‘3-D MAPPING AND VISUALIZATION USING “TRIVIM”-AN OPEN SOURCE SOFTWARE’

THESIS SUBMITTED TO

Symbiosis Institute of Geoinformatics

FOR PARTIAL FULFILLMENT OF THE M.Sc. DEGREE

By

Tanvi Kulshreshtha

PRN: 14070241047

(Batch: 2014-16)

Symbiosis Institute of Geoinformatics

Symbiosis International University

5th Floor, Atur Centre, Gokhle Cross Road

Model Colony Pune – 411016

CERTIFICATE

Certified that this thesis titled '3-D Mapping and Visualization using "TRIVIM"- an open source software' is a bonafide work done by Miss. Tanvi Kulshreshtha at Indian Institute of Remote Sensing, Dehradun, Uttrakhand and Symbiosis Institute of Geoinformatics, under our supervision.

Supervisor Internal	Supervisor External
Dr. Navendu Chaudhary	Mr. K. Shiva Reddy [Scientist "SD"]
	Dr. Sameer Saran [Scientist "SF" and Head (GID)]

<u>ACKNWLEDGEMENT</u>

It is with a sense of gratitude, I acknowledge the effort of entire hosts of well-wishers who have in some way or other contributed in their own special way to the success and completion of this summer internship project.

First of all, I express my sage sense of gratitude and indebtedness to the dean of our college **<u>Dr. T.P. Singh</u>** of "<u>Symbiosis Institute of Geoinformatics, Pune</u>", for the immense support and faith.

I would like to thank my project external supervisors at "<u>IIRS</u>" **<u>Mr. K. Shiva Reddy</u>** [Scientist "SD"] and **<u>Dr. Sameer Saran</u>** [Scientist "SF" and Head (GID)] and my Internal Supervisor **<u>Dr. Navendu Chaudhary</u>** who has guided me throughout my project, gave an opportunity to work under their immense expertise. I express my gratitude to my mentors who were kind enough with their valuable suggestion, encouragement and motivation and for their kind advices that guided me through the project.

And Last but not the least I would express my love to my mother Dr. Vinita Kulshreshtha for her immense support and motivation in bringing me here and along the completion of my project.

TANVI KULSHRESHTHA

INDEX

Contents

LIST OF FIGURES

LIST OF TABLES

PREFACE

Digital Elevation Model is a surface geometry model to describe the undulating characteristics of the topography surface, is a discrete digital topographic expression. . The research work is done on the 3D modelling of buildings across the streets. And acquiring their meta-data related to the building including the location parameters. A calculated and planned field work is carried out for the data collection. Data is collected in a systematic workflow with the help of an android applications installed in the android supporting mobile phones and uploaded to the server supported backend. The application used is, ODK that stands for “Open Data Kit”. It has a form builder online webpage with the help of which the required form of selective entities are designed and then uploaded in KML format.

The data is processed ready for the “TRIVIM” the open source software of IIRS, ISRO. It is a research and development application for non-commercial use in training and capacity building for close range photogrammetry. This open source freeware application enables generation of 3D street scenario using a set of sequence of overlapping photographs (minimum being 60%) using DSLR’s/consumer grade camera/mobile phone camera. The application comprises of a series of simple steps which generate a photo textured 3D model that can be imported as kml for 3D visualization in an earth explorer such as Google Earth ,Bing 3D, ISRO Bhuvan 3D etc. The product would provide a visualization and decision support tool by creating georeferenced, photorealistic models attached with attribute database for a variety of applications (e.g. urban planners).

The model displayed provides the related meta-data of the building like-the building is under commercial or residential category or the area of the building and also the different location parameters associated with it. After testing and implementation it will be launched for the commercial and the personal use based on the requirements.

The project work is carried out at GID Dept. of IIRS, Dehradun. The study area is also the city Dehradun, Uttrakhand and Field work is carried out on the major streets of Dehradun.

CHAPTER I
INTRODUCTION

1.1 BACKGROUND

Evolution in photogrammetry and computer vision has changed the view of human of analyzing the 3D virtual reality. With the advancement of technology, creation of 3D models became easy and efficient. Through close range photogrammetric applications it became easy to create 3D model of buildings or any other object. Through a camera and set of images captured by that camera of the object can reconstruct the 3D of that object.

