Tangible Images

Digital Signal Processing Application Assignment Report

Manav Kataria
Jhelum Chakravorty
K. L. Srinivas
Neelam Maniar
Hussain Manasawala

IIT Bombay

Instructed by:
Prof. V. M. Gadre

November 2, 2009

Acknowledgement

We would like to take this opportunity to express our sincere thanks and gratitude to our instructor Prof. Vikram M. Gadre for his invaluable guidance and support and the Teaching Associates for their help.

Abstract

This project describes an algorithm for generation of forces from two-dimensional bitmapped images. The forces allow the user to feel the contours of the image with sufficient realism using haptic devices. We also propose an algorithm to generate textures for virtual surfaces in a simple manner. The main focus of this project pertains to generating a model for synthesizing haptic interaction with static images. We hope to compare our approach and improve upon the work by Manivannan and Vasudevan [1].

Primary Keywords : Haptic Feedback, Image Processing, Virtual Reality

Secondary Keywords : Haptic Interaction, Rendering, Two Dimensional Bitmapped Digital Image, Image Segmentation, Image Texture, Haptic Texture

Contents

List of Figures

Chapter 1

Overview

The project can be divided into the following major tasks / phases. Proposed Algorithm :

1. Conceive a (tangible) surface model from a given bitmap image using depth-detection or contour [1] method. This model could be programmed on a haptic device to feel the shape/geometry of the generated virtual surface.

2. Identify image-segments with known texture properties using gradient-method, [2] or others. The image segments would be mapped to haptic textures (see 3).

3. Map segments to haptic texture models like sand, ice, etc.

4. Generating Forces : If feasible, implement the algorithm on the haptic device Novint Falcon using Haptik Library [3] on Matlab.

5. If time permits, design a suitable GUI that synchronizes Haptics Interface Point from Falcon and the input image.

Image Segmentation : Image segmentation is a process of dividing an image into a number of different regions such that each region is homogeneous with respect to certain visual characteristics (but the union of any two adjacent regions is not.)[4]Algorithm :

1. Convert image to gray scale.

2. Detect edge using canny filter or others.

3. Use gradient magnitude [2] method to segment image.

Processing content :

- Edge Detection
- Image Texture Identification and
- Image Segmentation (using image filters, convolution and correlation as required)

Haptic Feedback : What is Haptics : The science of sensing and manipulation through touch is called Haptics [6]. Recent advances in science and technology allow a user to touch and feel a virtual object using Haptic devices. One such device called Novint Falcon is a part of the Haptics Lab, IIT Bombay. If feasible we would attempt to physically generate Haptic Surfaces or Textures [5].
Haptic Surface : Shape or Geometry of an object that a user can feel or touch. We intend to define a $3-D$ landscape from an image that is tangible/touchable. The generation of Haptic Surface may be based on depth detection or contour method [1].
Algorithm :

- Start with a bounded rectangular planar surface model
- Add height or depth to the planes locales corresponding to the image properties at the respective locale. May be derived from depth-detection/contour based methods.

- OR-
Haptic Texture : One of the possible objectives would be to generate textures haptically to give the user a feel of the image/objects coarseness or smoothness. Haptic Texture can be defined as the repetitive or random deviation from the nominal surface that forms the three dimensional surface topography. For e.g attributes of surface like lay, waviness, roughness and flaws that a user can feel through touch Image-segmentation using gradient method, histogram or others would be used as input to generate haptic textures.

Chapter 2

Depth Detection

2.1 Introduction

2.1.1 Stereopsis : Human Stereo Vision

Stereopsis is the process of visual perception leading to the sensation of depth from the two slightly different projections of the world onto the retinas of the two eyes. The difference in the two retinal images is called Binocular Disparity which refers to the difference in image location of an object seen by the left and right eyes, resulting from the eyes' horizontal separation. In computer vision, binocular disparity refers to the same difference seen by the two different cameras instead of eyes.
The adjacent figures elucidate the idea of distance and depth detection by stereo vision of eyes.

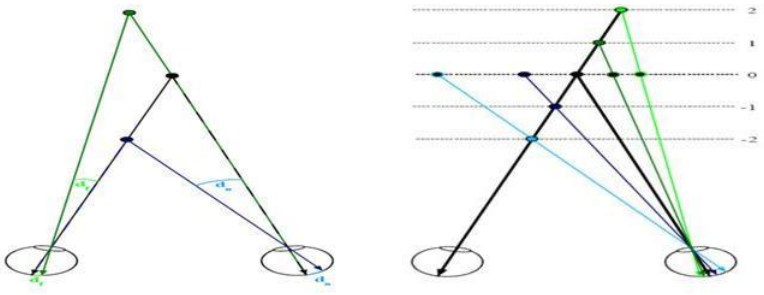

Figure 2.1: Detection of Distance: Far and Near ; Detection of Depth in the Plane

