CHAPTER-1 INTRODUCTION

1.1 FOREWORD

There are many applications of 3D Reconstruction in today's world of technology. The 3-D information can be obtained from a pair of images, also known as a stereo pair, by estimating the relative depth of points in the scene. These estimates are represented in a stereo disparity map, which is constructed by matching corresponding points in the stereo pair. Using 3D reconstruction one can determine any object's 3D profile, as well as knowing the 3D coordinate of any point on the profile.The 3D reconstruction of objects is a generally scientific problem and core technology of a wide variety of fields, such as Computer Aided Geometric Design (CAGD), Computer Graphics, Computer Animation, Computer Vision, medical imaging, computational science, Virtual Reality, digital media, etc.

1.2 WORKFLOW DIAGRAM

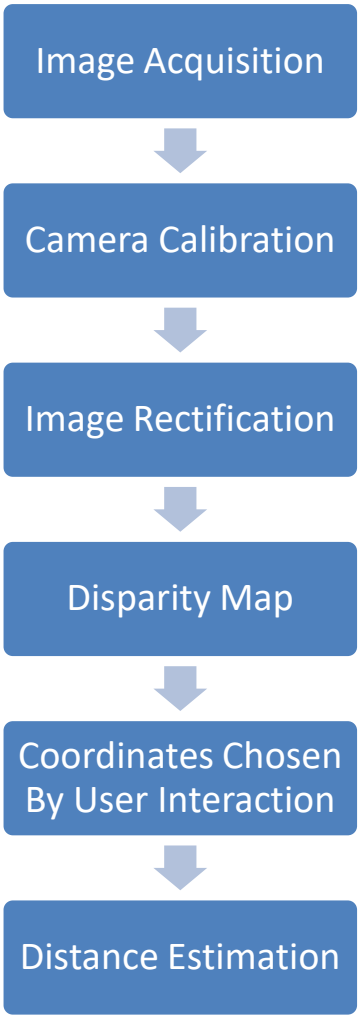


Figure 1

1.3 STEREO IMAGING

We all are familiar with the stereo imaging capability that our eyes give us. To what degree can we emulate this capability in computational systems? Computers accomplish this task by finding correspondences between points that are seen by one imager and the same points as seen by the other imager. With such correspondences and a known baseline separation between cameras, we can compute the 3D location of the points. Although the search for corresponding points can be computationally expensive, we can use our knowledge of the geometry of the system to narrow down the search space as much as possible. In practice, stereo imaging involves four steps when using two cameras.

1. Mathematically remove radial and tangential lens distortion; this is called distortion. The outputs of this step are undistorted images.

2. Adjust for the angles and distances between cameras, a process called rectification. The outputs of this step are images that are row-aligned and rectified.

3. Find the same features in the left and right camera views, a process known as correspondence. The output of this step is a disparity map, where the disparities are the differences in x-coordinates on the image planes of the same feature viewed in the left and right cameras: xl – xr.

4. If we know the geometric arrangement of the cameras, then we can turn the disparity map into distances by triangulation. This step is called reprojection , and the output is a depth map.

1.4 OPERATING ENVIRONMENT

Here we are going to develop a '3D Reconstruction using Stereo Imaging Application' using MATLAB software. MATLAB supports an object oriented programming language like C, C++.

1.4.1 WINDOWS

Microsoft Windows (or simply Windows) is a meta family of graphical operating systems developed, marketed, and sold by Microsoft.

Microsoft introduced an operating environment named *Windows* on November 20, 1985, as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984.

Windows10

On September 30, 2014, Microsoft announced Windows 10 as the successor to Windows 8.1. It was released on July 29, 2015, and addresses shortcomings in the user interface first introduced with Windows 8. Changes include the return of the Start Menu, a virtual desktop system, and the ability to run Windows Store apps within windows on the desktop rather than in full-screen mode. Windows 10 is said to be available to update from qualified Windows 7 with SP1 and Windows 8.1 computers from the Get Windows 10 Application (for Windows 7, Windows 8.1) or Windows Update (Windows 7).

On November 12, 2015, an update to Windows 10, version 1511, was released.[1] This update can be activated with a Windows 7, 8 or 8.1 product key as well as Windows 10 product keys. Features include new icons and right-click context menus, default printer management, four times as many tiles allowed in the Start menu, Find My Device, and Edge updates.

1.4.2 MATLAB

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

The MATLAB platform is optimized for solving engineering and scientific problems. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. It is used for machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, robotics, and much more.

Figure 2

1.4.3 SYSTEM CONFIGURATION

The Face Detection Program using Feature Extraction and Neural Networks has been tested on:

- Windows 10
- Processor: Intel I3
- Installed Memory (RAM) : 4 GB (3.68 GB usable)
- System Type : 64-bit Operating System, x64-based processor
- MATLAB 2014a

1.5 RATIONALE FOR USING MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include the following:

- math and computation
- algorithm development
- data acquisition
- modelling, simulation, and prototyping
- data analysis, exploration, and visualization
- scientific and engineering graphics
- application development, including graphical user interface building
-

1.5.1 IMAGE PROCESSING IN MATLAB

Here we are going to explain an introduction on how to handle images in MATLAB. When working with images in MATLAB, there are many things to keep in mind such as loading an image, using the right format, saving the data as different data types, how to display an image, conversion between different image formats, etc. This worksheet presents some of the commands designed for these operations. Most of these commands require you to have the image processing tool box installed with MATLAB.

To find out if it is installed, type ver at the MATLAB prompt. This gives you a list of what tool boxes that are installed on your system. An image may be defined as a two-dimensional function f(x, y), where x and y are spatial coordinates, and the amplitude of at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y, and the amplitude values of are all finite, discrete quantities, we call the image a digital image. The field of digital image processing refers to processing digital images by means of a digital computer.

Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term most widely used to denote the elements of a digital image.

1.5.2 IMAGE FORMATS SUPPORTED BY MATLAB

The following image formats are supported by MATLAB: png,bmp, hdf, jpeg, pcx, tiff, xwb. Most images found on the internet are jpeg images which is the name for one of the most widely used compression standards for images. As jpeg format is lossy compression and png format is lossless compression ,that's why we prefer png format. An image named "myimage.png" is stored in the png format and we will see later on that we can load an image of this format into MATLAB as well as store the image taken by stereo camera in this format.

CHAPTER -2 IMAGE ACQUISITION BY STEREO CAMERA

2.1 INTRODUCTION

2.1.1 STEREO CAMERA

A **stereo camera** is a type of camera with two or more lenses with a separate image sensor or film frame for each lens. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images, a process known as stereo photography. Stereo cameras may be used for making stereo views and 3D pictures for movies, or for range imaging. The distance between the lenses in a typical stereo camera (the intra-axial distance) is about the distance between one's eyes (known as the intra-ocular distance) and is about 6.35 cm, though a longer base line (greater inter-camera distance) produces more extreme 3-dimensionality.

In the 1950s, stereo cameras gained some popularity with the Stereo Realist and similar cameras that employed 135 film to make stereo slides.

3D pictures following the theory behind stereo cameras can also be made more inexpensively by taking two pictures with the same camera, but moving the camera a few inches either left or right. If the image is edited so that each eye sees a different image, then the image will appear to be 3D. This method has problems with objects moving in the different views, though works well with still life.

Stereo cameras are sometimes mounted in cars to detect the lane's width and the proximity of an object on the road.

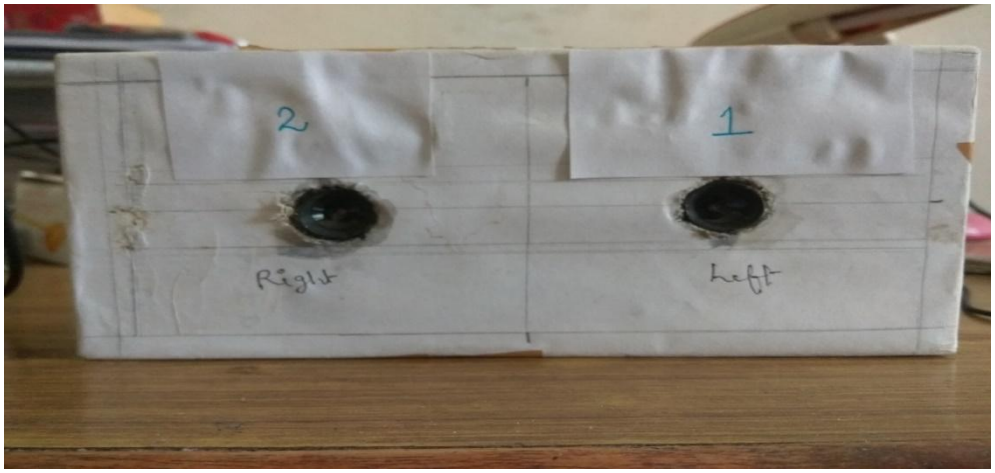


Figure 3

Features

- Image Sensor : CMOS Sensor
- Image Resolution : 2 Mega Pixels
- Anti-Flicker : 50Hz
- Image Quality : RGB24 or I420
- Exposure : Auto or Manual
- Angle of View : 58degree
- Interface : USB 2.0
- Frame Rate : 30fps
- Lens : f=6.0

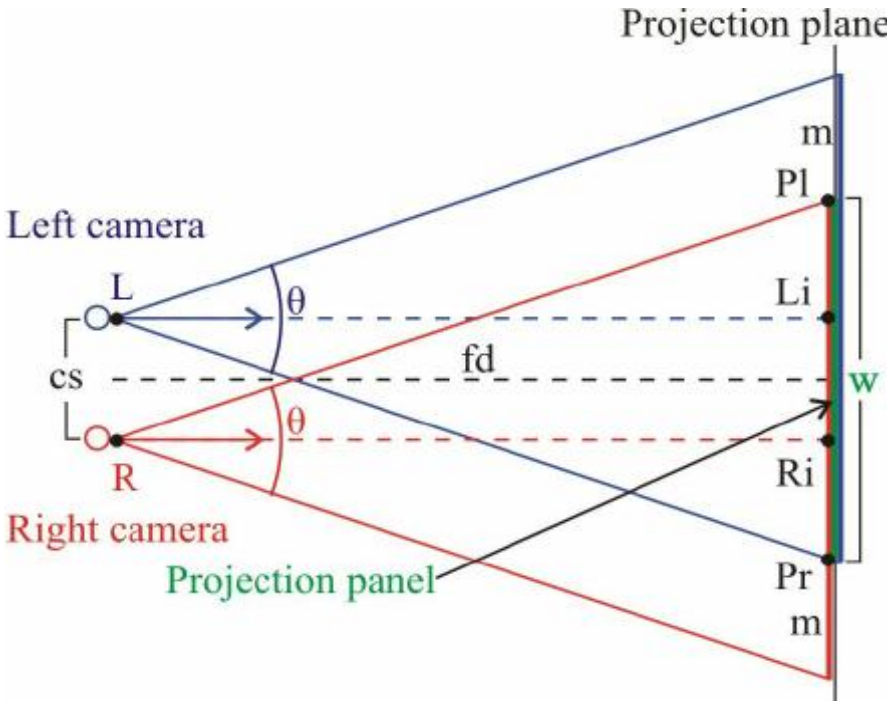


Figure 4

2.2 IMAGE ACQUISITION

There is a slight variation in the relative position of the objects in the left and right view of the same object as taken by the stereo camera. The stereo camera lens work just like our eye. The distance between the lens is approximately equal to the distance present in the left and right eye. The image acquisition by both the lens can be seen clearly in the figure below:

Fig5(a) :In this figure the image is shown of chessboard which is captured by the left lens of the camera.

Fig5(b) :In this figure the image is shown of the same chessboard which is captured by the right lens of the camera.

Similarly Fig6(a) and Fig6(b) represents the left and right image of the plant as taken by left and right lens of the camera.

Figure 5(a)

Figure 5(b)

Figure 6(a)

Figure 6(b)

CHAPTER - 3 STEREO CAMERA CALIBRATION

3.1 GOAL

In this section,

- We will learn about distortions in camera, intrinsic and extrinsic parameters of camera etc.
- We will learn to find these parameters, undistort images etc.