1.1.1 PHOTOGRAMMETRY

1.1.1.1 Definition

Photogrammetry word is composed of three Greek words, “photo”- light, “gram”-drawing, “metry”- measurement. “It is the science of making measurements from photographs”[1]. Photogrammetry is a non-contact measurement method used to extract overall geometric properties (i.e. shape and location) of an object and 3D coordinates (X, Y, Z) of selected points on object’s surface[2]. It is a science which measures the properties of the objects without being touched and then interpretating the information. This method can be applied in any situation where the object is considered, and it can be photographically recorded. Photogrammetry works on the principles of ‘feature recognition’ and ‘triangulation’. First, the digital images are captured of the objects by the high-resolution camera and then those images are processed using photogrammetric software package. The features present are being recognized by observing the image characteristics like shape, texture, etc. and then the images are cross-referenced to build links between the similar features in different images. Now to attain the geometry of the object a spatial location of each feature is solved. 3D reconstruction of an image is done [1].

1.1.1.2 Photogrammetry Process

In photogrammetry, many fields are used optics and projection geometry. Certain elements contribute in photogrammetry like the light of the source, sensors, camera device, and image processing to reconstruct the objects with precision and of good quality shown in Fig 1.1. Certain steps need to be followed shown in Fig 1.2 to generate two-dimensional or three-dimensional models which include image acquisition, i.e., capturing of the high-quality images. It is the process of converting the three-dimensional real world of the objects into two-dimensional images. The three-dimensional world cannot be mapped into two-dimensional entirely; some information is lost, i.e., depth. There occur some geometric changes caused by the shape of the object, perspective imaging, and optical lens defect and position of camera and object[3]. There also occur color changes as the reflected electromagnetic radiation recorded in the image is affected by light sensitive recording medium like film, electronic sensor and the transmission medium like glass, air.

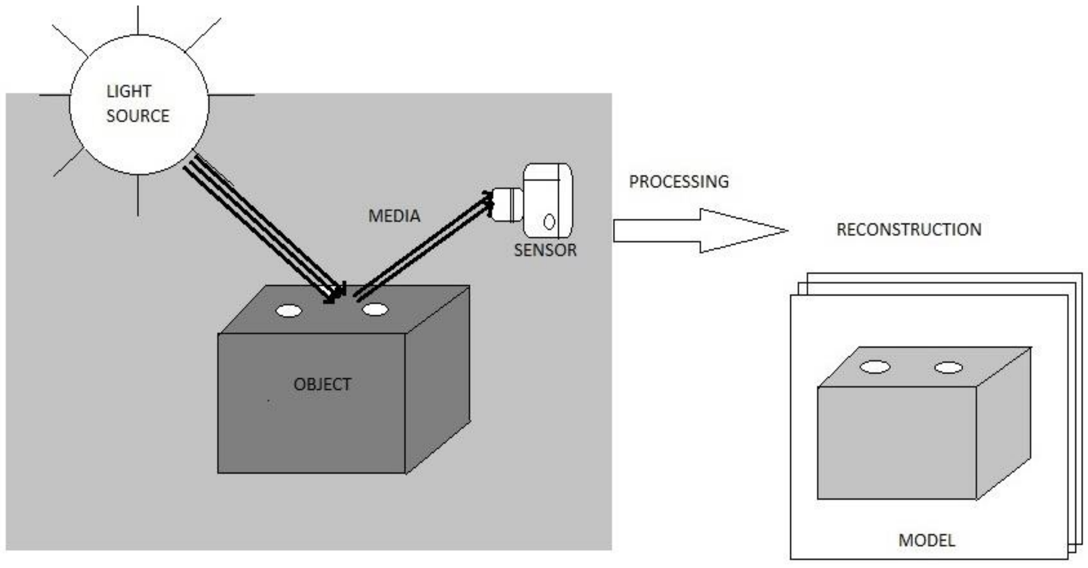


Fig 1.1: From Object to Model

After capturing the images, the measurement and interpretation of images are done in which feature recognition and analyzing of the image are done. Measurement of the image includes measuring the points in the image by its form, brightness or color distribution[3]. All these measurements depend on many factors like distance from the camera, the focal length of the camera lens and other camera parameters.

