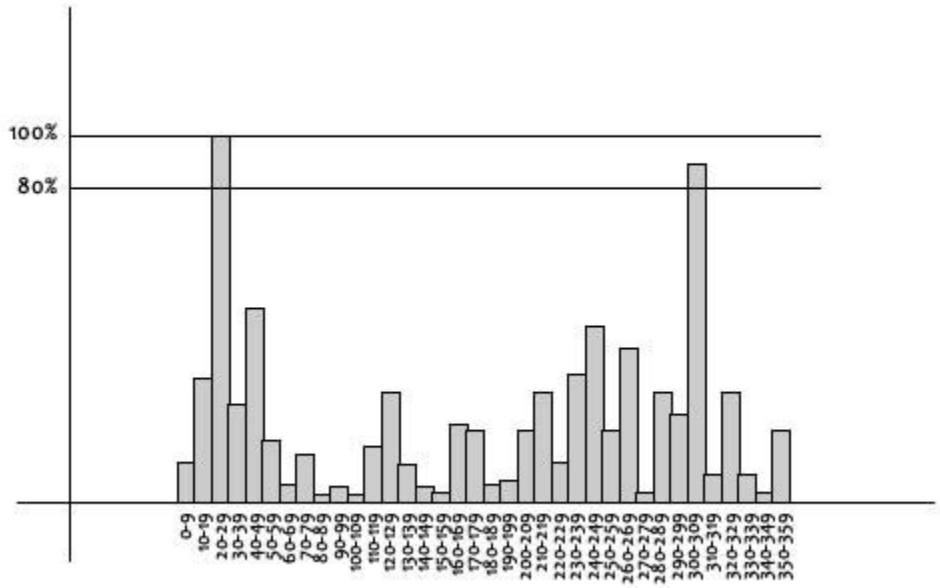

Figure 4.9 Example Histogram of gradient

The window size, or the “orientation collection region”, is equal to the size of the kernel for Gaussian Blur of amount 1.5*sigma.

To assign an orientation we use a histogram and a small region around it. Using the histogram, the most prominent gradient orientation(s) are identified. If there is only one peak, it is assigned to the keypoint. If there are multiple peaks above the 80% mark, they are all converted into a new keypoint (with their respective orientations).

Next, we generated a highly distinctive “fingerprint” for each keypoint. This fingerprint, or “feature vector”, has 128 different numbers.

6) Generate SIFT features.

Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Let’s say we have 5,000 features. With this representation, we can easily identify the feature we’re looking for (say, a beam or column of bridge structure).

Till now, we had scale and rotation invariance. Now we create a fingerprint for each keypoint. This is to identify a keypoint. If an eye is a keypoint, then using this fingerprint, we'll be able to distinguish it from other keypoints, like ears, noses, fingers, etc.

We want to generate a very unique fingerprint for the keypoint. We also want it to be relatively lenient when it is being compared against other keypoints. Things are never EXACTLY same when comparing two different images. To do this, a 16×16 window around the keypoint is taken. This 16×16 window is broken into sixteen 4×4 windows.

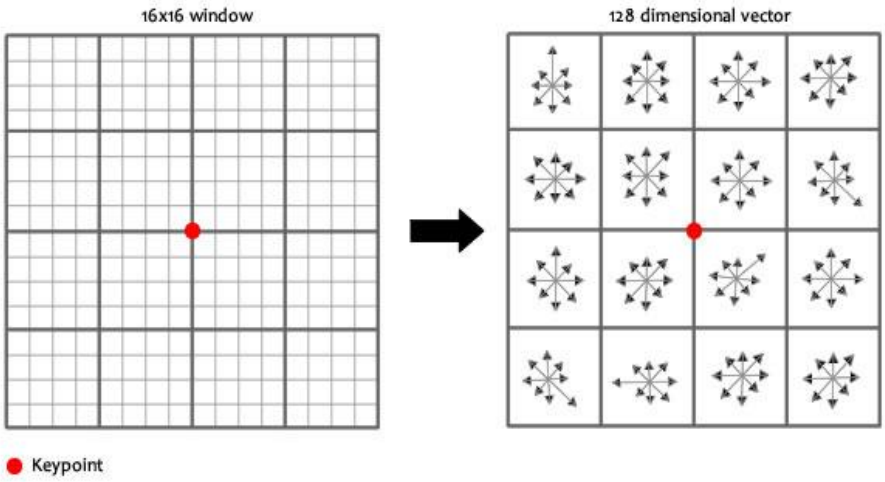

Figure 4.10 Calculating Gradient Magnitude

Within each 4×4 window, gradient magnitudes and orientations are calculated. These orientations are put into an 8 bin histogram.