3.2 CALIBRATION

Geometric camera calibration, also referred to as *camera resectioning*, estimates the parameters of a lens and image sensor of an image or video camera. You can use these parameters to correct for lens distortion, measure the size of an object in world units, or determine the location of the camera in the scene. These tasks are used in applications such as machine vision to detect and measure objects. They are also used in robotics, for navigation systems, and 3-D scene reconstruction. Camera parameters include intrinsics, extrinsics, and distortion coefficients. To estimate the camera parameters, you need to have 3-D world points and their corresponding 2-D image points. You can get these correspondences using multiple images of a calibration pattern, such as a checkerboard. Using the correspondences, you can solve for the camera parameters. After you calibrate a camera, to evaluate the accuracy of the estimated parameters, you can:

- Plot the relative locations of the camera and the calibration pattern
- Calculate the reprojection errors.
- Calculate the parameter estimation errors.

3.3 CAMERA MODEL

The Computer Vision System Toolbox™ calibration algorithm uses the camera model proposed by Jean-Yves Bouguet . The model includes:

- The pinhole camera model .

- Lens distortion .

  The pinhole camera model does not account for lens distortion because an ideal pinhole camera does not have a lens. To accurately represent a real camera, the full camera model used by the algorithm includes the radial and tangential lens distortion.

3.3.1 PINHOLE CAMERA MODEL

A pinhole camera is a simple camera without a lens and with a single small aperture. Light rays pass through the aperture and project an inverted image on the opposite side of the camera. Think of the virtual image plane as being in front of the camera and containing the upright image of the scene.

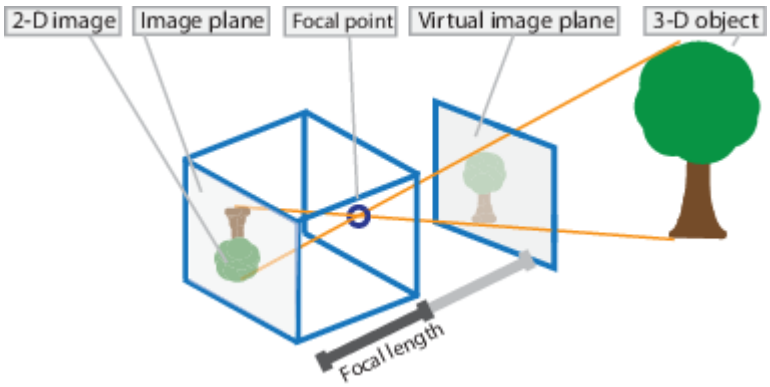


Figure 7

The pinhole camera parameters are represented in a 4-by-3 matrix called the *camera matrix*. This matrix maps the 3-D world scene into the image plane. The calibration algorithm calculates the camera matrix using the extrinsic and intrinsic parameters. The extrinsic parameters represent the location of the camera in the 3-D scene. The intrinsic parameters represent the optical center and focal length of the camera.

$$w\,[x\ y\ 1] = [X\ Y\ Z\ 1]\,P$$

Scale factor — Image points — World points

$$P = \begin{bmatrix} R \\ t \end{bmatrix} K$$

Camera matrix — Extrinsics: Rotation and translation — Intrinsic matrix

The world points are transformed to camera coordinates using the extrinsics parameters. The camera coordinates are mapped into the image plane using the intrinsics parameters.

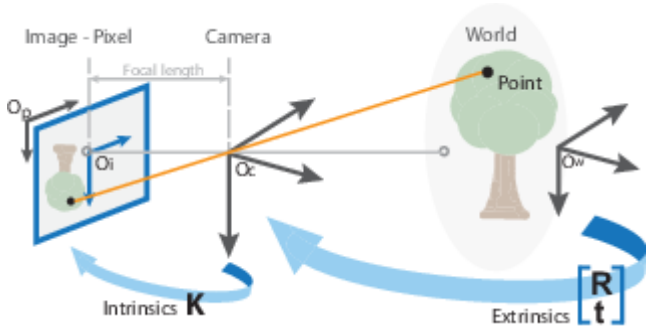


Figure 8

3.4 CAMERA CALIBRATION PARAMETERS

The calibration algorithm calculates the camera matrix using the extrinsic and intrinsic parameters. The extrinsic parameters represent a rigid transformation from 3-D world coordinate system to the 3-D camera's coordinate system. The intrinsic parameters represent a projective transformation from the 3-D camera's coordinates into the 2-D image coordinates.

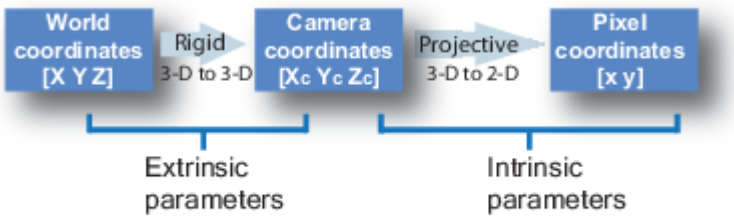


Figure9

Extrinsic Parameters

The extrinsic parameters consist of a rotation, *R*, and a translation, *t*. The origin of the camera's coordinate system is at its optical center and its *x*- and *y*-axis define the image plane.

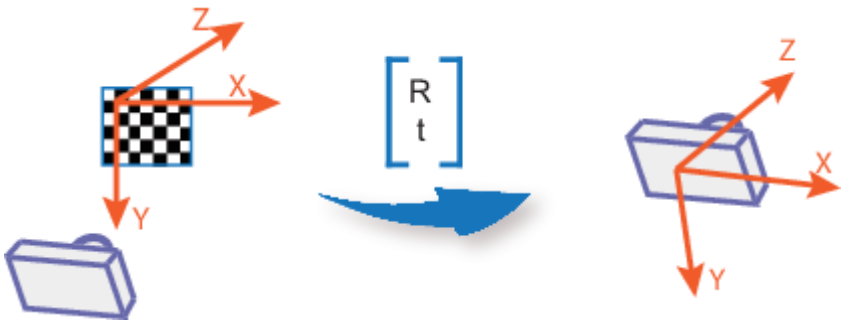


Figure10

Intrinsic Parameters

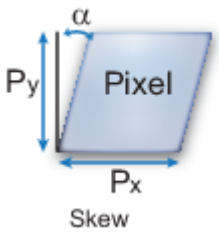
The intrinsic parameters include the focal length, the optical center, also known as the *principal point*, and the skew coefficient. The camera intrinsic matrix, *K*, is defined as:

$[f_x \; 0 \; 0]$

$[s \; f_y \; 0]$

$[c_x \; c_y \; 1]$

The pixel skew is defined as:



3.5 DISTORTION IN CAMERA CALIBRATION

The camera matrix does not account for lens distortion because an ideal pinhole camera does not have a lens. To accurately represent a real camera, the camera model includes the radial and tangential lens distortion.

Radial Distortion

Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center. The smaller the lens, the greater the distortion.

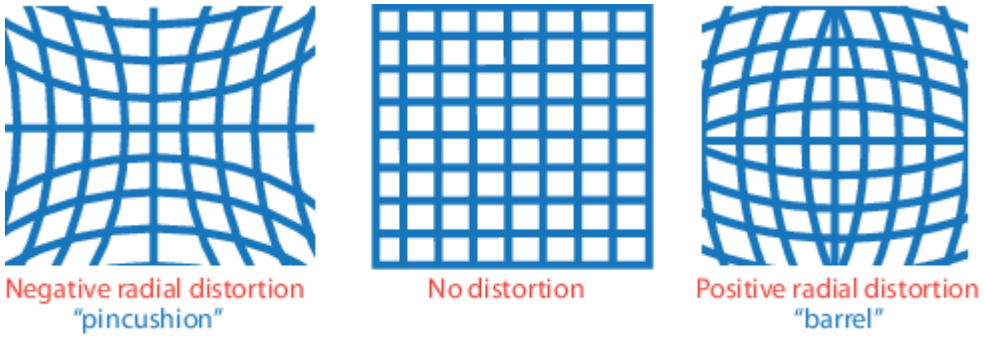


Figure 11

The radial distortion coefficients model this type of distortion. The distorted points are denoted as $(x_{distorted}, y_{distorted})$:

$$x_{distorted} = x(1 + k_1*r^2 + k_2*r^4 + k_3*r^6)$$

$$y_{distorted} = y(1 + k_1*r^2 + k_2*r^4 + k_3*r^6)$$

- x, y — Undistorted pixel locations. x and y are in normalized image coordinates. Normalized image coordinates are calculated from pixel coordinates by translating to the optical center and dividing by the focal length in pixels. Thus, x and y are dimensionless.
- k_1, k_2, and k_3 — Radial distortion coefficients of the lens.
- r^2: $x^2 + y^2$

Typically, two coefficients are sufficient for calibration. For severe distortion, such as in wide-angle lenses, you can select 3 coefficients to include k_3.

Tangential Distortion

Tangential distortion occurs when the lens and the image plane are not parallel. The tangential distortion coefficients model this type of distortion.

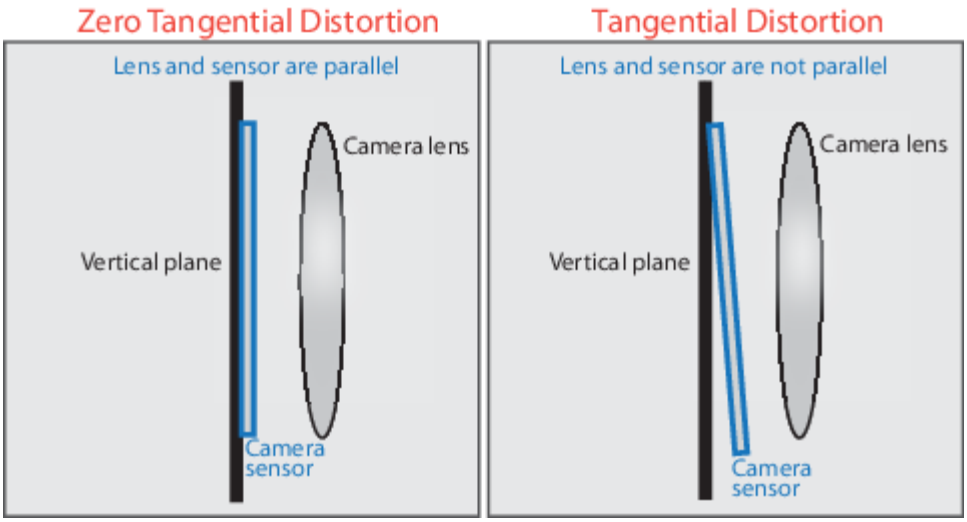


Figure 12

The distorted points are denoted as ($x_{distorted}$, $y_{distorted}$):

$$x_{distorted} = x + [2 * p_1 * x * y + p_2 * (r^2 + 2 * x^2)]$$

$$y_{distorted} = y + [p_1 * (r^2 + 2 * y^2) + 2 * p_2 * x * y]$$

- x, y — Undistorted pixel locations. x and y are in normalized image coordinates. Normalized image coordinates are calculated from pixel coordinates by translating to the optical center and dividing by the focal length in pixels. Thus, x and y are dimensionless.
- p_1 and p_2 — Tangential distortion coefficients of the lens.
- $r^2 = x^2 + y^2$
-