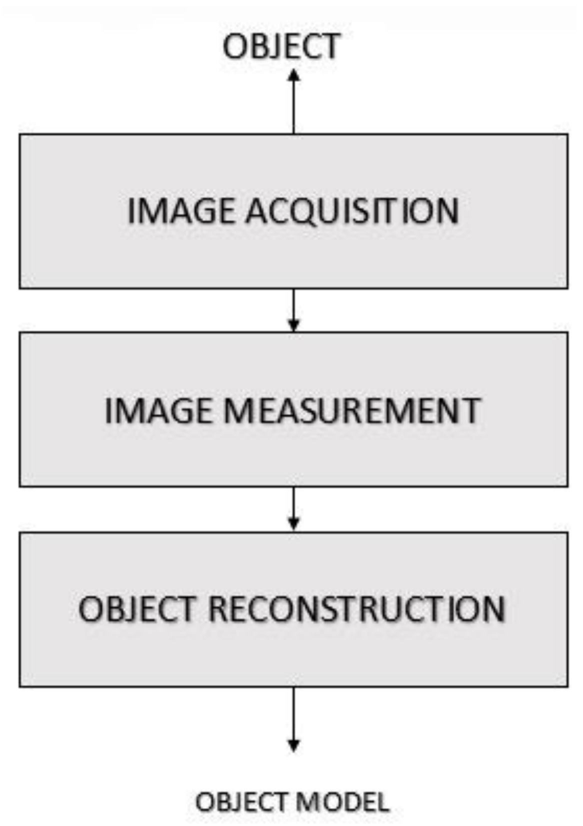


Fig 1.2: Steps to be followed to generate Model from Object

In next step with the help of measured parameters and mathematical transformation, the object is reconstructed. To perform each task instruments are used along with the methods and human knowledge and experience is also used.

1.1.2 CLOSE RANGE PHOTOGRAMMETRY

Photogrammetry is a science of measurement of three-dimensional data from the photographs. It is a technique for obtaining geometric information like position, size, and shape of any object. Photogrammetry discipline can be classified in a number of ways based on their objects and sensor platforms shown in Table 1.1.

Object	**Sensor Platform**	**Specialization**	
Planet	Space Vehicle	Space Photogrammetry	
Earth's Surface	Air Space Vehicle	Aerial Photogrammetry	
Industrial Part	Tripod	Industrial Photogrammetry	**Close Range Photogrammetry (CRP)**
Historical Building	Tripod	Architectural Photogrammetry	
Human Body	Tripod	Biostereometrics	

Table 1.1: Different areas of specialization of photogrammetry, their objects, and sensor platforms[4]