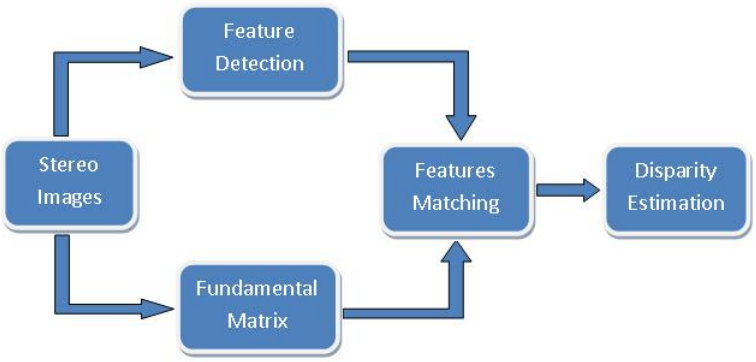

Figure 2.2: Block Diagram for Depth Detection

2.2 Computer Stereo Vision

The stereo vision of human eye motivates the computer vision to compute depth of an image through disparity estimation between two stereo images. It follows the same mechanism that human eye does. Depth detection of an input image is done by the disparity estimation of stereo vision. Stereo vision refers to the ability to infer information about the $3-D$ structure of a scene using two images captured from different viewpoints. The disparity map gives the relative depths between the stereo images.

2.3 Disparity Estimation Model

Fig. (2.2) depicts the process of depth detection.
The working principle of each block in the previous figure is given below in brief :

- **Stereo images** are the pair of images of the same object or scene taken by two cameras separated by a small distance.
- **Feature detection** involves the detection of the feature points which are the edges and corners in the stereo images. Harris' algorithm helps in finding the corners.

- **Fundamental matrix** describes the *epipolar geometry* of stereo vision system. It correlates the corresponding points of the stereo images using epipolar geometry.
- **Feature matching** is the procedure to match the corresponding points in the stereo images. This method calls for the computation of normalized cross correlation between the corresponding points in the stereo image pair.
- **Disparity Estimation** maps the disparity between similar points in the two images to a gray scale intensity. The pixel intensities describe the relative depth of points in the scene.

2.4 Literature Review

2.4.1 Combined Edge and Corner Detection using Harris' Algorithm

The mathematical basis of the edge and corner detection is explained here in brief. The content of this section is taken from reference [7]
Denoting the image intensities by I, the change E produced by a shift (x, y) is given by :

$$\begin{aligned} E_{x,y} &= \sum_{u,v} w_{u,v}[I_{x+u,y+v} - I_{u,v}]^2 \\ &= \sum_{u,v} w_{u,v}[xX + yY + O(x^2 + y^2)]^2 \end{aligned} \tag{2.1}$$

where the first gradients are approximated by:

$$\begin{aligned} X &= I \otimes (-1, 0, 1) \\ &= \frac{\partial I}{\partial x} \end{aligned} \tag{2.2}$$

$$\begin{aligned} Y &= I \otimes (-1, 0, 1)^T \\ &= \frac{\partial I}{\partial y} \end{aligned} \tag{2.3}$$

Hence for small shifts E can be written as :

$$E(x, y) = Ax^2 + 2Cxy + By^2 \tag{2.4}$$

where $A = X^2 \otimes w$, $B = Y^2 \otimes w$, $C = XY \otimes w$. The response is noisy because the window w is binary and rectangular. Hence a smooth circular window, such as Gaussian is used which is given by the following :

$$w_{u,v} = \exp(-\frac{(u^2 + v^2)}{2\sigma^2}) \tag{2.5}$$

The change E for small shift (x, y) can be concisely written as :

$$E(x, y) = (x, y)M(x, y)^T \tag{2.6}$$

where the 2×2 symmetric matrix M is given by :

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \tag{2.7}$$

It is also necessary to have the measure of corner and edge quality or response. The size of the response will be used to select isolated corner pixels and find the edge. It is also necessary to have the measure of corner and edge quality or response. The size of the response will be used to select isolated corner pixels and find the edge pixels. Consider the trace and determinant of the matrix M as Tr and Det, then the corner response is given by :

$$R = Det - k(Tr)^2 \tag{2.8}$$

R has high positive value in the corner region, high negative value in the edge region and small value in the flat region.