Any gradient orientated in the range of 0-44 degrees is added to the first bin, 45-89 added to the next bin. And so on. And (as always) the amount added to the bin depends on the magnitude of the gradient. Unlike the past, the amount added also depends on the distance from the keypoint. So gradients that are far away from the keypoint will add smaller values to the histogram. This is done using a "Gaussian weighting function". This function simply generates a gradient (it's like a 2D bell curve). We multiple it with the magnitude of orientations, and we get a weighted gradient.

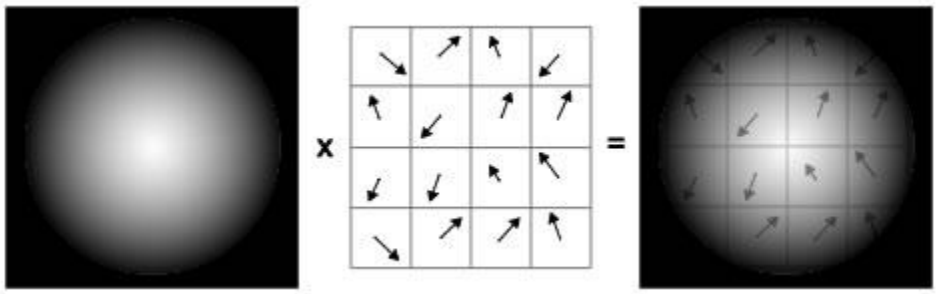

Figure 4.11 Calculating Gradient Orientation

Doing this for all 16 pixels, we would've "compiled" 16 totally random orientations into 8 predetermined bins. We do this for all sixteen 4×4 regions. So we end up with 4x4x8 = 128 numbers. Once we have all 128 numbers, we normalize them (divide by root of sum of squares). These 128 numbers form the "feature vector". This keypoint is uniquely identified by this feature vector.

We might have seen that in the pictures above, the keypoint lies "in between". It does not lie exactly on a pixel. That's because it does not. The 16×16 window takes orientations and magnitudes of the image "in-between" pixels. So we need to interpolate the image to generate orientation and magnitude data "in between" pixels.

This feature vector introduces a few complications. We need to get rid of them before finalizing the fingerprint.

1. **Rotation dependence:** The feature vector uses gradient orientations. Clearly, if we rotate the image, everything changes. All gradient orientations also change. To achieve rotation independence, the keypoint's rotation is subtracted from each orientation. Thus each gradient orientation is relative to the keypoint's orientation.

2. **Illumination dependence:** If we use threshold to consider numbers that are big, we can achieve illumination independence. So, any number (of the 128) greater than 0.2 is changed to 0.2. This resultant feature vector is normalized again. And now we have an illumination independent feature vector.

Figures 4.13, 4.14 and 4.15 shows the results of implementing SIFT on an image of the overpass. The image in figure 4.13 is the original image without any rotation. The image in figure 4.14 is rotated to an arbitrary value for experimental purpose. These two figures shows the local features of the image and its orientation. Figure 4.15 shows the matching of above mentioned image and also the visualization of that matching. The overpass shown in figure 4.12 is perfectly functional, but for the experimental purpose, the image is rotated to a small arbitrary value and that amount of rotation is detected using Least Mean Square Difference Image filter.

Figure 4.12 Original image of an overpass under consideration.