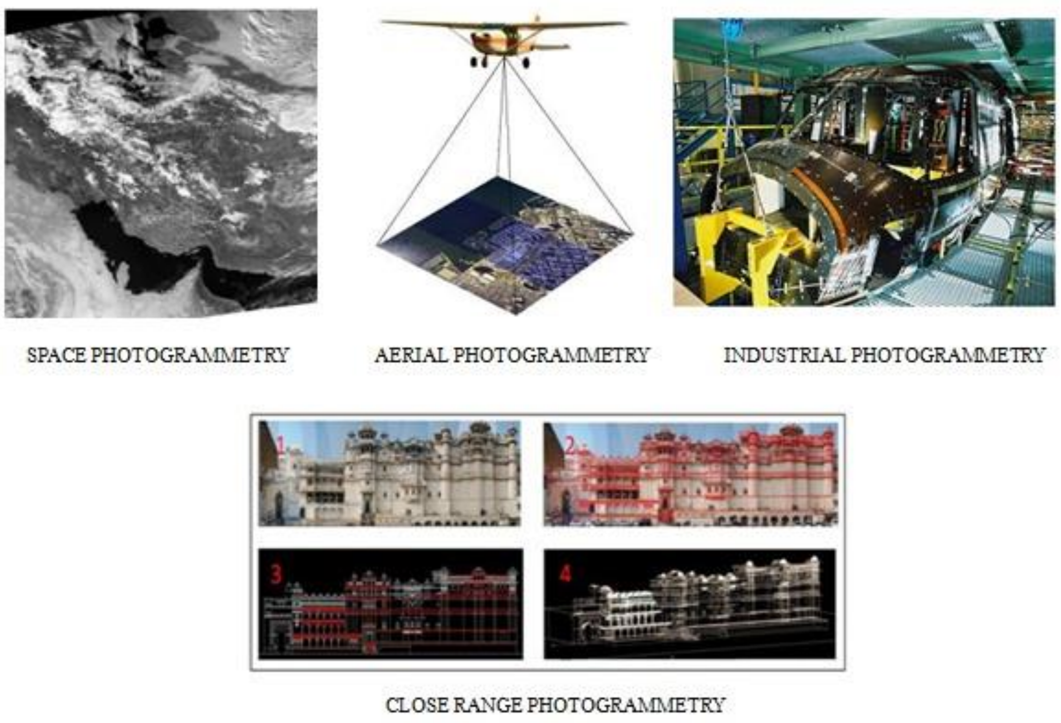


Fig 1.3: Different Areas of Photogrammetry

There are several different areas of photogrammetry present shown n Fig 1.3. One such standard method is Close Range Photogrammetry. In Close Range Photogrammetry, the camera is placed close to the object showing in Fig 1.4 and is typically hand-held on a tripod or can be on a vehicle too.

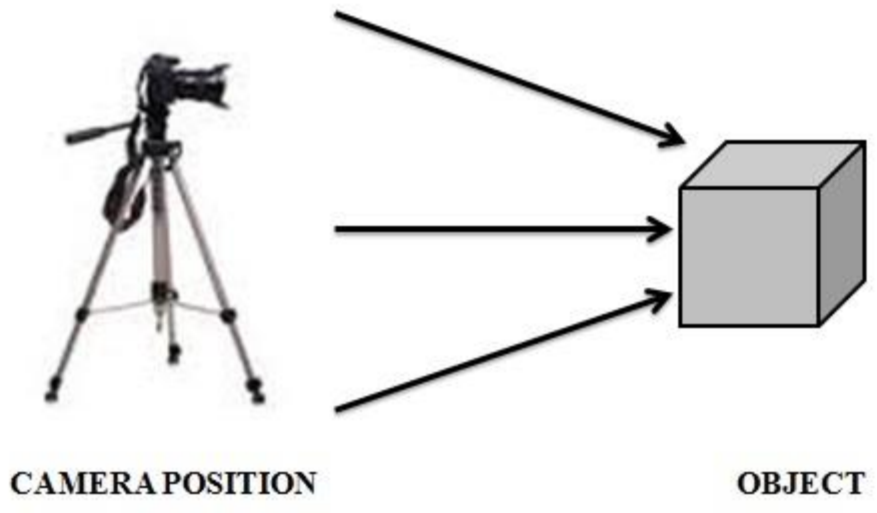


Fig 1.4: Close Range Photogrammetry where camera is placed close to the object

This photogrammetry is non-topographic i.e. the output produced will not be like terrain models, topographic maps but it will have drawings, 3D models, measurements and point clouds[1].Close Range Photogrammetry is used in many areas like automotive, machine, shipbuilding industries, aerospace industries, architecture, heritage conservation, archeology, engineering, etc. In 3D city models, an approach of close range photogrammetry is used [1].

1.1.3 3D MODELING

“3D Modeling is the process of developing a mathematical representation of any three-dimensional surface of an object via specialized software. This product is called a 3D Model. It can be displayed as a two-dimensional image through a process called 3D rendering”[5]. 3D modeling is assumed to be a complete procedure that starts from acquiring data and ends up with 3D model. Often 3D modeling is meant as a process of converting a measured point cloud into a triangulated network or textured surface; overall it is a process of object reconstruction[6]. 3D model of the object doesn’t contain information about the shape of the model; it defines the position of the vertex points which form the corners of the objects and places a skin between these points and contains information of UV mapping. UV mapping is the process of applying a texture to the faces of the object[7].