2.4.2 Fundamental Matrix

The contents of this and subsequent section are referred from [10].
In computer vision, the fundamental matrix F is a 3×3 matrix which relates corresponding points in stereo images. In *epipolar* geometry, with the image coordinates U_r and U_l of corresponding points of stereo image pair, FU_r describes an epipolar line on which the corresponding point U_l on the other image must lie. That means for all pairs of corresponding points, the following holds :

$$U_l^T F U_r = 0 \tag{2.9}$$

2.4.3 Eight Point Algorithm

This is a linear method for extracting the fundamental matrix from a list of matched pairs of points. The fundamental matrix has rank 2 and 7 degrees of freedom and hence can be recovered from only seven correspondences. However, the algorithm is much simpler if we have eight correspondences. For any given matched pair of points $U_l = [c_l \; r_l]$ and $U_r = [c_r \; r_r]$, the *epipolar constraint* as mentioned above is given by:

$$\begin{pmatrix} c_l & r_l & 1 \end{pmatrix} \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} c_r \\ r_r \\ 1 \end{pmatrix} = 0 \tag{2.10}$$

Given 8 such correspondences, we can recover the matrix F from the homogeneous system given below :

$$A^{8\times 9} f^{9\times 1} = 0 \tag{2.11}$$

where $A = \begin{pmatrix} c_{l_1}c_{r_1} & c_{l_1}r_{r_1} & c_{l_1} & r_{l_1}c_{r_1} & r_{l_1}r_{r_1} & c_{l_1}c_{r_1} & c_{r_1} & r_{r_1} & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ c_{l_8}c_{r_8} & c_{l_8}r_{r_8} & c_{l_8} & r_{l_8}c_{r_8} & r_{l_8}r_{r_8} & c_{l_8}c_{r_8} & c_{r_8} & r_{r_8} & 1 \end{pmatrix}$,

and $f = \begin{pmatrix} F_{11} & F_{12} & F_{13} & \cdot & \cdot & \cdot & \cdot F_{33} \end{pmatrix}^T$.

The least square solution for F is the eigenvector corresponding to the smallest eigenvalue of A, i.e. the last column of the matrix V in the *Singular Value Decomposition* (SVD), $A = UDV^T$. This gives the intermediate matrix F'. To obtain the correct rank 2 fundamental matrix, SVD of F' is done which is given by :

$$F = USV^T \tag{2.12}$$

Let the diagonal matrix obtained from the SVD is $S = diag(r, s, t)$, then the correct rank 2 fundamental matrix F is given by:

$$F = U \times diag(r, s, 0) \times V^T \tag{2.13}$$

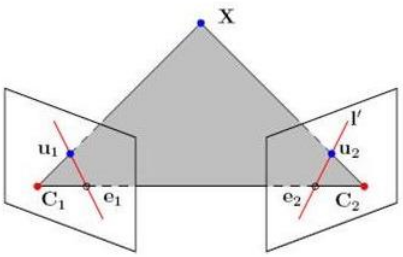

Figure 2.3: Epipolar Geometry

To summarize, the steps for generic 8 - Point Algorithm are listed below:

- For each matched point, add a new row to matrix A mentioned above.
- Find F', an intermediate value for F by taking the singular value decomposition (SVD) of A.
- Refine F' by taking SVD, setting the smallest diagonal value equal to zero and recombining.

2.4.4 Epipolar Geometry

The content of this section is largely referred from [8]. The epipolar geometry is motivated by stereo matching i.e. by searching for the projections of a scene point X in two different views of the same rigid scene. Let's suppose that $X \in \mathbb{R}^3$ projects onto $u_1 \in \mathbb{R}^2$ in the first view and onto $u_2 \in \mathbb{R}^2$ in the second. From the fundamental properties of central projection follows that the centers of the cameras C_1, C_2 and the points u_1, u_2 and X are coplanar. Scene points together with baseline $\overline{C_1C_2}$ create a group of planes called *Epipolar Planes*. Each of such planes intersects the projective planes of the two views in straight line called *Epipolar Line*. Fig. (2.3) illustrates the epipolar geometry.

Fig. (2.4) show the corner points and the epipolar lines.

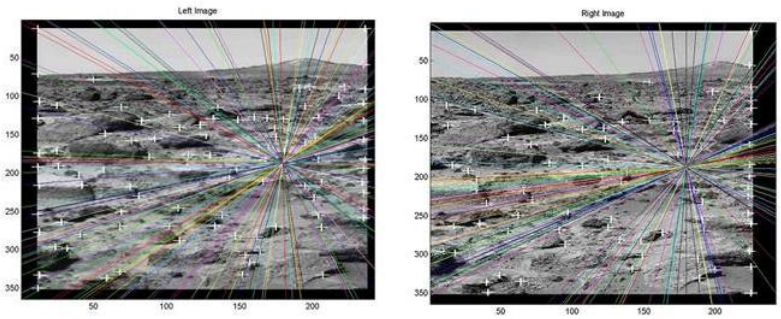

Figure 2.4: Corner Points and Epipolar Lines

Figure 2.5: Disparity Map

2.4.5 Feature Matching

The matching value of pixel p in one stereo image and pixel q in the other image is computed using the modified normalized cross - correlation with a window of suitable size defined by :

$$MatchingValue(p,q) = \frac{\sum_{p_i \in A_p, q_j \in A_q}[(I_{p_i} - \bar{I}_p) \times (I_{q_j} - \bar{I}_q)]}{[\sum_{p_i \in A_p}(I_{p_i} - \bar{I}_p)^2 \times \sum_{q_j \in A_q}(I_{q_j} - \bar{I}_q)^2]^{\frac{1}{2}}} \tag{2.14}$$

where A_p and A_q are windows of same size with centers p and q respectively, I_{p_i} and I_{q_j} are the corresponding pixel intensities of pixels p_i and q_j and $\bar{I}_p$ and $\bar{I}_q$ are the average pixel intensities in the two windows A_p and A_q respectively. The SSD (Sum of Squared Distance) algorithm calculates the sum of the squared distances between corresponding points in the stereo images which when minimized gives the matched points. The location of the best match can further be refined to subpixel