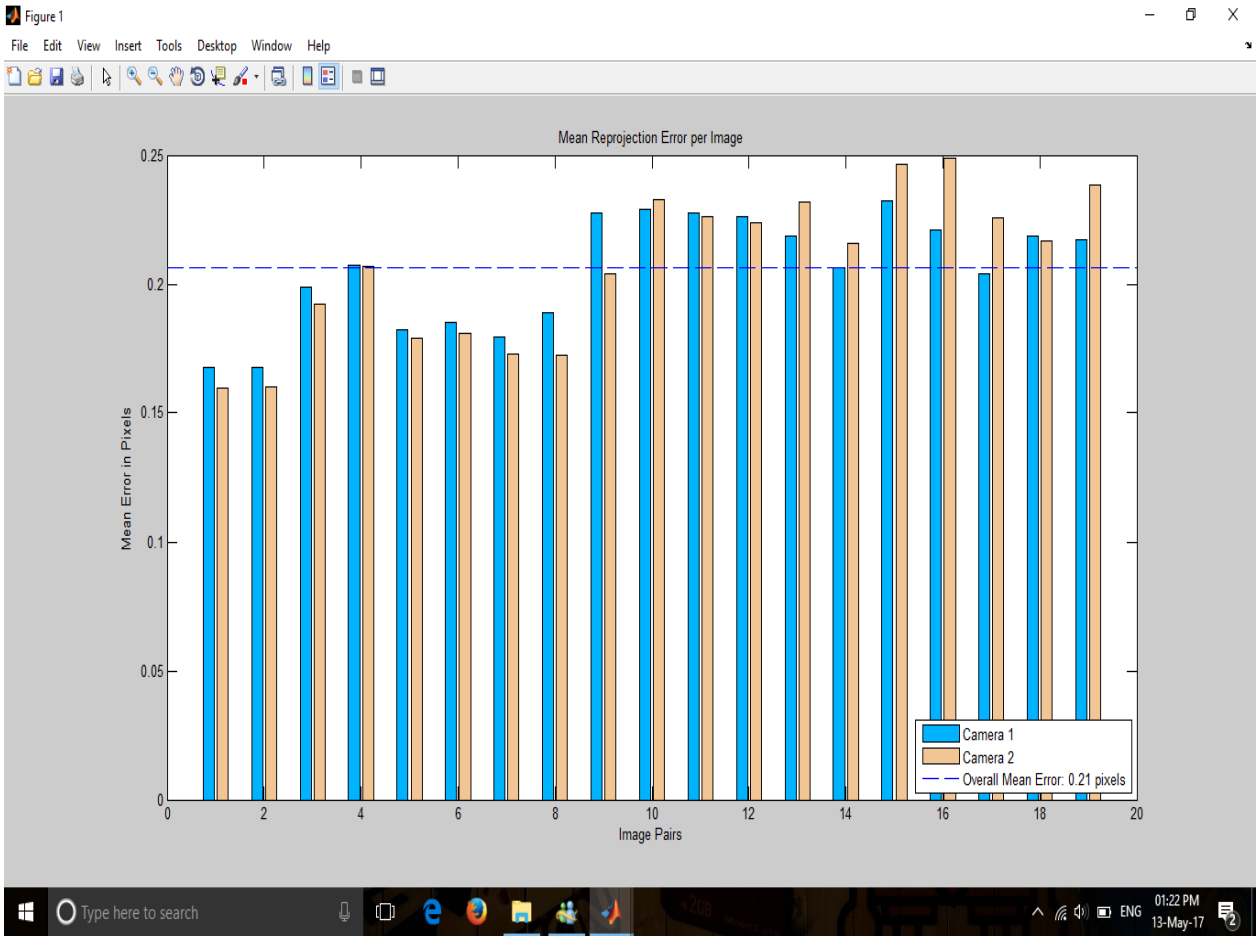
3.6 CODE

As mentioned above, we need atleast 10 test patterns for camera calibration. SO we have come up with some images of chess board (see samples/cpp/left01.jpg -- left14.jpg), so we will utilize it. For sake of understanding, consider just one image of a chess board. Important input datas needed for camera calibration is a set of 3D real world points and its corresponding 2D image points. 2D image points are OK which we can easily find from the image. (These image points are locations where two black squares touch each other in chess boards)

What about the 3D points from real world space? Those images are taken from a static camera and chess boards are placed at different locations and orientations. So we need to know (X, Y, Z)

values. But for simplicity, we can say chess board was kept stationary at XY plane, (so Z=0 always) and camera was moved accordingly. This consideration helps us to find only X,Y values. Now for X,Y values, we can simply pass the points as (0,0), (1,0), (2,0), ... which denotes the location of points. In this case, the results we get will be in the scale of size of chess board square. But if we know the square size, (say 30 mm), and we can pass the values as (0,0),(30,0),(60,0),..., we get the results in mm. (In this case, we don't know square size since we didn't take those images, so we pass in terms of square size).

3D points are called object points and 2D image points are called image points.



CHAPTER-4 RECTIFICATION

Image rectification is a transformation process used to project two-or-more images onto a common image plane. This process has several degrees of freedom and there are many strategies for transforming images to the common plane.

- It is used in computer stereo vision to simplify the problem of finding matching points between images (i.e. the correspondence problem).
- It is used in geographic information systems to merge images taken from multiple perspectives into a common map coordinate system.

The objective of stereo rectification is to make the corresponding epipolar lines of image pairs be parallel to the horizontal direction, so that the efficiency of stereo matching is improved as the corresponding points stay in the same horizontal lines of both images. In this paper，a simple and convenient rectification method of calibrated image pairs based on geometric transformation is proposed, which can avoid the complicated calculation of many previous algorithms such as based on epipolar lines, based on fundamental matrix or directly depend on corresponding points. This method is divided into two steps including coordinate system transformation and re-projection of image points. Firstly, we establish two virtual cameras with parallel optical axis by coordinate system transformation based on the pose relationship of the two cameras from calibration result. Secondly, we re-project the points of the original image onto new image planes of the virtual cameras through geometrical method, and then realized the stereo rectification. Experiments of real stereo image pairs show that the proposed method is able to realize the rectification of stereo image pairs accurately and efficiently.

4.1 EPIPOLAR GEOMETRY

Image rectification methods have been known for long by photogrammetrists and have been improved later by software approaches. Most of this early methods involved to know the camera projection matrices. Then, this constraint has been relaxed with methods taking advantage of

epipolar geometry to align the two images.The main issue of image rectification is that the rectified pair is not unique. Most of existing methods deal with how to find an image rectification that minimizes the image distortion.

Here we attempt to reduce the amount of distortion by finding the rectification transform that is closest to preserving orthogonality about the image centers. However, orthogonality is not an adequate criterion since even an Euclidean image rectification can involve a loss of orthogonality. Correcting this non-orthogonality might decrease the Euclidean property of the rectified images. So, a linear method is proposed that minimizes the horizontal parallax among the point correspondences used for the rectification. Decomposition of the rectification process is done into affine and projective components.Also propose a method that prevent the rectification process from a distortion on a selected common plane specified from 3 points correspondence on the two images. Also a method is proposed to find the transformation that best preserves the sampling of the original stereo pair, i.e. each pixel in the unrectified image should map to a single pixel in the rectified image. Now we optimize each transformation in order to minimize perspective distortions, such that the rectified images look like the original images as closely as possible. We perform an image rectification by three successive rotations. Do not explicitly compute epipolar geometry but generate a rectifying pair of homographies that are are conform to the fundamental matrix form of a rectified image pair.Finally a rectification method based on a reprojection onto a cylindrical surface instead of a plane in order to reach an optimal pixel distribution on the rectified images to avoid any pixel loss. All these methods minimize an image distortion and thus are well suited for depth from stereo methods but not for stereoscopic rendering since they do not consider the camera transformation. In other words, there is no guarantee to obtain a pair of rectified images that corresponds or is close to an Euclidean camera setup. Moreover, most of these methods are based on epipolar geometry and hence cannot be directly extended to handle more than two views.

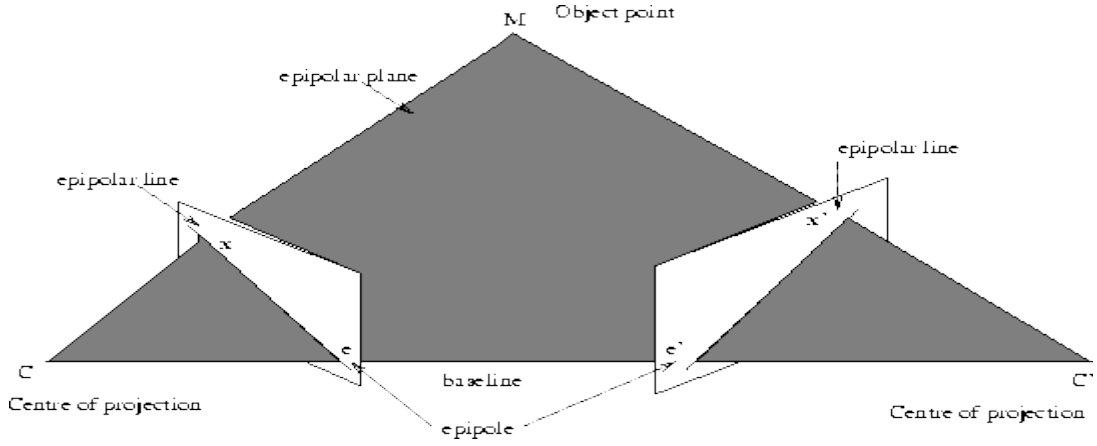


Figure 13

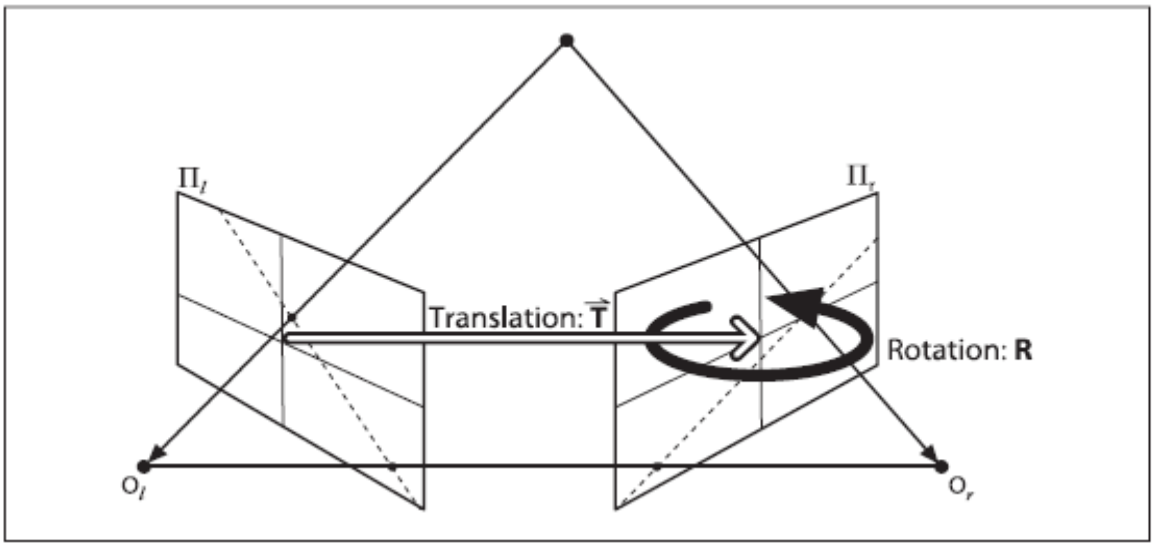


Figure 14

Figure 12-9. Th e essential geometry of stereo imaging is captured by the essential matrix E, which contains all of the information about the translation T and the rotation R, which describe the locationof the second camera relative to the fi rst in global coordinates

Th e matrix E contains information about the translation and rotation that relate the two cameras in physical space (see Figure 12-9), and F contains the same information as E in addition to information about the intrinsics of both cameras. Because F embeds information about the intrinsic parameters, it relates the two cameras in pixel coordinates.

The essential matrix E is purely geometrical and knows nothing about imagers. It relates the location, in physical coordinates,of the point P as seen by the left camera to the location of the same point as seen by the right camera (i.e., it relates pl to pr). The fundamental matrix F relates the points on the image plane of one camera in image coordinates (pixels) to the points on the image plane of the other camera in image coordinates

4.2 SYNTAX

```
[J1,J2] = rectifyStereoImages(I1,I2,stereoParams)
[J1,J2] = rectifyStereoImages(I1,I2,tform1,tform2)
[J1,J2] = rectifyStereoImages(___,interp)
[J1,J2] = rectifyStereoImages(___,Name,Value)
```

4.3 DESCRIPTION

[J1,J2] = rectifyStereoImages(I1,I2,stereoParams) returns undistorted and rectified versions of I1 and I2 input images using the stereo parameters stored in the stereoParams object.

Stereo image rectification projects images onto a common image plane in such a way that the corresponding points have the same row coordinates. This image projection makes the image appear as though the two cameras are parallel. Use the disparity function to compute a disparity map from the rectified images for 3-D scene reconstruction.

[J1,J2] = rectifyStereoImages(I1,I2,tform1,tform2) returns rectified versions of I1 and I2 input images by applying projective transformations tform1 and tform2. The projective transformations are returned by the estimateUncalibratedRectification function.

[J1,J2] = rectifyStereoImages(___,interp) additionally specifies the interpolation method to use for rectified images. You can specify the method as 'nearest', 'linear', or 'cubic'.