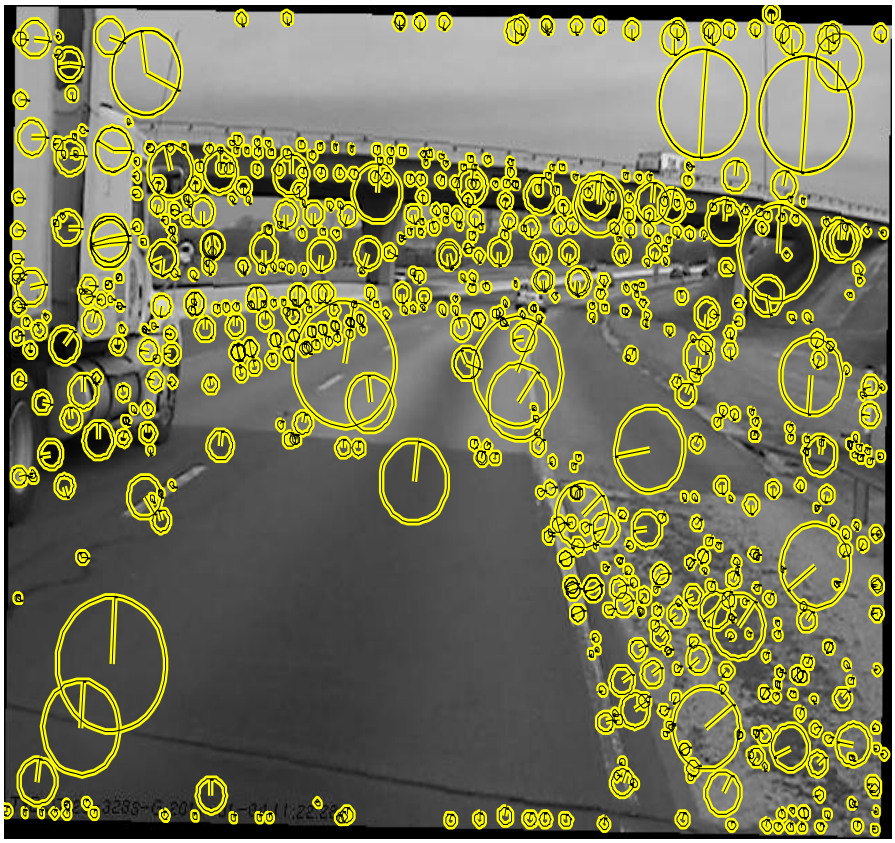

Figure 4.13 Local Features of the original image and its orientation

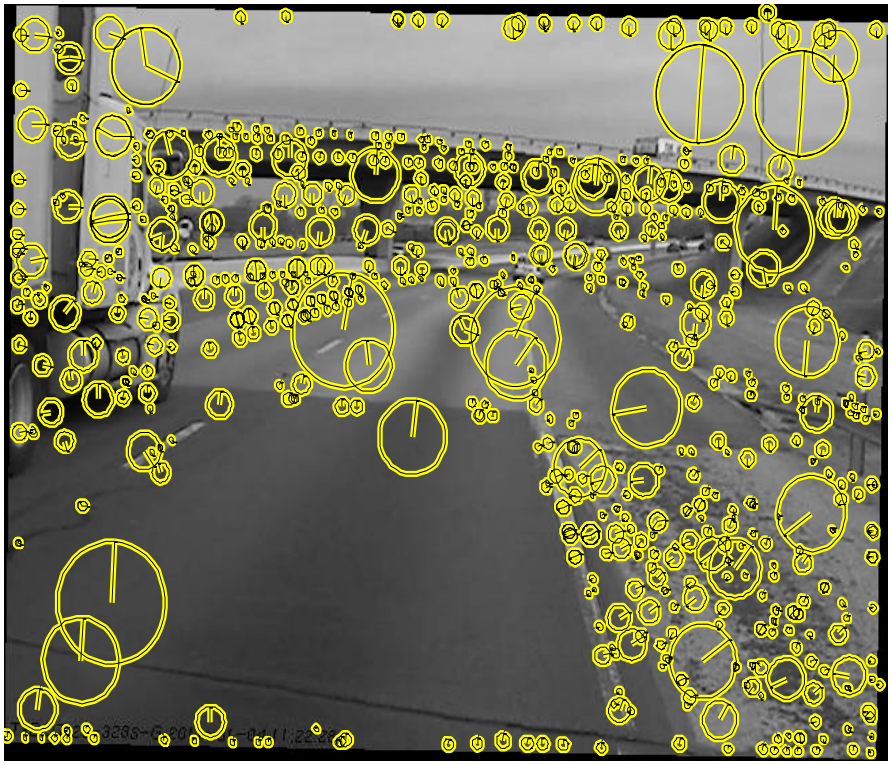

Figure 4.14 Local features of the rotated image and its orientation

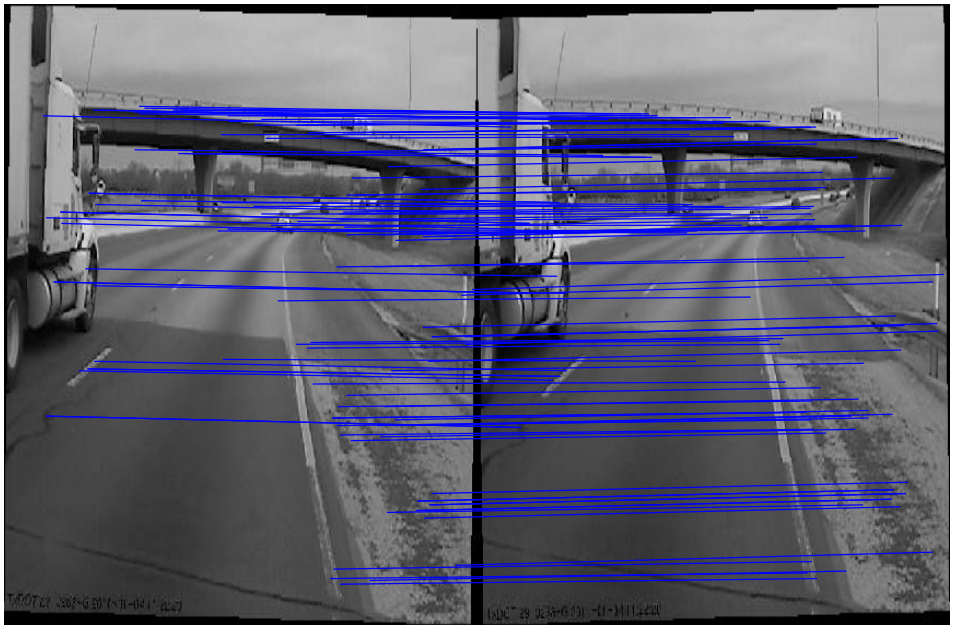

Figure 4.15 Image Matching of the two images.

By implementing the mentioned algorithm we make sure that image that we want to compare for structural movement is viewed from same position and there is minimum error.

4,3 Least Mean Square Image Difference

After matching the two images from a view position, we can now detect any structural movement simply by using Least Mean Square Error Estimator (LMSE). Basically LMSE uses steepest decent to find the weigh filter to minimize the error term. It is similar to difference filter, but LMS image difference is more robust in terms of size and scale factor.[18] [4]

In least square filtering, $q(u,v)$ is commonly chosen as a finite difference approximation of Laplacian operator. Here, for example we have taken a square with spacing $\Delta x = \Delta y = 1$, and $\alpha = \frac{1}{4}$, we get

$$q(u,v) \triangleq -\delta(u,v) + \alpha[\delta(u-1,v) + \delta(u+1,v) + \delta(u,v-1) + \delta(u,v+1)] \quad (4.5)$$

For detailed explanation and derivation of LMSE estimator please refer [18].

Using the above equation we can get the difference in two frames and thus we can detect any structural movement. As discuss earlier, we have rotated the original image to an arbitrary value, now using LMSE Difference we can detect and measure the rotation. Also the error coefficient when multiplied with a scale can lead us to measuring the displacement. Since this experimental image does have more information about the overpass i.e. its structural size, here we are not able to measure any displacement in bridge structure, none the less we are able to detect any movement. Figure 4.16 shows the result of implementation of LMS Difference to the two images.

Figure 4.16 Detection of structural changes of the overpass

To conclude, this chapter covered the methodology to remove any error due to camera jitter by integrating several sensors and using post processing techniques, it also includes the concept of taking bridge surface or road profile as reference. We covered the implementation of different computer vision algorithms on our experimental image of an overpass.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this research project a novel method for periodic Bridge Health Monitoring using application of computer vision and 3D road profile is proposed.

There is need for an embedded framework for the bridge monitoring and maintenance. In this proposed method, we integrated 3D road profile with the video data of the bridge to detect the structural movement. The sensors such as lasers, accelerometers, gyroscopes, and cameras were integrated to obtain the data of the structure. This structure data was then processed using computer vision algorithms to detect the structural changes.

Structural movement is detected by comparing the structure position in two images, with respect to a stable reference. The bridge deck movement is detected by comparing the 3D road profile of the deck. The bridge deck is taken as a reference.