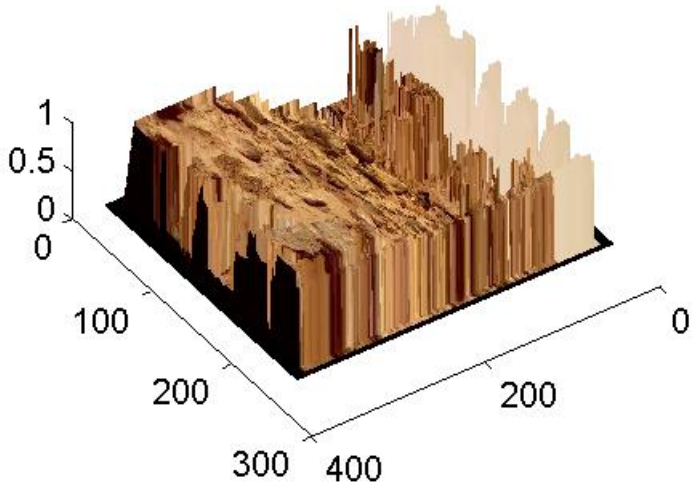

Figure 2.6: Depth Detection

accuracy by fitting a quadratic polynomial curve to the minimum SSD value and its two neighboring values. The location of the maximum value of the polynomial curve is the subpixel location. This can be implemented using *Hough Transform*.

2.4.6 Disparity Estimation and Depth Detection

The disparity values are then determined by computing the *Euclidean distance* between each pair of corresponding points. Finally assemble all the disparity values into a map, where pixel intensities describe the relative depth of points within a scene. Disparity Filling may be performed to estimate the Disparity values of unreliable pixels using both binocular and monocular information. This makes the Disparity image smoother. Fig. (2.5) shows the disparity map.

Chapter 3

Texture Segmentation

3.1 Introduction

Texture is the surface property of an object. In images, texture gives an idea about how a portion of the image is different from the other in terms of the following parameters :

- *RGB* value
- Grayscale value
- Noise content
- Luminance
- Geometry(edges)
- Wavelets

A dictionary-like definition of texture segmentation would be the following: "the partitioning of an image into regions, each of which contains a single texture distinct from its neighbors".
An image consists of an array of pixels, and we want to give each pixel a label. A region consists of a connected group of pixels that share the same label.
The texture segmentation system involves the following three steps:

- Decomposition of input image using a filter bank,
- Feature extraction,
- Clustering.

3.2 Decomposition of Input image Using a Filter Bank

A *filter bank* is an array of band-pass filters that separates the input signal into multiple components, each one carrying a single frequency subband of the original signal.

Here two - dimensional *Gabor* filters are used. A Gabor filter is a linear filter whose impulse response is defined by a harmonic function multiplied by a *Gaussian* function. Hence, in frequency domain Gabor filter's impulse function is the convolution of the *Fourier transform* of the harmonic function and the Fourier transform of the Gaussian function. The Gabor filter in the spatial domain is given by :

$$g_{\lambda\theta\psi\sigma\gamma}(x,y) = \exp(-\frac{x'^2+\gamma^2y'^2}{2\sigma^2})\cos(2\pi\frac{x'}{\gamma}+\psi) \tag{3.1}$$

where $x' = x\cos\theta + y\sin\theta$ and $y' = -x\cos\theta + y\sin\theta$.

In this equation λ represents the wavelength of the cosine factor, θ represents the orientation of the normal to the parallel stripes of a Gabor function in degrees, ψ is the phase offset in degrees, σ is the standard deviation of the Gaussian (which determines the linear size of the receptive field), γ is the spatial aspect ratio and specifies the ellipticity of the support of the Gabor function.

The parameter λ is the wavelength and $f = \frac{1}{\lambda}$ is the spatial frequency of the cosine factor. The ratio $\frac{\sigma}{\lambda}$ determines the spatial frequency bandwidth of simple cells. The half-response spatial frequency bandwidth b (in octaves) and the ratio $\frac{\sigma}{\lambda}$ are related as follows :

$$\begin{aligned} b &= \log_2 \frac{\frac{\sigma}{\lambda}+\sqrt{\frac{\ln 2}{2}}}{\frac{\sigma}{\lambda}-\sqrt{\frac{\ln 2}{2}}} \\ \frac{\sigma}{\lambda} &= \frac{1}{\pi}\sqrt{\frac{\ln 2}{2}}\frac{2^b+1}{2^b-1} \end{aligned}$$

We have taken $b = 1$ octave. $\psi = 0$ and $\psi = 90°$ returns the real part and the imaginary part of the Gabor filter respectively. We use the real part of the Gabor filter which is an even - symmetric filter.