[J1,J2] = rectifyStereoImages(___,Name,Value) uses additional options specified by one or more Name,Value pair arguments

4.4 EXAMPLE

From the dataset taken by our stereo camera, an image is selected. Firstly, camera calibration is

being done and after that its rectification. After rectification the vertical shift in the left and the right image of the same object/scene as taken by the left and right lens of the stereo camera respectively. It can be seen clearly in the picture given below:

Figure 15:Before Rectification :

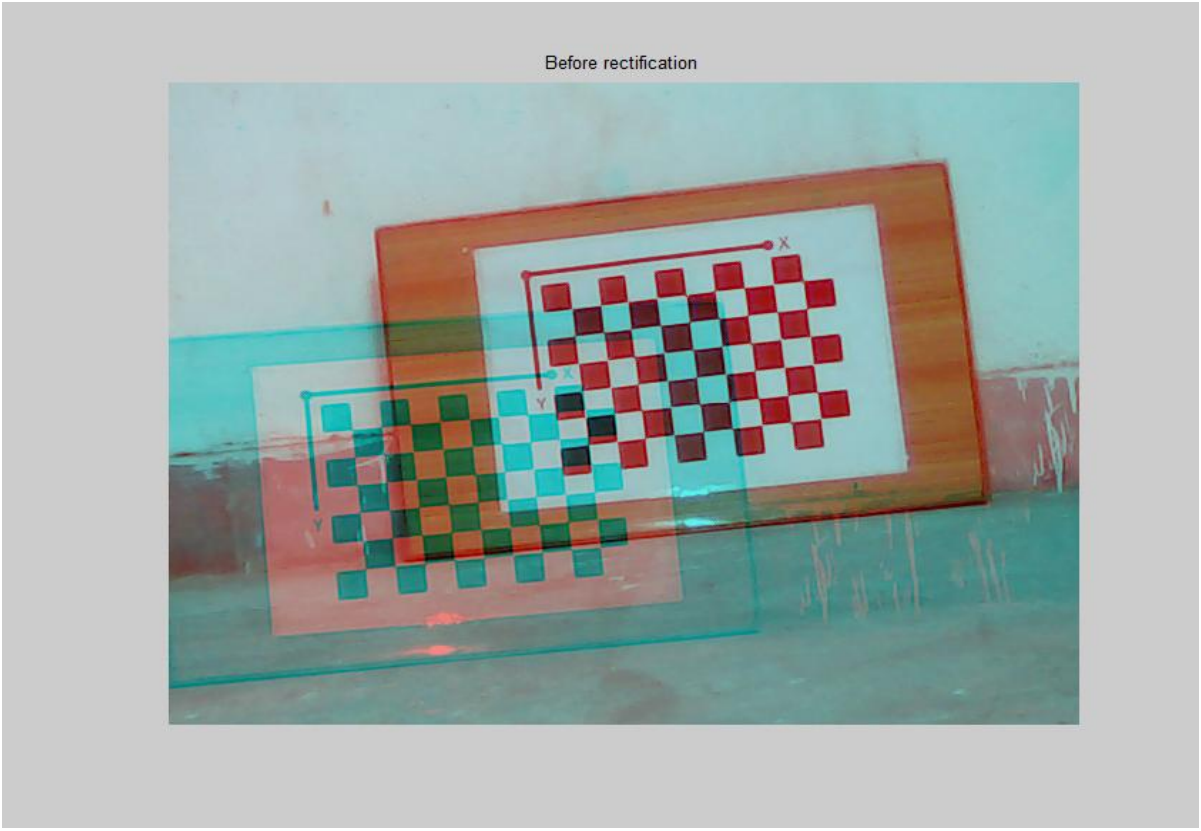
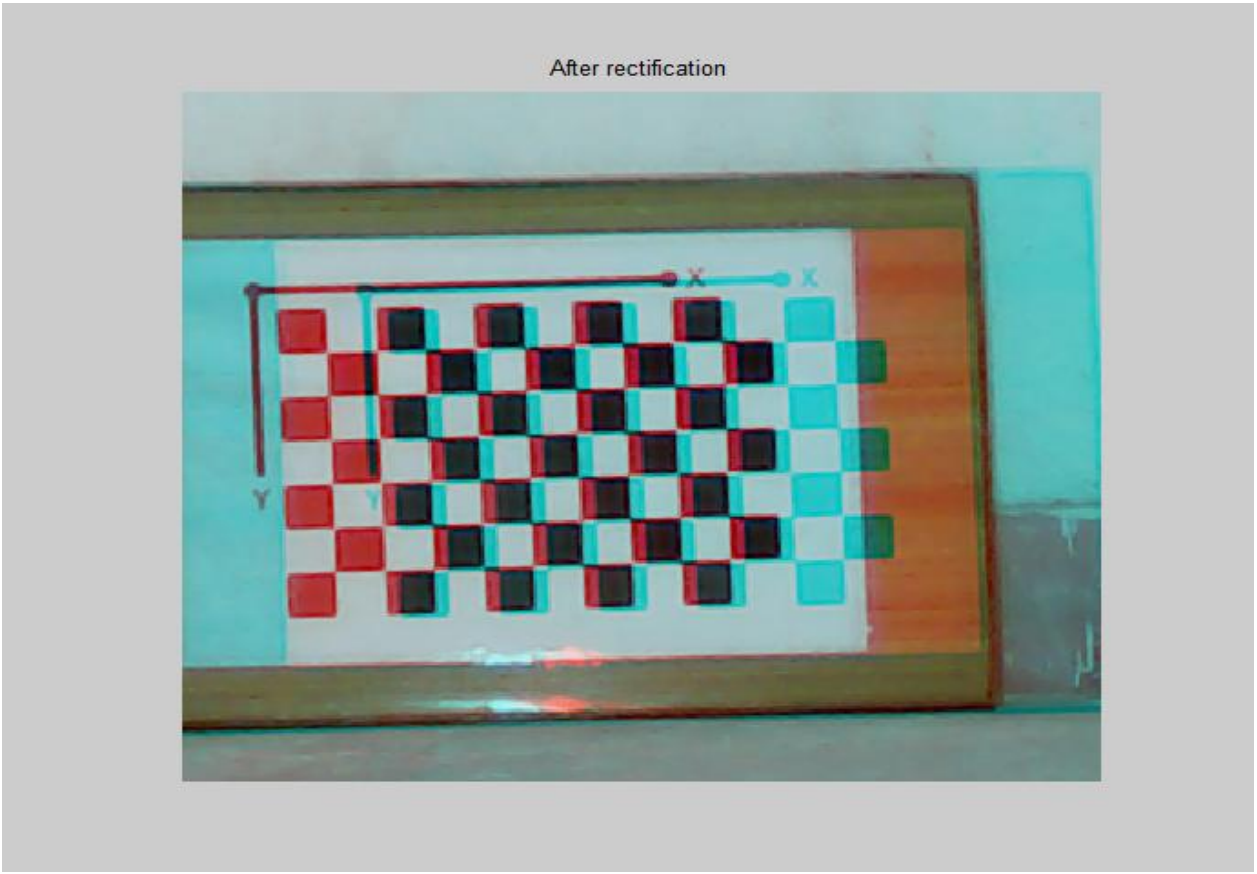


Figure 16:After Rectification :



CHAPTER-5 DISPARITY MAP

5.1 OVERVIEW

Disparity map refers to the apparent pixel difference or motion between a pair of stereo image. To experience this, try closing one of your eyes and then rapidly close it while opening the other. Objects that are close to you will appear to jump a significant distance while objects further away will move very little. That motion is the disparity.

In a pair of images derived from stereo cameras, you can measure the apparent motion in pixels for every point and make an intensity image out of the measurements.

5.2 TRIANGULATION

Triangulation in stereo analysis is the task of computing the 3D position of points in the images, given the disparity map and the geometry of the stereo setting.

5.2.1 CAMERA MODEL

The pinhole camera model is often used for 3D reconstruction

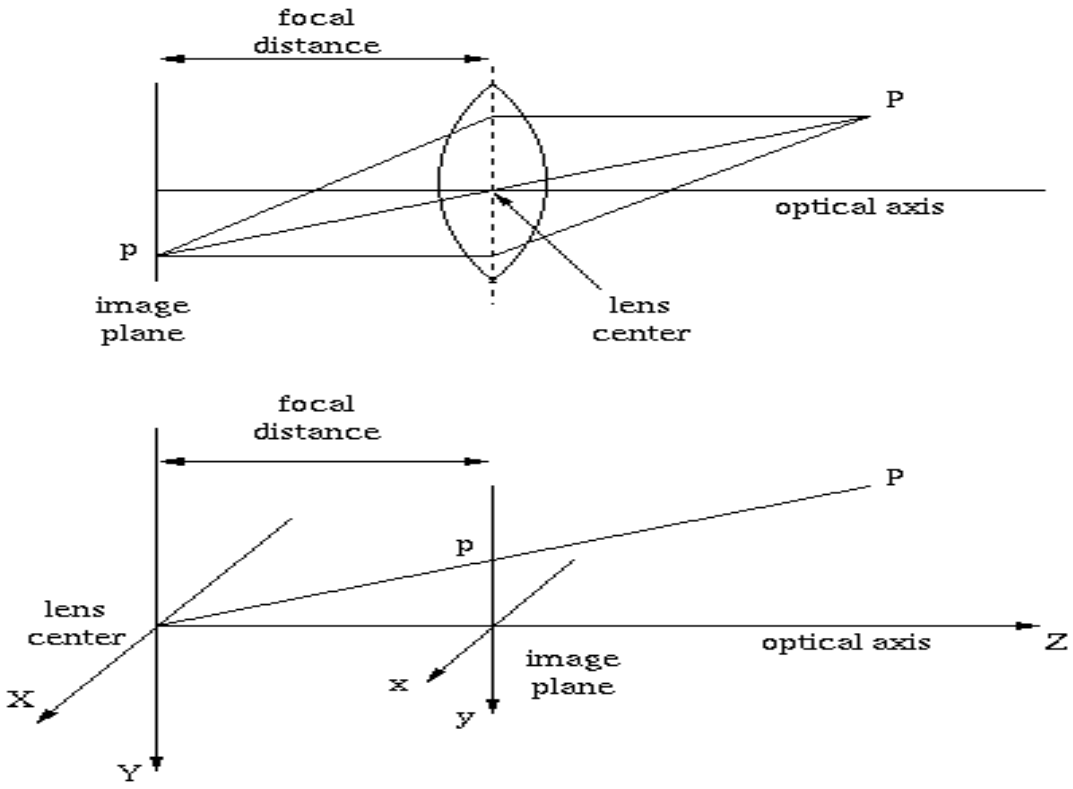


Figure 17

The relation between the world coordinates of a point P(X,Y,Z) and the coordinates on the image plane (x,y) in a pinhole camera is

$x = f * X / Z$

$y = f * Y / Z$

where f is the focal distance of the lens.

5.2.2 FROM FRAME TO IMAGE COORDINATES

A digitalized image is usually stored in a frame buffer, that can be seen as a matrix of pixels with W columns and H rows. Let (i,j) be the discrete frame coordinates of the image with origin in the upperleft corner, (Ox,Oy) be the focal point of the lens (the intersection between the optical axes and the image plane) in the frame coordinates, and (x,y) be the image coordinates.

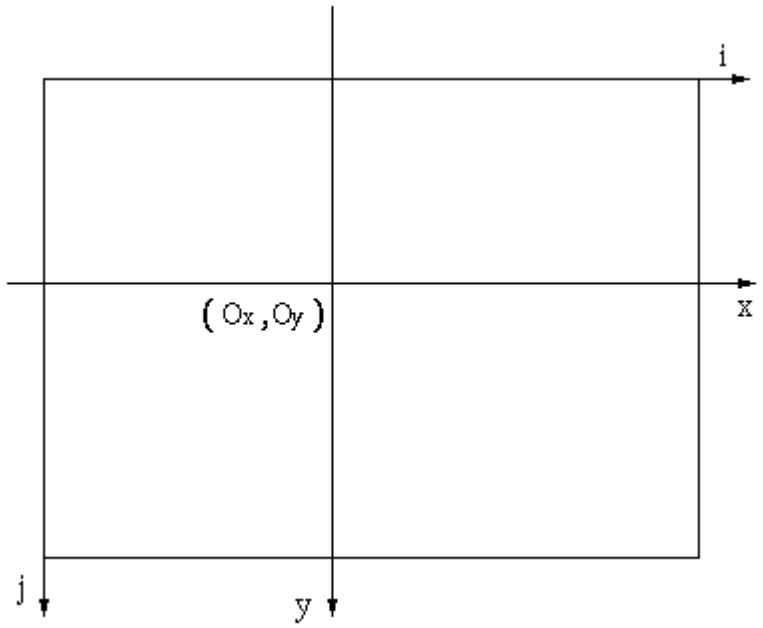


Figure 18

Image coordinates relate to frame coordinates in the following way:

x = (i - Ox) * Sx

y = (j - Oy) * Sy

where Sx,Sy are the horizontal and vertical distances of two adjacent pixels in the frame buffer.