3D modeling can be assumed as a description of the shape of an object by determination of its main frame of reference or by the generation of Digital Surface Model i.e. description of its surface(s) with a large number of points. Photography can offer a relatively unending record of things of a transient nature. Therefore, collecting accurate three-dimensional measurements and generating a permanent record from which information can be extracted. Therefore, if a scale measure is there in the object's images, it is to be taken as input; it can supply a rough inference of the object's magnitude such as length, width, and height. So, a more broadly applicable set of methods to obtain three-dimensional measurements is the primary apprehension of a discipline called photogrammetry[5].

To determine the unique 3D location i.e. x, y, z coordinate points, photogrammetry uses several images of a single object taken from the different angle of reference. This collection of 3D points results in the information of analogous point data models for advanced processing. The several images collected should be overlapped so that the accurate representation of a three-dimensional surface can be achieved. Overlapping should be between left and right photos about 50 to 80 percent of their surface.

1.1.4 3D CITY MODELS

Photogrammetry plays a central role in facilitating visualization technology to become more practical and profitable. Virtual Reality and 3D Visualization are on the threshold of altering the practices of urban environmental planning and design. Photo-textured models are easy to understand as it can be easy to analyze the elements and view can be oriented in position and scale accordingly[9]. Close range photogrammetry combines the acquisition of building geometry and textures resulting in high accuracy of building reconstruction[10]. The demands for the 3D models are rapidly increasing. There is a shift from traditional 2D GIS to 3D GIS. Fig 1.5 shows the generated 3D model of Tirumala Temple [8].

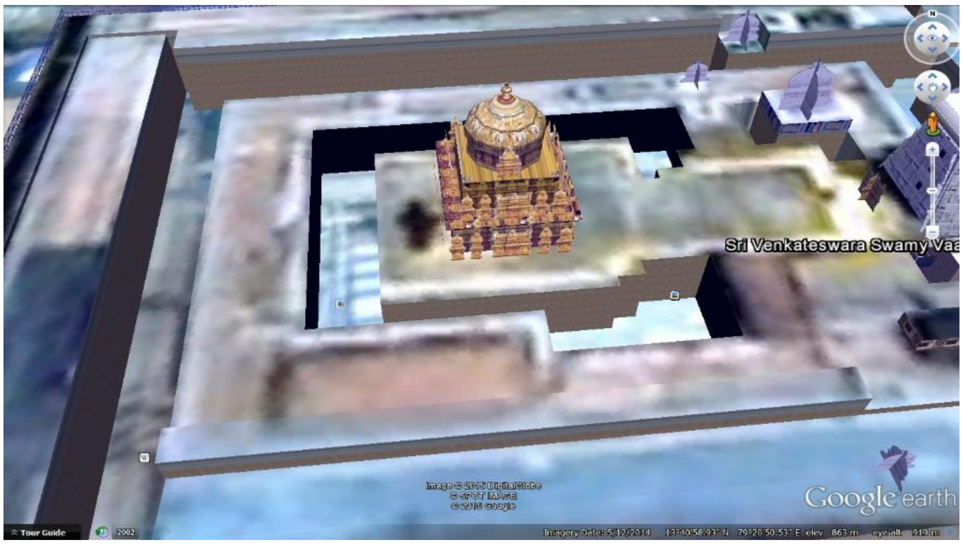


Fig 1.5: Generated 3D model of Tirumala Temple

“3D city models are the digital representations of the Earth’s surface and related objects belonging to urban areas”[11].

“3D city models are digital models of urban areas that represent terrain surfaces, sites, buildings, vegetation, infrastructure and landscape elements as well as related objects belonging to urban areas.” [12]

3D City Models allows “for visually integrating heterogeneous geo-information within a single framework and, therefore, create and manage complex urban information spaces.”[2][13]

Previously some manual methods were used for 3D modeling, but they were time-consuming and required operator’s knowledge and expertise. Now, automatic generation of 3D City Models is done. In the automatic generation systems for 3D City Models, the material data may include laser profiler data, aerial image, and 2D digital maps. With this material information, more detailed and accurate 3D City Models can be generated automatically [14].

There are many applications of 3D city models:

- Spatial planning in early stages
- Calculation floodplains
- Noise and environmental analyzes
- Design process for urban and regional development
- Telecommunications
- Height calculation of building[11]

Commonly, three basic approaches are second-hand for Virtual 3D City Models generation.