3.3 Choice of Filter Parameters

θ is taken as :

$$\theta : 0^\circ, 30^\circ, 60^\circ, 90^\circ, 120^\circ, 150^\circ$$

The following values of frequencies are used :

$$\begin{aligned} F_L(i) &= 0.25 - \frac{2^{i-0.5}}{N_c} \\ F_H(i) &= 0.25 + \frac{2^{i-0.5}}{N_c} \quad i = 1, 2, \cdots, \log_2(\frac{N_c}{8}) \end{aligned}$$

where N_c is the width of image which is a power of 2. Note that $0 < F_L(i) < 0.25$ and $0.25 \leq F_H(i) < 0.5$.

3.4 Feature Extraction

Now, to compute features, each filtered image is subjected to a nonlinear transformation. Specifically, we use the following bounded nonlinearity :

$$\tanh(\alpha t) = \frac{1 - e^{-2\alpha t}}{1 + e^{-2\alpha t}} \tag{3.2}$$

Instead of identifying individual blobs and measuring their attributes, we simply compute the average absolute deviation (AAD) from the mean in small overlapping windows for each filtered image. This is similar to the *Texture Energy* measure. Formally, the feature image corresponding to each filtered image is given by :

$$e_k(x, y) = \frac{1}{M^2} \sum_{(a,b) \in Wxy} |\psi(r_k(a, b))| \tag{3.3}$$

We use Gaussian smoothing function which is given by :

$$g(x, y) = \exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{3.4}$$

We choose $\sigma = 3\sigma_s$, where σ_s is the scale parameter of Gabor filter as recommended in the literature [9].

Figure 3.1: Original Image

3.5 Clustering

The final step is to cluster the pixels into a number of clusters representing the texture regions. The *K-means algorithm* used to serve the purpose is described briefly in the following :

1. Initialize the *centroids* of K-clusters randomly.
2. Assign each sample to the nearest centroid.
3. Calculate centroids (means) of the K-clusters.
4. If centroids are unchanged, clustering is done. Otherwise go to step 2.

3.6 Experimental Results

The image segmentation system mentioned above was implemented and tested against textured image. the output result is as shown in Fig. (3.1). Fig. (3.2) shows the segmented image wrapped over the original image. Fig. (3.3) shows the $3-D$ view for different segments.