5.2.3 RADIAL DISTORTION

Lens distortion can be modelled by the following equations:

x = xd * (1 + k1 * rd^2)

y = yd * (1 + k1 * rd^2)

where (xd,yd) are the image coordinates of the distorted image, rd = SQRT(xd^2 + yd^2), and k1 is a constant depending on the distortion of the lens.

The relation between image and frame coordinates in presence of lens distortion is:

x = (id - Ox) * Sx * (1 + k1 * rd^2)

y = (jd - Oy) * Sy * (1 + k1 * rd^2)

5.2.4 SCANNING AND SAMPLING

If the distances of two adjacent cells in the camera digitalization device are known (Px,Py), then the distances of pixels in the image are

Sx = ax * Px

Sy = Py

where ax is a scale factor due to the displacement in horizontal scanning and sampling frequencies. On the contrary, there is no displacement in vertical sampling.

5.3 TRIANGULATION PROCESS

The 3D position (X,Y,Z) of a point P, can be reconstructed from the perspective projection of P on the image planes of the cameras, once the relative position and orientation of the two cameras are known.

We choose the 3D world reference system to be the left camera reference system. The right camera is translated and rotated with respect from the left one, therefore six parameters describe this transformation.
The simplest case arise when the optical axes of two cameras are parallel, and the translation of the right camera is only along the X axis.

5.3.1 GEOMETRY FOR PARALLEL CAMERAS

Let us consider the optical setting in the figure, that is also called *standard model*.

1. L and R are two pinhole cameras with parallel optical axes. Let f be the focal length of both cameras.
2. The baseline (that is the line connecting the two lens centers) is perpendicular to the optical axes. Let b be the distance between the two lens centers.
3. XZ is the plane where the optical axes lie, XY plane is parallel to the image plane of both cameras, X axis equals the baseline and the origin O of (X,Y,Z) world reference system is the lens center of the left camera.

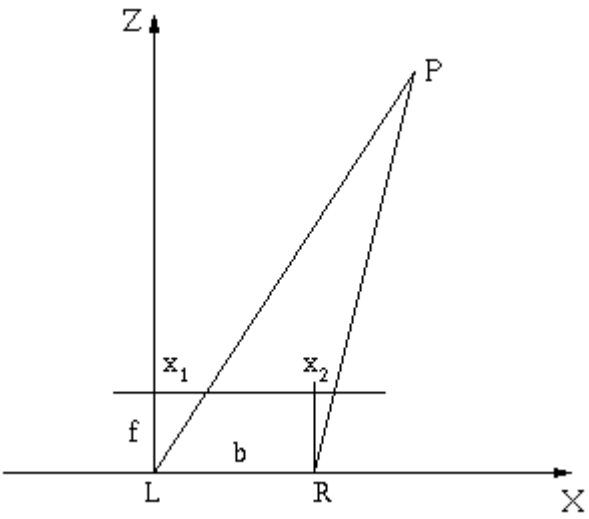


Figure 19

In this setting the equations of stereo triangulation are:

Z = (b * f) / (x1 - x2)

X = x1 * Z / f

Y = y1 * Z / f

5.3.2 NON-PARALLEL CAMERAS

In the general case, the right camera can be rotated with respect to the left one in three directions. In all this section we assume that rotation angles are small (*small angle approximation*).

Rotation around Y axis (*theta*)

In this case the optical axes are not parallel, but they both lie on the XZ plane, so they intersect in a point (0,0,Z0), that is called *fixation point* and could also be behind the cameras (Z0<0). If *theta* is the rotation angle, then Z0 = b / tan(*theta*).

Under small angle approximation, we can still assume the right image plane to be parallel to the left image plane and hence to XY plane.

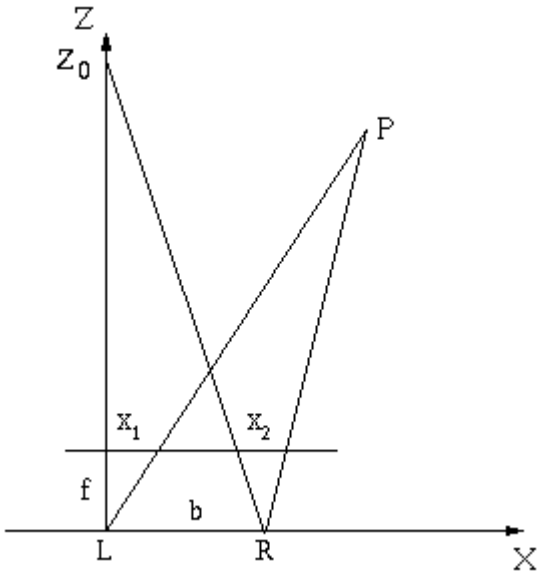


Figure 20

In this case we have:

Z = (b * f) / (x1 - x2 + f * b / Z0)

X = x1 * Z / f

Y = y1 * Z / f

Rotation around X axis (*phi*)

Rotation around X axis only affects the Y coordinate in reconstruction. Let *phi* be the rotation angle, then stereo triangulation is

Z = (b * f) / (x1 - x2)

X = x1 * Z / f

Y = y1 * Z / f + tan(*phi*) * Z

Rotation around Z axis (*psi*)

Rotation around optical axis is usually dealt with by rotating the image before applying matching and triangulation.
In the following the rotation angle of the right camera around its optical axis will be called *psi*.

5.4 GENERAL CASE

Given the translation vector T and rotation matrix R describing the transormation from left camera to right camera coordinates, the equation to solve for stereo triangulation is

p' = RT (p - T)

where p and p' are the coordinates of P in the left and right camera coordinates respectively, and RT is the transpose (or the inverse) matrix of R.

5.5 SYNTAX

disparityMap = disparity(I1,I2)

d = disparity(I1,I2,Name,Value)

5.6 DESCRIPTION

disparityMap = disparity(I1,I2): returns the disparity map, disparityMap, for a pair of stereo images, I1 and I2.

d = disparity(I1,I2,Name,Value): Additional control for the disparity algorithm requires specification of parameters and corresponding values. One or more Name,Value pair arguments specifies an additional option.

5.7 EXAMPLE

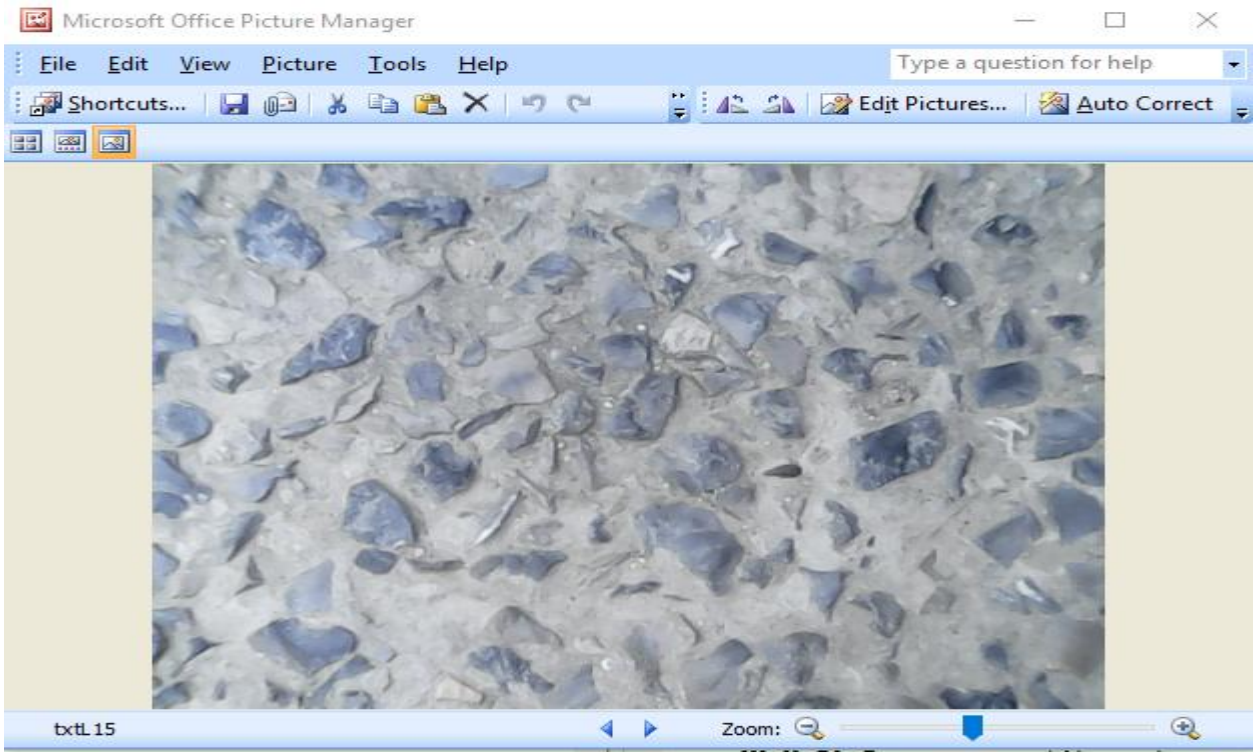


Figure 21(a)

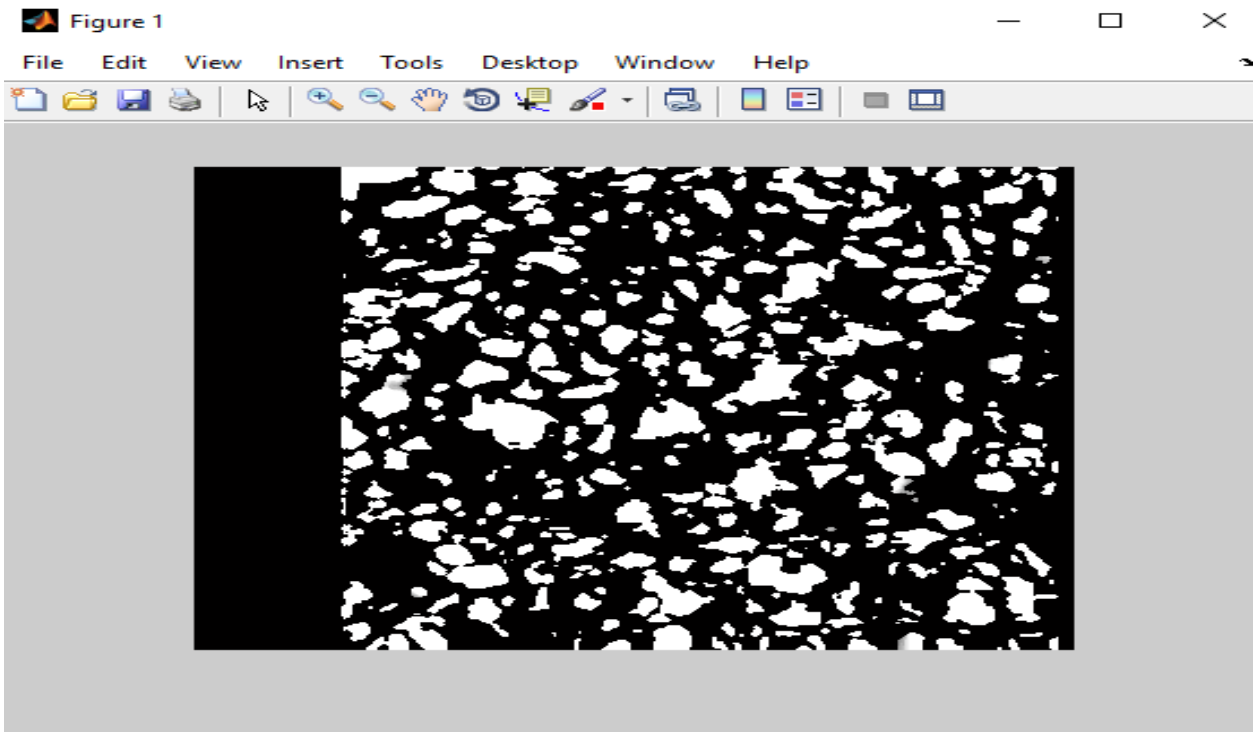


Figure 21(b)