1. Conventional techniques such as Vector Map data, DEM, Aerial images are used by researchers,
2. Techniques based on High-resolution satellite images with LASER scanning, and
3. Using Terrestrial images by using Close Range Photogrammetry with DSM & Texture mapping.

We will be considering the third approach for the generation of the 3D City Model. Urban 3D geographic information system refers to the GIS system which can simulate and analyze the spatial objects within the urban area, and it can be widely in urban infrastructure construction and management, urban transport, real estate, urban planning and management.
Various software's exists for 3D modeling. Commercial software available for 3D modeling are not open source and free therefore cannot be commonly used. Also all the task required to achieve an aim is not present in one software. It is the software which consists of these features.

⌘ **Trivim**: Trivim is a research and development application for non-commercial use in training and capacity building for close range photogrammetry developed at the Indian institute of remote sensing (IIRS) ISRO, Dehradun India.

This open source freeware application enables generation of 3D street scenario using a set of sequence of overlapping photographs (minimum being 60%) using DSLR's/consumer grade camera/mobile phone camera. The application comprises of a series of simple steps which generate a photo textured 3D model that can be imported as kml for 3D visualization in an earth explorer such as Google Earth ,Bing 3D, ISRO Bhuvan 3D etc. The product would provide a visualization and decision support tool by creating georeferenced, photorealistic models attached with attribute database for a variety of applications (e.g. urban planners).

The software process through a systematic approach followed:

- Inbuilt capability for calibrating the camera.
- Field planning so that number of photographs required and time required can be previously estimated.
- Generating point cloud from the captured images.

- Georeferencing these point cloud and Georeferencing the images.
- Storing of the spatial information in the form of database along with the object attributes such as textures and elevation of the object.
- 3D model generation of the building with correct orientation and position.
- Visualizing the model
- Database query

There were certain features present in previous software which are tested and certain changes need to be made like Georeferencing the images, auto height extraction of the object and 3D model should be generated with correct position and orientation. Table 1.2 shows the status of previous software's.

Features	**Previous Status**
Calibrating the camera	Yes
Field Planning	Yes
Point Cloud Generation	Yes
Georeferencing the Point Cloud	Yes
Georeferencing the images	No
Texture Extraction	Yes
Auto Height Extraction	No
3D Model Generation	Partial
Database Query	Yes

Table 1.2: Features and Status of previous software.

➢ **OBJECTIVE:** The main objective of the project is 3D Mapping and visualization of Dehradun City using "TRIVIM" an open source software developed by Indian Institute of Remote Sensing, ISRO.

CHAPTER II

LITERATURE REVIEW

2.1 PYTHON PROGRAMMING LANGUAGE

Python is an interpreted, object- oriented, high-level language with dynamic semantics. It's high-level built in data structures, combined with dynamic typing and dynamic binding makes it suitable for Rapid Application Development as well for scripting or glue language to connect existing components together. It's simple, easy to learn and hence less cost of program maintenance. No compilation step, easy to build, edit-test-debug cycle makes it effective (python.org).Python is an easy to learn, easy to read, easy to maintain, contains a broad standard library, supports interactive mode, portable, extendable, provide an interface to all major commercial databases and scalable language.

Many libraries are used in Python such as NumPy, PIL (Python Imaging Library), matplotlib, PyQt, pycollada.

NumPy is the package to perform scientific computing with Python (numpy.org). PIL (Python Imaging Library) adds image processing capabilities to Python interpreter. Library supports many file formats and provides powerful image processing and graphics capabilities (pythonware.com/products/pil). Matplotlib is a python library which is a 2D plotting library that produces publication quality figures (matplotlib.org). PyQt is a set of Python bindings for Qt cross-platform GUI toolkit (pypi.python.org/Pypi/PyQt4). To create, edit and load collada documents we use Collada library[15].