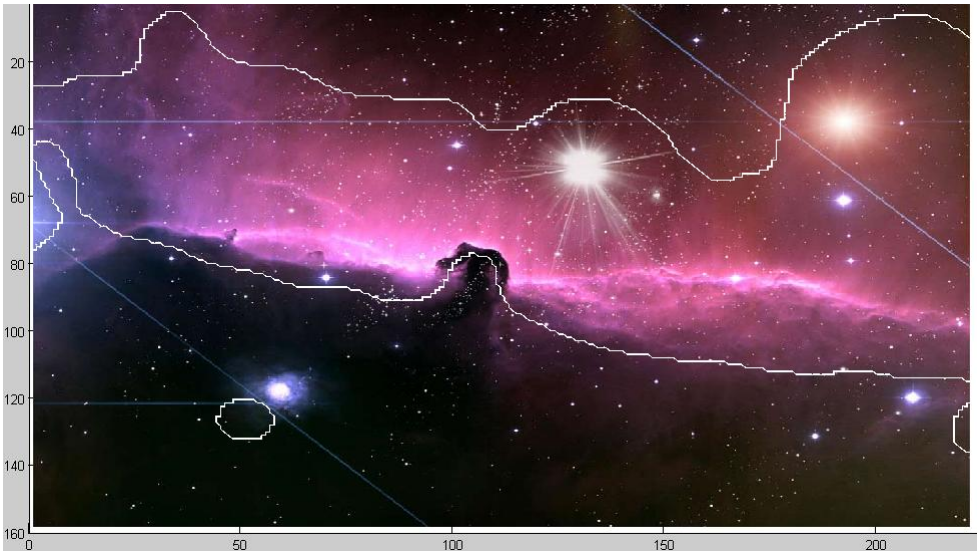

Figure 3.2: Segmented Image

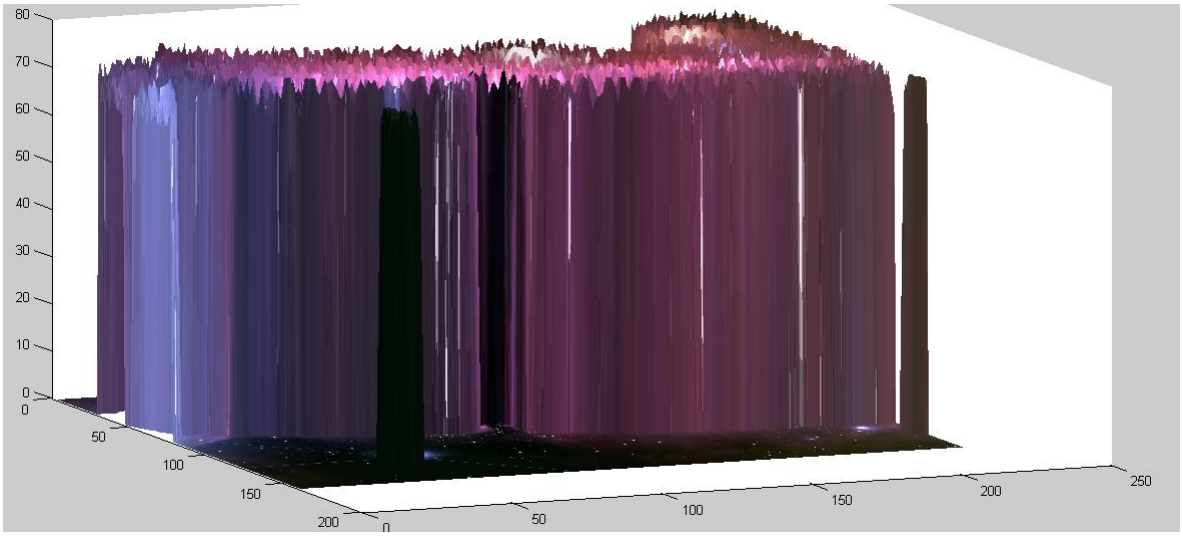

Figure 3.3: $3-D$ View of Segmented Image

Chapter 4

Haptics

4.1 Introduction

The science of sensing and manipulation through touch is called *Haptics*. Recent advances in science and technology allow a user to touch and feel a virtual object (the shape, geometry, roughness and other properties) using devices called Haptic Interfaces that generate forces. One such device called Novint Falcon is a part of the Haptics Lab, IIT Bombay.

4.2 Haptic Rendering

Haptic Rendering is defined as the process of converting the spatial information of the haptic interface device into force feedback.
Haptic information channel (Haptic Device) is bidirectional. The output of the chan-

Figure 4.1: Novint Falcon

Figure 4.2: Haptic Rendering

nel is dependent on the user input. The user input is in form of the spatial information of the haptic device (position of the control stick of Novint Falcon which the user is holding) and the scene (two dimensional bitmapped image) that is to be haptically felt/perceived. The output is in form of the forces fed back to the user through the control stick.

Two major tasks in haptic rendering are :

- Collision detection
- Collision response.

Collision detection is to detect collisions between the control stick of Novint Falcon and the objects in the image (scene). *Collision response* is to respond to the collision detection by feeding the forces back to the control stick. The magnitude and the angle of force fed are calculated by force rendering algorithm.

The generation of forces for Haptic Surface is based on two methods :

- **Depth Detection** : Generates a grayscale disparity matrix based on the depth of the objects in the image; protruding objects appear lighter and hence have high grayscale values and vice versa.
- **Image Segmentation/Texture Detection** : The partitioning of an image into regions, each of which contains a single texture distinct from its neighbors. In this project, we have used Gabor filter for texture segmentation.

4.3 Algorithm for Generation of Forces

- Generate a grayscale disparity matrix of the image by depth detection.

- Map the spatial dimensions ($M \times N$) of this matrix to the two dimensions XY (left-right, forward-backward) of the Novint Falcon workspace.
- The matrix contains the grayscale values of the image, which are mapped to the third dimension Z (up-down) of the Novint Falcon workspace.
- During runtime, detect the collision of the control stick of Novint Falcon which the user is holding, with the objects of the image through grayscale values.
- Send the force proportional to the grayscale value of the matrix at that position, to the control stick which the user is holding. This makes the user feel this force, which restricts him or her to penetrate into the Novint Falcon workspace in the Z direction; user hence feeling the surface.

Let,

- X be left-right coordinate of the Novint Falcon workspace,
- Y be forward-backward coordinate of the Novint Falcon workspace,
- Z be the up-down coordinate of the Novint Falcon Workspace,
- The size of the grayscale valued image matrix G be $M \times N$,
- G be the grayscale value vector containing $M \times N$ elements; $m \leq M$, $n \leq N$,
- $[Pos(1)\ Pos(2)\ Pos(3)]^T$ be the location of the control stick in the XYZ workspace during runtime at instant t.