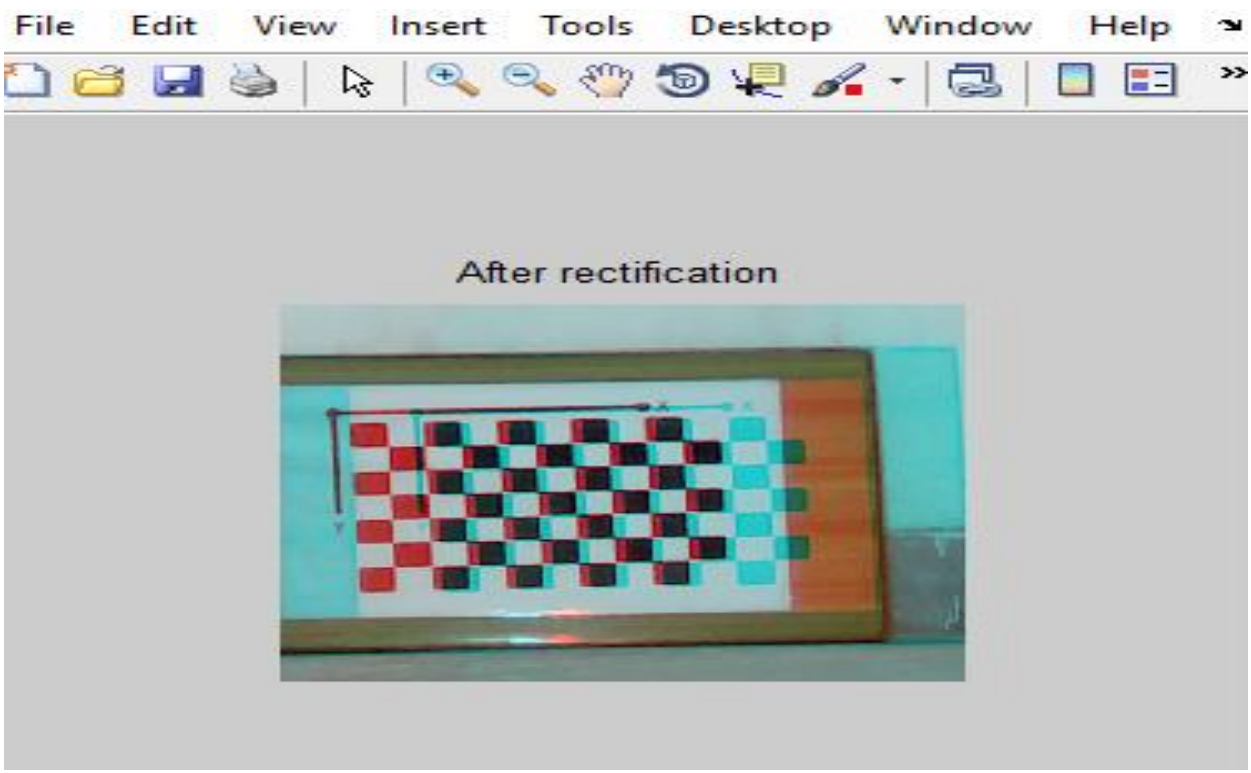


Figure 22(a)

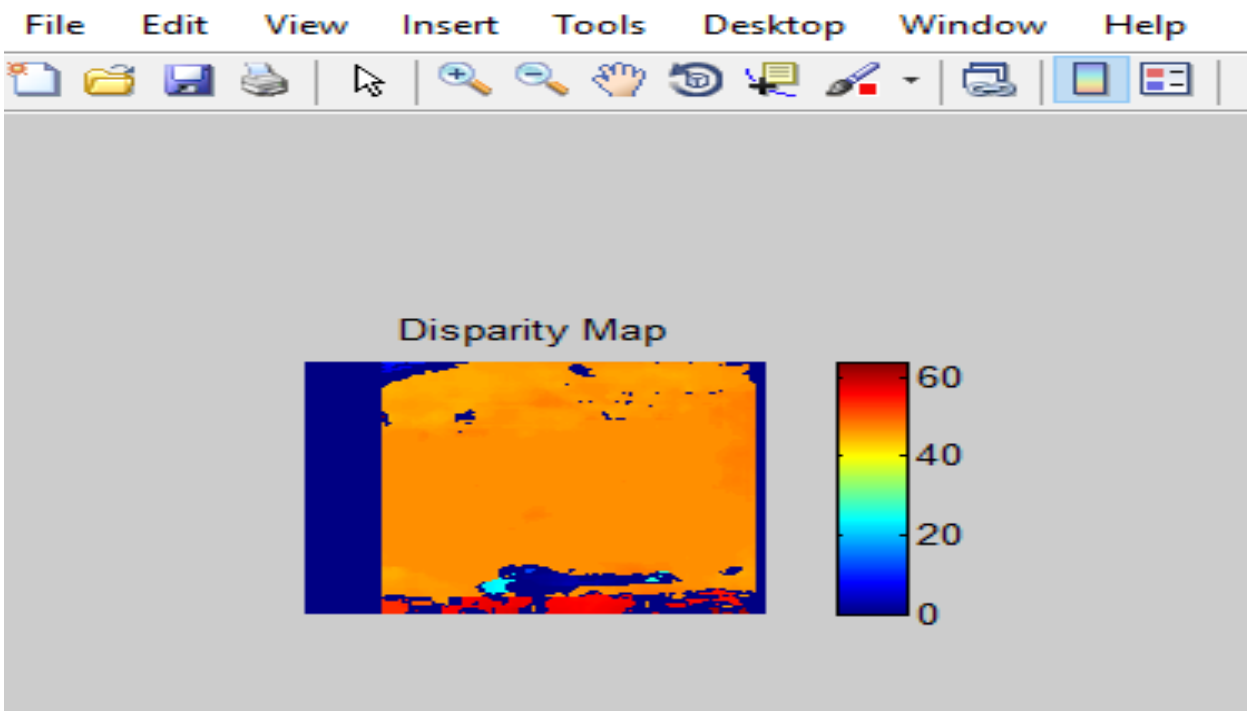


Figure 22(b)

In the figures shown above the disparity map of the pebbles is shown. It can be seen clearly as the pebbles are at a certain height from the level of land ;so the land in background is of black colour and the pebbles are shown in white colour Given below is the black and white disparity map (in Fig(b)) of the image as shown in Fig (a). Closer the object to the camera ,higher is its intensity and vice-versa.

CHAPTER-6 IMAGE RECONSTRUCTION

6.1 INTRODUCTION

In computer vision and computer graphics, **3D reconstruction** is the process of capturing the shape and appearance of real objects. This process can be accomplished either by active or passive methods. If the model is allowed to change its shape in time, this is referred to as non-rigid or spatio-temporal reconstruction.

6.2 MOTIVATION AND APPLICATIONS

The research of 3D reconstruction has always been a difficult goal. Using 3D reconstruction one can determine any object's 3D profile, as well as knowing the 3D coordinate of any point on the profile.The 3D reconstruction of objects is a generally scientific problem and core technology of a wide variety of fields, such as Computer Aided Geometric Design (CAGD), Computer Graphics, Computer Animation, Computer Vision, medical imaging, computational science, Virtual Reality, digital media, etc. For instance, the lesion information of the patients can be presented in 3D on the computer, which offers a new and accurate approach in diagnosis and thus has vital clinical value.

3D reconstruction system finds its application in a variety of field they are

- Medicine
- Film industry
- Robotics
- City planning
- Gaming
- Virtual environment
- Earth observation
- Archaeology
- Augmented reality
- Reverse engineering

- Animation
- Human computer interaction

6.3 ACTIVE METHODS

Active methods, i.e. range data methods, given the depth map, reconstruct the 3D profile by numerical approximation approach and build the object in scenario based on model. These methods actively interfere with the reconstructed object, either mechanically or radiometrically using rangefinders, in order to acquire the depth map, e.g. structured light, laser range finder and other active sensing techniques. A simple example of a mechanical method would use a depth gauge to measure a distance to a rotating object put on a turntable. More applicable radiometric methods emit radiance towards the object and then measure its reflected part. Examples range from moving light sources, colored visible light, time-of-flight lasers to microwaves or ultrasound. See 3D scanning for more details.

6.4 PASSIVE METHODS

Passive methods of 3D reconstruction do not interfere with the reconstructed object; they only use a sensor to measure the radiance reflected or emitted by the object's surface to infer its 3D structure through image understanding.[2] Typically, the sensor is an image sensor in a camera sensitive to visible light and the input to the method is a set of digital images (one, two or more) or video. In this case we talk about image-based reconstruction and the output is a 3D model. By comparison to active methods, passive methods can be applied to a wider range of situations.

6.4.1 MONOCULAR CUES METHODS

Monocular cues methods refer to use image (one, two or more) from one viewpoint (camera) to proceed 3D construction. It makes use of 2D characteristics(e.g. Silhouettes, shading and texture) to measure 3D shape, and that’s why it is also named Shape-From-X, where X can be silhouettes, shading, texture etc. 3D reconstruction through monocular cues is simple and quick, and only one appropriate digital image is needed thus only one camera is adequate. Technically, it avoids stereo correspondence, which is fairly complex.

Shape-from-shading Due to the analysis of the shade information in the image, by using Lambertian reflectance, the depth of normal information of the object surface is restored to reconstruct.

Photometric Stereo This approach is more sophisticated than the shape-of-shading method. Images taken in different lighting conditions are used to solve the depth information. It is worth mentioning that more than one image is required by this approach.

Shape-from-texture Suppose such an object with smooth surface covered by replicated texture units, and its projection from 3D to 2D causes distortion and perspective. Distortion and perspective measured in 2D images provide the hint for inversely solving depth of normal information of the object surface.

6.4.2 BINOCULAR STEREO VISION

Binocular Stereo Vision obtains the 3-dimensional geometric information of an object from multiple images based on the research of human visual system.[6] The results are presented in form of depth maps. Images of an object acquired by two cameras simultaneously in different viewing angles, or by one single camera at different time in different viewing angles, are used to restore its 3D geometric information and reconstruct its 3D profile and location. This is more direct than Monocular methods such as shape-from-shading.

Binocular stereo vision method requires two identical cameras with parallel optical axis to observe one same object, acquiring two images from different points of view. In terms of trigonometry relations, depth information can be calculated from disparity. Binocular stereo vision method is well developed and stably contributes to favorable 3D reconstruction, leading to a better performance when compared to other 3D construction. Unfortunately, it is computationally intensive, besides it performs rather poorly when baseline distance is large.

6.4.3 PROBLEM STATEMENT AND BASICS

The approach of using Binocular Stereo Vision to acquire object’s 3D geometric information is on the basis of visual disparity.[7] The following picture provides a simple schematic diagram of horizontally sighted Binocular Stereo Vision, where b is the baseline between projective centers of two cameras.

The origin of the camera's coordinate system is at the optical center of the camera's lens as shown in the figure. Actually, the camera's image plane is behind the optical center of the camera's lens. However, to simplify the calculation, images are drawn in front of the optical center of the lens by f. The u-axis and v-axis of the image's coordinate system O1uv are in the same direction with x-axis and y-axis of the camera's coordinate system respectively. The origin of the image's coordinate system is located on the intersection of imaging plane and the optical axis. Suppose such world point P whose corresponding image points are P1(u1,v1) and P2(u2,v2) respectively on the left and right image plane. Assume two cameras are in the same plane, then y-coordinates of P1 and P2 are identical, i.e.,v1=v2. According to trigonometry relations,

$$u_1 = f\frac{x_p}{z_p}$$

$$u_2 = f\frac{x_p - b}{z_p}$$

$$v_1 = v_2 = f\frac{y_p}{z_p}$$

where(x_p, y_p, z_p) are coordinates of P in the left camera's coordinate system, f is focal length of the camera. Visual disparity is defined as the difference in image point location of a certain world point acquired by two cameras,

$$d = u_1 - u_2 = f\frac{b}{z_p}$$

based on which the coordinates of P can be worked out.