2.2 GUI DESIGN

PyQt is a python binding of cross-platform GUI toolkit Qt. It is one of the Python's options for GUI programming. It is implemented a Python plug-in. It is developed by Riverbank Computing. It's available under to similar terms to Qt versions older than 4.5; an i.e. variety of licensed as GPL and commercial licenses. It implements around 440 classes and over 6000 functions and methods. Its primary components include QtCore, QtGui, QtDesigner, Qt, etc. The GUI required is made using QtDesigner and stored as .ui file then compiled into a .py file using a set of commands. These .py files of GUI can then be used for development by importing and linking.

2.3 CALIBRATION AND USE OF OpenCV

Camera calibration is vital in 3D Model generation. Camera calibration is a necessary step to achieve accuracy and remove distortion in images. Camera Parameters are calculated in camera calibration. There are many camera calibration methods like calibration method based on 3D calibration reference proposed by [16] in which calibration of the reference object is made up of two planar templates which are perpendicular to each other, but this method is expensive as the 3D coordinate of the object surface calibration point need to be known before calibration. Another method is [17] which is based on a 2D template, in which more than two images are captured from different angles without knowing the position and displacement information, we can easily get the internal and external parameters, so this method is easy.

We use OpenCV for camera calibration. OpenCV (Open Computer Vision Library) is an open source library developed by Intel. It consists of C functions and a small amount of C++ classes, uses many algorithms of computer vision and image processing. It can process image and matrix operations. OpenCV uses method and pinhole model as camera model[17]. Using OpenCV on the chessboard images in Fig 2.1, the distortion coefficients and the camera matrix containing the intrinsic and extrinsic parameters of the camera used to click the photos can be determined. Tangential distortion occurs because the image taking lenses is not perfectly parallel to the imaging plane. We have five distortion parameters in OpenCV[18].

$$\text{Distortion}_{\text{coefficient}} = (k_1\ k_{2\ p1\ p2\ k3})$$

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Here we have unknown parameters f_x and f_y, camera focal length and c_x and c_y which are the optical centers expressed in pixel coordinates. The matrix containing these four parameters is the camera matrix. The distortion coefficients are the same despite camera resolution. The process of determining these two matrices is the calibration. Here we use classical black-white chessboard images.

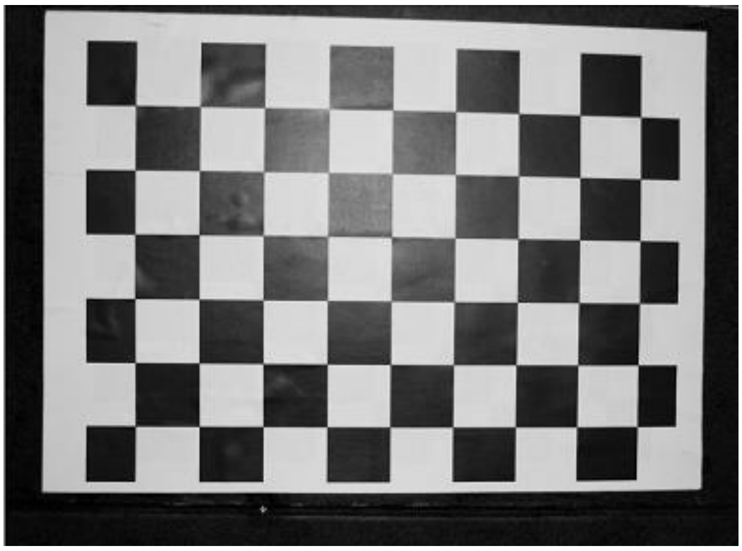


Fig 2.1: Chessboard Image

The module of calibration process based on OpenCV is [19]:

1. Read the gray scale chessboard images, captured from different angles.
2. Processing the images for corner detection.
3. Converting the format of corner point coordinates to CvMat Format of OpenCV.
4. Calculating internal parameters and external parameters.

2.4 POINT CLOUD GENERATION

2.4.1 POINT CLOUD

A point cloud is a collection of data points in some coordinate system. In a three-dimensional coordinate system, these points are referred to X, Y, Z coordinates, used to represent the external structure of an object. Point cloud generation creates a 3D point cloud from a set of overlapping 2D images. Example of point cloud can be seen in Fig 2.2.