When $Pos(1) = m$, $Pos(2) = n$ and $Pos(3) = G(m, n)$, the force proportional to gray scale value, $G(m, n)$ is fed to the control stick of Novint Falcon.
Hence, when the user touches a virtual object in the image, the haptic rendering algorithm generates adequate force to simulate the presence of the object and its properties like the shape, texture, etc.

4.4 Software Courtesy

- *Haptik Library* : An open source library that acts as a Hardware Abstraction Layer to provide access to haptic devices,
- Matlab.

Chapter 5

User Interface Logic

5.1 Objective

- Exciting yet *intuitive visualization* of the interaction between the physical and virtual world.
- *Immersible multisensory interaction* by captivating sense of touch and vision.
- Expressing *work flow* in an clean manner.
- *Falcon Emulation Mode* enables the developer/user to work without the Novint Falcon and use the mouse instead. Limited to two-dimensional interaction.
- Ability to log of events, errors, warnings and suppress them at will.

5.2 Features of the User Interface

The main features of the user interface are listed below :

- Mapping Falcon's workspace to Image workspace.
- **Depth Visualization** : Three-dimensional plots of disparity-maps [define] to visualize the depth of the rendered image.
- **Texture Visualization** : Three-dimensional plots of haptic texture surfaces.
- *Image Warps* to the three-dimensional terrain to deliver sufficient realism [Ref : matlab].

- *Realtime Cursor* for falcon that interprets Novint Falcon's physical position in the real world and plots a corresponding cursor in virtual three-dimensional world.
- **Collision Detection** visualization.
- Live visualization of *Force Feedback Vectors* at point of collision along with their magnitude and direction.
- Visualization of *Surface Normals* to facilitate understanding of force feedback over rough terrain.

5.3 Future Directions

- Data Cursor at bottom right of the graph, indicating collision and coordinates of different cursors along with feedback force.
- Better Lighting effects on graph.
- Auditory feedback.
- Enable Falcon Emulation Mode to go three-dimensional.

5.4 Front End Development Status

The respective figures (Fig. (5.1) and (5.2)) show the front end of the user interface and the surface with normals at every vertex.
Fig. (5.3) shows the $3-D$ cursor interaction.