Therefore, once the coordinates of image points is known, besides the parameters of two cameras, the 3D coordinate of the point can be determined

$$x_p = \frac{bu_1}{d}$$

$$y_p = \frac{bv_1}{d}$$

$$z_p = \frac{bf}{d}$$

6.5 EXPLANATION

The point cloud is generated for a rectified image here. The system detects a number of points to estimate depth at various coordinates i.e. the z-coordinate of(x,y,z) real world coordinates. The textured image will detect a large number of points whereas a whole plane surface like that of a chessboard or a flat wall will detect only few points as a single point is able to calculate the depth at every image point. The point cloud generated can be rotated 3-dimensionally. It can be seen clearly in the figures given below:

Figure 23(a) : Original Image

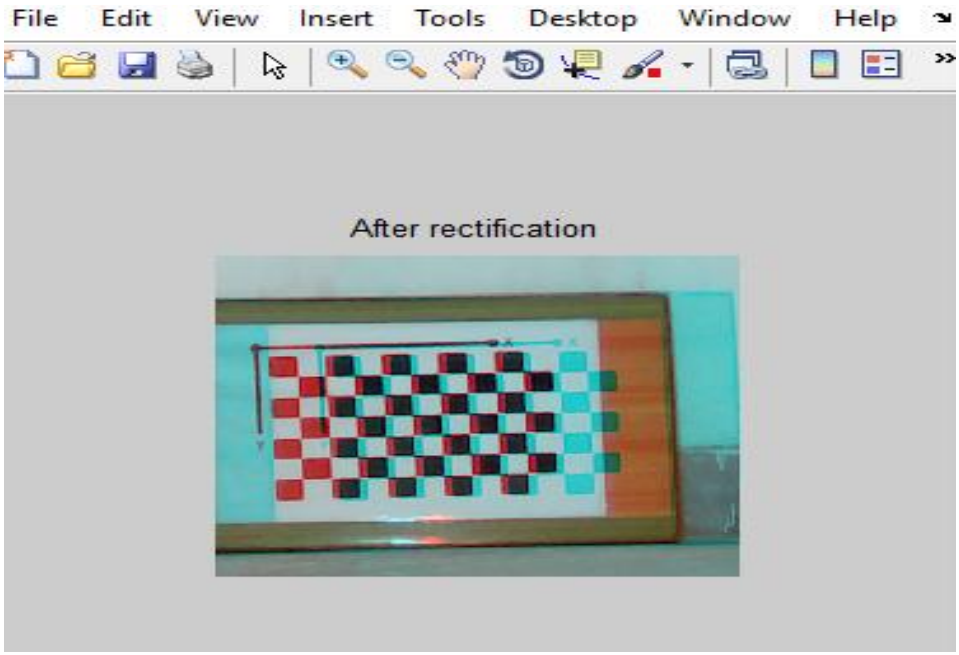


Figure 23(b) Point Cloud

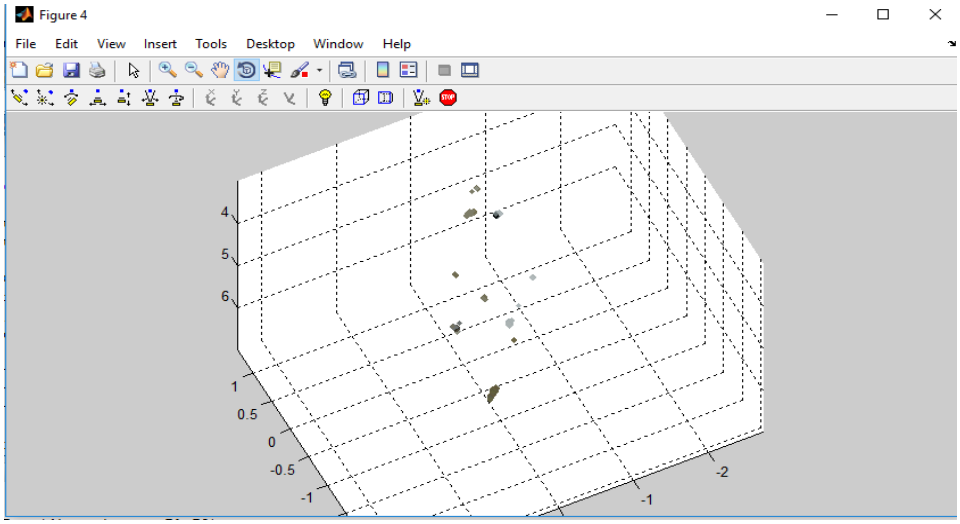
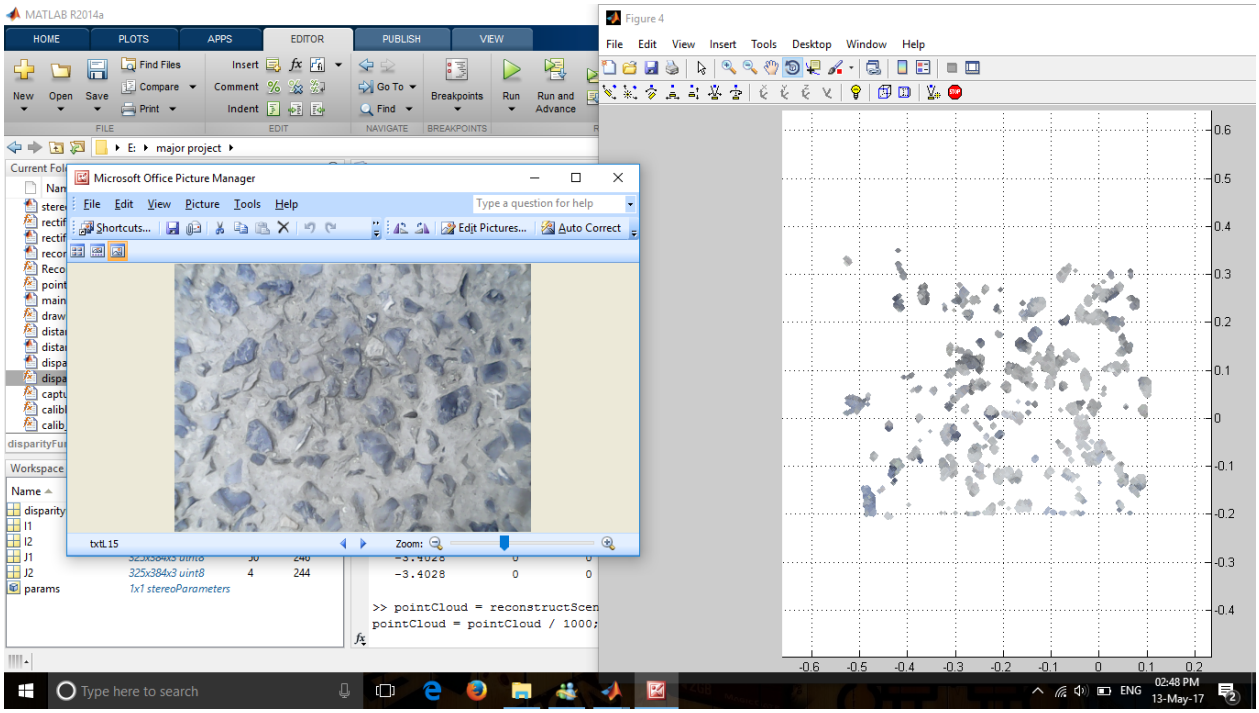


Figure 24


MATLAB R2014a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Microsoft Office Picture Manager
File Edit View Picture Tools Help
Figure 4
File Edit View Insert Tools Desktop Window Help
>> pointCloud = reconstructScen
pointCloud = pointCloud / 1000;
Type here to search
ENG
02:48 PM
13-May-17

CHAPTER-7 DISTANCE ESTIMATION BETWEEN TWO POINTS

7.1 EUCLEDIAN DISTANCE

In mathematics, the **Euclidean distance** or **Euclidean metric** is the "ordinary" straight-line distance between two points in Euclidean space. With this distance, Euclidean space becomes a metric space. The associated norm is called the **Euclidean norm.** Older literature refers to the metric as **Pythagorean metric**. A generalized term for the Euclidean norm is the $\mathbf{L^2}$ **norm** or L^2 distance.

7.1.1 DEFINITION

The **Euclidean distance** between points **p** and **q** is the length of the line segment connecting them (pq).

In Cartesian coordinates, if $\mathbf{p} = (p_1, p_2,..., p_n)$ and $\mathbf{q} = (q_1, q_2,..., q_n)$ are two points in Euclidean *n*-space, then the distance (d) from **p** to **q**, or from **q** to **p** is given by the Pythagorean formula:

$$\begin{aligned} \mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned}$$

The position of a point in a Euclidean *n*-space is a Euclidean vector. So, **p** and **q** are Euclidean vectors, starting from the origin of the space, and their tips indicate two points. The **Euclidean norm**, or **Euclidean length**, or **magnitude** of a vector measures the length of the vector:

$$\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + \cdots + p_n^2} = \sqrt{\mathbf{p} \cdot \mathbf{p}},$$

where the last equation involves the dot product.

A vector can be described as a directed line segment from the origin of the Euclidean space (vector tail), to a point in that space (vector tip). If we consider that its length is actually the distance from its tail to its tip, it becomes clear that the Euclidean norm of a vector is just a special case of Euclidean distance: the Euclidean distance between its tail and its tip.

The distance between points **p** and **q** may have a direction (for example, from **p** to **q**), so it may be represented by another vector, given by

$$\mathbf{q} - \mathbf{p} = (q_1 - p_1, q_2 - p_2, \cdots, q_n - p_n).$$

In a three-dimensional space (*n*=3), this is an arrow from **p** to **q**, which can be also regarded as the position of **q** relative to **p**. It may be also called a displacement vector if **p** and **q** represent two positions of the same point at two successive instants of time.

The Euclidean distance between **p** and **q** is just the Euclidean length of this distance (or displacement) vector:

$$\|\mathbf{q} - \mathbf{p}\| = \sqrt{(\mathbf{q} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})}.$$

which is equivalent to equation 1, and also to:

$$\|\mathbf{q} - \mathbf{p}\| = \sqrt{\|\mathbf{p}\|^2 + \|\mathbf{q}\|^2 - 2\mathbf{p} \cdot \mathbf{q}}.$$

7.1.2 ONE DIMENSION

In the context of Euclidean geometry, a metric is established in one dimension by fixing two points on a line, and choosing one to be the origin. The length of the line segment between these points defines the unit of distance and the direction from the origin to the second point is defined as the *positive direction*. This line segment may be translated along the line to build longer segments whose lengths correspond to multiples of the unit distance. In this manner real numbers can be associated to points on the line (as the distance from the origin to the point) and these are the Cartesian coordinates of the points on what may now be called the real line. As an alternate way to establish the metric, instead of choosing two points on the line, choose one point to be the origin, a unit of length and a direction along the line to call positive. The second point is then uniquely determined as the point on the line that is at a distance of one positive unit from the origin.

The distance between any two points on the real line is the absolute value of the numerical difference of their coordinates. It is common to identify the name of a point with its Cartesian

coordinate. Thus if *x* and *y* are two points on the real line, then the distance between them is given by:

$$\sqrt{(x-y)^2} = |x-y|.$$

In one dimension, there is a single homogeneous, translation-invariant metric (in other words, a distance that is induced by a norm), up to a scale factor of length, which is the Euclidean distance. In higher dimensions there are other possible norms.

7.1.3 TWO DIMENSIONS

In the Euclidean plane, if p = **(p_1, p_2) and** q = **(q_1, q_2) then the distance is given by**

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2}.$$

This is equivalent to the Pythagorean theorem.

Alternatively, it follows from (*2*) that if the polar coordinates of the point **p** are (r_1, θ_1) and those of **q** are (r_2, θ_2), then the distance between the points is

$$\sqrt{r_1^2+r_2^2-2r_1r_2\cos(\theta_1-\theta_2)}.$$

7.1.4 THREE DIMENSIONS

In three-dimensional Euclidean space, the distance is

$$d(p,q) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+(p_3-q_3)^2}.$$

7.1.5 *N* DIMENSIONS

In general, for an *n*-dimensional space, the distance is

$$d(p,q) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_i-q_i)^2+\cdots+(p_n-q_n)^2}.$$

7.2 SQUARED EUCLIDEAN DISTANCE

The standard Euclidean distance can be squared in order to place progressively greater weight on objects that are farther apart. In this case, the equation becomes

$$d^2(p,q) = (p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2.$$

Squared Euclidean Distance is not a metric as it does not satisfy the triangle inequality, however, it is frequently used in optimization problems in which distances only have to be compared.