For the feature detection, performance of three different algorithms, Speeded Up Robust Feature (SURF), Scale Invariant Feature Transform (SIFT), and Principal Component Analysis (PCA) – SIFT for this particular application were studied. Although SURF is the fastest to compute features, it is not stable to rotation and illumination changes. Performance of PCA-SIFT is affected by the blurring in the image. Though SIFT is slow by embedded application standards it gives near real time performance on General Purpose Processors. SIFT is also invariant to scale, rotation, and blurring and it produces a large number of keypoints. [15]

The SIFT algorithm was improved as per the requirement of the application for feature detection. An Image matching algorithm was developed using features produced by SIFT. Finally, $\sum\Delta$ filter was used to detect the changes in bridge structures.

The research work covered the complexity involved in detecting the bridge movement using video analysis. It covered the solution to challenges associated with motion camera system. This includes removing the vehicle movement, jitter reduction, illumination, change in view point, etc.

Experimental results are conclusive enough to state that this project is feasible for bridge monitoring systems. It is important to note that only the transverse bridge movement can be detected.

Since this is a the research application there is a lot of room for improvement.

- Improve the accuracy of movement measurement
- Reduce the computational complexity of the SIFT algorithm
- Develop the embedded board with FPGA and DSP integrated on board for implementing the SIFT. [19]

REFERENCES

[1] K. Wardhana and F. Hacliprionо (2000), Analysis of Recent Bridge Failures in the United States. *J. Perform. Contr. Facil., 17 (3)* pp.144-150

[2] J. Lee, I Fukuda, M. Shinozuka, S. Cho, and C. Yun (2006). Development and Application of a Vision Based Displacement Measurement System for Structural Health Monitoring of Civil Structures. *Smart Structures and System 3 (3).* pp. 373-384

[3] R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge university press, second edition, 2003.

[4] L. Shapiro and G. Stockman. Computer Vision. Prentice Hall, 2001

[5] R. Gonzalez and R. Woods. Digital Image Processing. Prentice Hall, third edition, 2008

[6] A. Jain. Fundamental od Digital Image Processing. Prentice Hall, 1989

[7] I. Abdel-Qader, O. Abudyayjen, and M. Kelly (2003). Analysis of Edge Detection Techniques for Crack Identification in Bridges. *Journal of Computing in Civil Engineering* 4(3) pp.255-263.

[8] L. S. Davis. A Survey of Edge Detection Techniques (1975). *Computer and Image Processing* 4(3) pp. 248-270

[9] http://homepages.inf.ed.ac.uk/rbf/HIPR2/roberts.htm

[10] J. Canny (1986). A Computational Approach to Edge Detection. *IEEE Trans. Pattern Analysisand and Machine Intelligence* 8 (6) pp. 679-698.

[11] C. Harris and M. S. Plessey A COMBINED CORNER AND EDGE DETECTOR Research Roke Manor, United Kingdom © The Plessey Company pic. 1988

[12] B. Kisačanin, S. S. Bhattacharyya, and S. Chai. Embedded Computer Vision. Springer, 2009

[13] http://opencv.willowgarage.com/wiki/Welcome

[14] http://www.mathworks.com/products/matlab/

[15] L. Juan and O. Gwun (2009). A Comparison of SIFT, PCA-SIFT, and SURF. *International Journal of Image Processing* 3 (4) pp. 143-152.

[16] D. G. Lowe. (2004, Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60(2),* pp. 91-110. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[17] D. G. Lowe (1999). Object Recognition from Local Scale-Invariant Features. *Proc. of International Conference on Computer Vision*.

[18] B. R. Hunt. (1973, The application of constrained least squares estimation to image restoration by digital computer. *Computers, IEEE Transactions on C-22(9),* pp. 805-812.

[19] C. Arth and H. Bischof (2008). Real-time object recognition using local features on a DSP-based embedded system. *Journal of Real-Time Image Processing 3(4),* pp. 233-253.

BIOGRAPHICAL INFORMATION

Kenan Modi received his Bachelor of Engineering in Electronics Engineering from Mumbai University in 2009. He enrolled in Master of Science in Electrical Engineering program at University of Texas at Arlington in 2009. He worked as Graduate Research Assistant in Transportation and Instrumentation Lab from August 2010. He will be receiving his Master of Science in Electrical Engineering degree in August 2011.