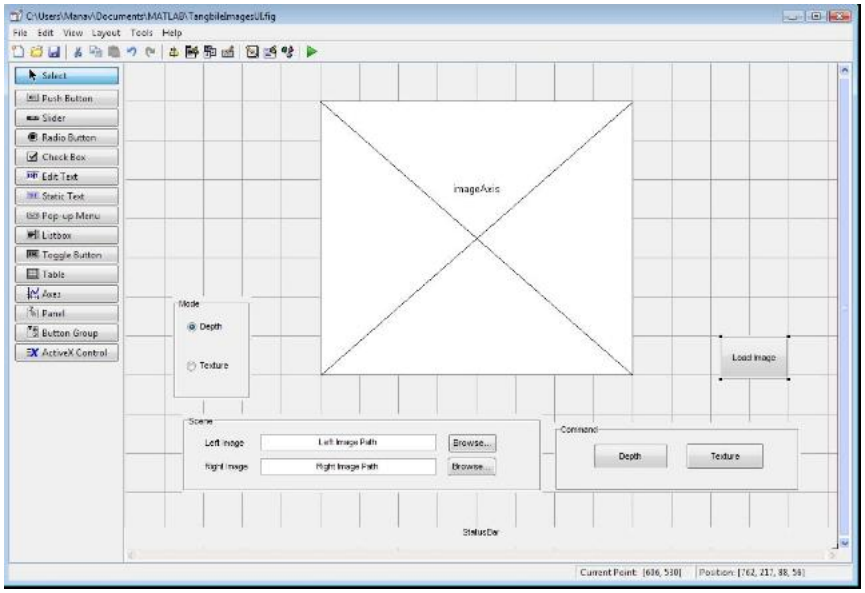

Figure 5.1: User Interface Front End

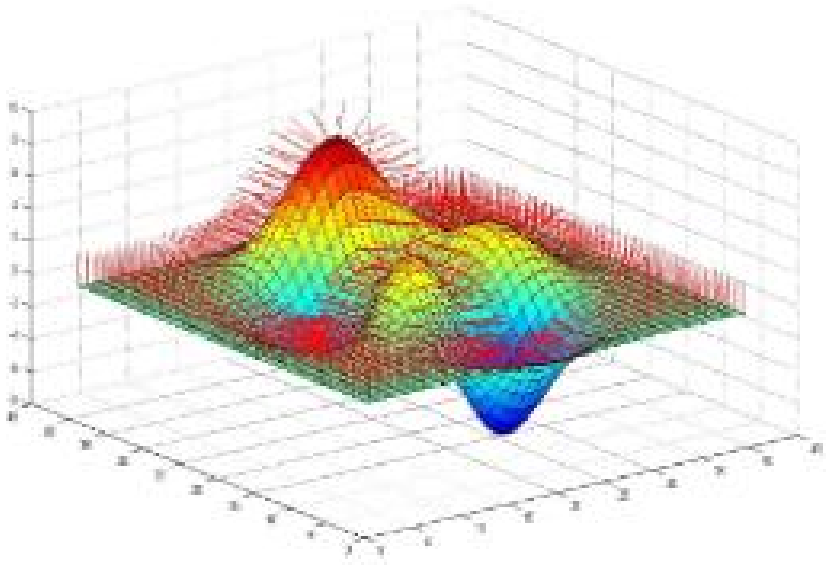

Figure 5.2: Surface with normals at every vertex

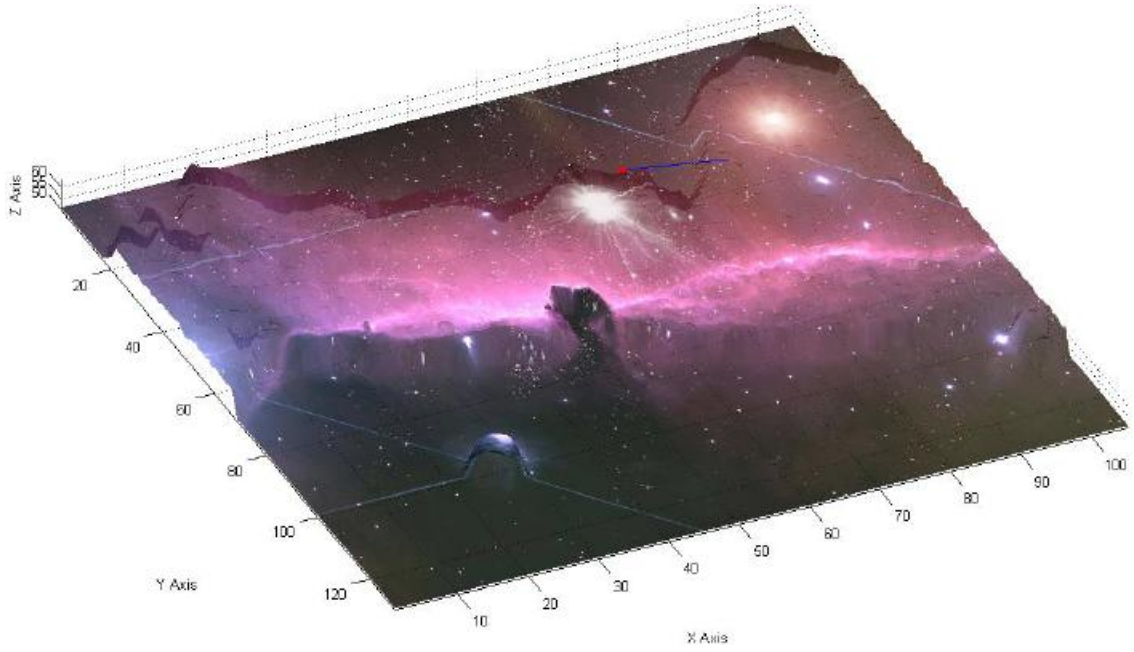

Figure 5.3: $3-D$ Cursor Interaction

Bibliography

[1] Vasudevan, H. and Manivannan, M. "Tangible Images : Runtime Generation of Haptic Textures From Images" for HAPTICS. IEEE Computer Society, Washington, DC. Proceedings of the 2008 Symposium on Haptic interfaces For Virtual Environment and Teleoperator Systems, 357-360 (March 13 - 14, 2008). DOI=http://dx.doi.org/10.1109/HAPTICS.2008.4479971

[2] Lakshmanan V., "A Heirarchical, Multiscale Texture Segmentation Algorithm for Real- World Scenes" PhD thesis, U. Oklahoma, Norman, OK, 2001

[3] de Pascale M. and Prattichizzo D. "The Haptik Library: A Component Based Architecture for Uniform Access to Haptic Devices" IEEE Robotics and Automation Magazine, vol. 14, no. 4, pp. 64-75, Dec. 2007

[4] Frucci M. et al. "Case-Based Reasoning for Image Segmentation by Watershed Transformation" Studies in Computational Intelligence, Volume 73/2008, 319-353

[5] Kataria M. and Tapudum "Haptics : The Story of Touch" to be published in Electrical Dept. Technical Newspaper, IIT Bombay, 2009-10

[6] Srinivasan, Mandayam A. "What is Haptics?" Touch Lab, Massachusetts institite of Technology

[7] Harris C., Stephens M. "A Combined Corner and Edge Detector", Alvey Vision Conf., pp. 147-151, 1988

[8] Heller J. "Stereo Reconstruction from Wide Angle Images", Master's Thesis, Charles University in Prague, 2007

[9] Jain A. K., Farrokhnia F. "Unsupervised Texture Segmentation using Gabor Filters" Pattern Recognition, vol. 24, no. 12, pp.1167-1186, 1991

[10] www.wikipedia.org