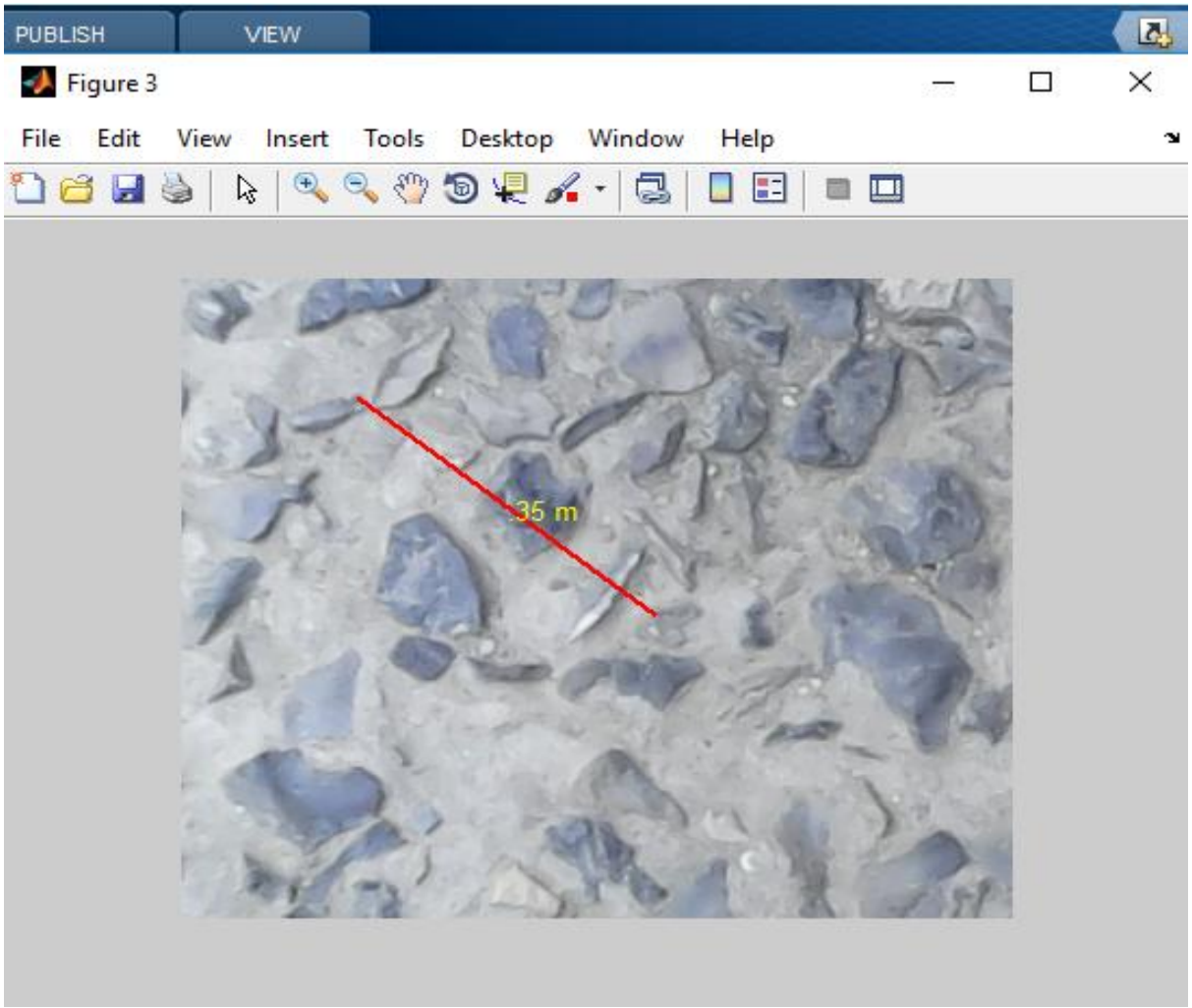


Figure 25

In the figure given above a distance is calculated between two real point coordinates as taken by the user.

CHAPTER-8 PROJECT CODE

Calibration.m :

calibration.m takes stereo image pairs of chessboard images as input and produces calibration parametes.

Rectify.m:

Rectify.m takes stereo parameters output by calibration.m and stereo image pair as input and produces rectify stereo image pair.

disparityFunction.m:

disparityFunc.m takes rectified image pairs and camera calibration parameters as input and outputs a disparity map which is visualized on screen.

Reconstruct.m:

Reconstruct.m takes disparity map and left and right original images as input and outputs 3 matrices having real world x, y and z coordinated named X, Y, Z.

distanceFunction.m:

distanceFunction.m takes X, Y, Z and points between which distance is to calculated p1(x1,y1) and p2(x2,y2)and outputs distance in meters.

drawLine.m:

drawLine.m takes X,Y,Z and rectified left image J1 as input and displays an interactive image on screen on which two point can be marked forming a straight line. After marking it shows the distance.

main.m

```
maxrun=8; % maximum number of times the setup will run without exiting
while(maxrun>0)
choice=input('press q for stereo image capture\n press c for calibration \npress l to launch
reconstructor\n press e for exit\n','s');
if choice=='e'
   break;
end

%To run stereo image capture
if choice=='q'
   captureFunction;
end
if choice=='c'
   fprintf('Running calibration....\n');
   params1=calibFunction;
   save=input('To save calibration parameters press y : \n','s');
   if save=='y'
      save('calib.mat',params1);
   end
end

%For reconstruction
if choice=='l'
   fprintf('Enter Input Stereo Images\n');
   filename1=input('Left Image: \n','s');
   filename2=input('Right Image: \n','s');

   I1= imread(fullfile(matlabroot,sprintf('%s.png',filename1)));
   I2= imread(fullfile(matlabroot,sprintf('%s.png',filename2)));
   [J1,J2]=rectify(I1,I2);
   disparitymap=disparityFunction(J1,J2);
   [X,Y,Z]=Reconstruct(disparitymap,I1,I2);
```

```
    drawLine(J1,X,Y,Z);


end
    maxrun=maxrun-1;
end
```

captureFunction.m

```
function captureFunction

i=0;
filename=input('enter file name','s');
fileext=input('enter extension','s');
camL=webcam(1);     % set the no. to left camera
camR=webcam(3);    % set the no. to right camera
pvL = preview(camL);
pvR = preview(camR);
stop=1;
i=0;
while(stop>0)
    stop=1;
i=i+1;
pause;
IL=snapshot(camL);
IR=snapshot(camR);
while stop<3
 stop=input('press 1,2 to view R,L image,3 to save,9 to reject');
if stop==1
 imshow(IL);
end
if stop==2
imshow(IR);
```

```
end
end

if stop==3
 fil=[filename,'L',num2str(i),fileext];
imwrite(IL,fil);
 fil=[filename,'R',num2str(i),fileext];
imwrite(IR,fil);
end
stop=input('press 0 to stop');
end
clear camL;
clear camR;
end
```

calibration.m

```
numImages = 19;
leftImages = cell(1, numImages);
rightImages = cell(1, numImages);
for i=1:numImages
    leftImages{i} = fullfile(matlabroot, 'd3', sprintf('imgL%d.png', i));
    rightImages{i} = fullfile(matlabroot, 'd3', sprintf('imgR%d.png', i));
end

[imagePoints, boardSize] = detectCheckerboardPoints(leftImages, rightImages);

worldPoints = generateCheckerboardPoints(boardSize, 31.5);

im = imread(leftImages{i});
params = estimateCameraParameters(imagePoints, worldPoints);
```

```
showReprojectionErrors(params);
```

rectify.m

```
function [J1,J2] = rectify(I1, I2)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

%I1 = imread(fullfile(matlabroot,left_image));
%I2 = imread(fullfile(matlabroot,right_image));
s = load('calib.mat');

[J1, J2] = rectifyStereoImages(I1, I2,s.params);
figure; imshow(cat(3, I1(:,:,1), I2(:,:,2:3)), 'InitialMagnification', 50);
title('Before rectification');
figure; imshow(cat(3, J1(:,:,1), J2(:,:,2:3)), 'InitialMagnification', 50);
title('After rectification');

end
```

disparityFunction.m

```
function disparitymap = disparityFunction( J1, J2 )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

disparitymap = disparity(rgb2gray(J1), rgb2gray(J2));
figure;
imshow(disparitymap, [0,64], 'InitialMagnification', 50);
colormap('JET');
colorbar;
```

```
title('Disparity Map');
end
```

reconstruct.m

```
pointCloud = reconstructScene(disparityMap, params);
pointCloud = pointCloud / 1000;

[reducedColorImage, reducedColorMap] = rgb2ind(J1, 128);
hFig = figure; hold on;
set(hFig, 'Position', [1 1 840   630]);
hAxes = gca;

X = pointCloud(:, :, 1);
Y = pointCloud(:, :, 2);
Z = pointCloud(:, :, 3);

for i = 1:size(reducedColorMap, 1)
    x = X(reducedColorImage == i-1);
    y = Y(reducedColorImage == i-1);
    z = Z(reducedColorImage == i-1);

    if isempty(x)
        continue;
    end

    % Eliminate invalid values.
    idx = isfinite(x);
    x = x(idx);
    y = y(idx);
    z = z(idx);

    % Plot points between 3 and 7 meters away from the camera.
```

```
    maxZ = 7;
    minZ = 3;
    x = x(z > minZ & z < maxZ);
    y = y(z > minZ & z < maxZ);
    z = z(z > minZ & z < maxZ);

    plot3(hAxes, x, y, z, '.', 'MarkerEdgeColor', reducedColorMap(i, :));
    hold on;
end

% Set up the view.
grid on;
cameratoolbar show;
axis vis3d
axis equal;
set(hAxes,'YDir','reverse', 'ZDir', 'reverse');
camorbit(-20, 25, 'camera', [0 1 0]);
end
```

distanceFunction.m

```
function [ dist ] = distanceFunction(X,Y,Z,x1,y1,x2,y2)
XNan = isnan(X);
YNan = isnan(Y);
ZNan = isnan(Z);
wfactor = 0.0019;
 X1=X(x1,y1);
    Y1=Y(x1,y1);
    Z1=Z(x1,y1);
     X2=X(x2,y2);
    Y2=Y(x2,y2);
```

```
    Z2=Z(x2,y2);
if XNan(x1,y1) == 0 && YNan(x1,y1) == 0 && ZNan(x1,y1) == 0 && XNan(x2,y2) == 0 &&
YNan(x2,y2) == 0 && ZNan(x2,y2) == 0
   res = sqrt(double(((X2-X1)^2)+((Y2-Y1)^2)+((Z2-Z1)^2)));
else
   ed = sqrt(double(((x2-x1)^2)+((y2-y1)^2)));
   res = ed * wfactor;
end



end
```

drawLine.m

```
function [x1,y1,x2,y2] = drawLine(J1)

figure,imshow(J1)

%# make sure the image doesn't disappear if we plot something else

hold on
%# define points (in matrix coordinates)

[x1,y1]= ginput(1);
[x2,y2]= ginput(1);
a= (x1+x2)/2;
b= (y1+y2)/2;
%# plot the points.
%# Note that depending on the definition of the points,
%# you may have to swap x and y

plot([x1,x2],[y1,y2],'Color','r','LineWidth',2);
text(a,b+2,'Hii','color','yellow','FontSize',10);
```

```
y1 = typecast(uint64(y1), 'int64');
y2 = typecast(uint64(y2), 'int64');
end
```

CONCLUSION

The 3d reconstruction has been done from 2d stereo images. With this application we are able to find the actual distance between two points selected in the image successfully. Even with the poor quality camera having image resolution up to 2 mega pixel we are able to reconstruct up to 60 percent of the 2d image points into 3d world points when using textured image. The results can be improved by using high quality camera, other advanced computer vision and parallel algorithms and high speed processors. Future work of our current application involves creation of complete 3d model of a structure which involves techniques such as image stitching, structure from motion etc. 3d reconstruction from 2d images has wide area of application including robot navigation, surveillance system etc.

REFERENCES

- Computer Vision Algorithm and Applications by Richard Szelisky
- Learning Opencv by Gary Bradski and Adrian Kaehler
- https://en.wikipedia.org/wiki/3D_reconstruction
- http://home.iitk.ac.in/~amit/courses/768/00/sray/index.html
- http://www.nature.com/subjects/3d-reconstruction#news-and-comment
- http://6.869.csail.mit.edu/fa13/lectures/lecture11shapefromX.pdf
- http://research.microsoft.com/apps/search/default.aspx?q=3d+reconstruction
- https://research.google.com/search.html#q=3D%20reconstruction