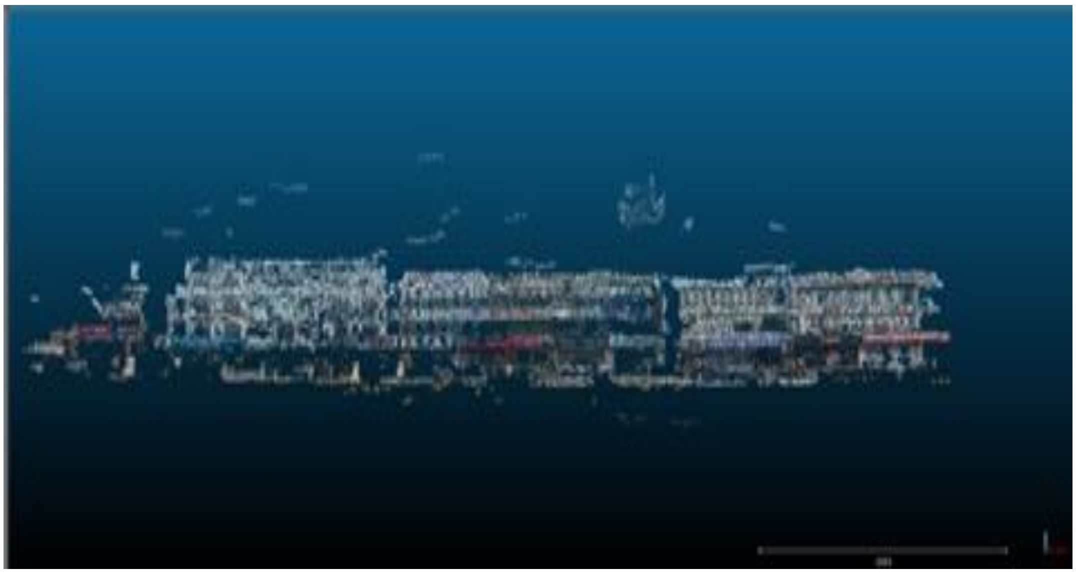


Fig 2.2: Point Cloud

To reconstruct object from images we have large number of freely available photogrammetric software's or algorithms. These software's are:

1. **SIFT:** Scale Invariant Feature Transform is a method developed by Lowe. This method is used for extracting distinctive invariant features from images. SIFT image features provide set of features of an object that are invariant to image scaling and rotation and partially invariant to change in illumination and 3D camera viewpoint[10]. Then matching is performed between different images captured. To extract these image features SIFT algorithm applies 4 stage filtering approach:
   i) **Scale –Space Extrema Detection:** The first stage of computation identifies those locations and scales that are identifiable from different views of the same object. For this we use "scale-space" function which is based on "Gaussian" function[20]. Difference Gaussian technique is then used to detect stable keypoint locations in the scale space.
   ii) **Keypoint Localization:** In this second stage the elimination of points occur from the list of keypoints on the basis of contrast or localization. The points having low contrast or that are poorly localized on edges are eliminated. This step is achieved by Laplacian.
   iii) **Orientation Assignment:** In this step one or more orientation is assigned to each keypoint location based on local image properties. All further operations are performed on this new data that has been transformed relative to the assigned

orientation, scale, and location for each feature, thereby providing invariance to these transformations[21].

iv) **Keypoint descriptor:** Local image gradient are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination[21].

This SIFT method generates a large number of features that densely cover the image over the full range of scales and locations. After generation of features and storing them in a database, image matching and recognition is performed by SIFT. Now each and every feature from the new image is matched with the database stored previously and finding candidate matching features based on the Euclidean distance of their feature vectors. The advantage of using SIFT is that it provide a set of features of an object that are not affected by many complications faced in other methods, like scaling and rotation. After obtaining features from the image, bundler is used.

2. **Bundler:** Bundler is a (SfM) structure from motion system for unordered image collections. Bundler takes a collection of images, images feature and image matches in the form of input and produces a 3D reconstruction of the camera and sparse scene geometry as output[22] shown in Fig 2.3. The scene is reconstructed incrementally, a few images at a time, using Sparse Bundle Adjustment (SBA). Included with the binary distribution is the Bundler executable (bin/bundler) as well as a number of other utility scripts and executables (in the bin/ directory)[22]. In addition, there are a number of example image sets (and example results) under the examples/ directory.

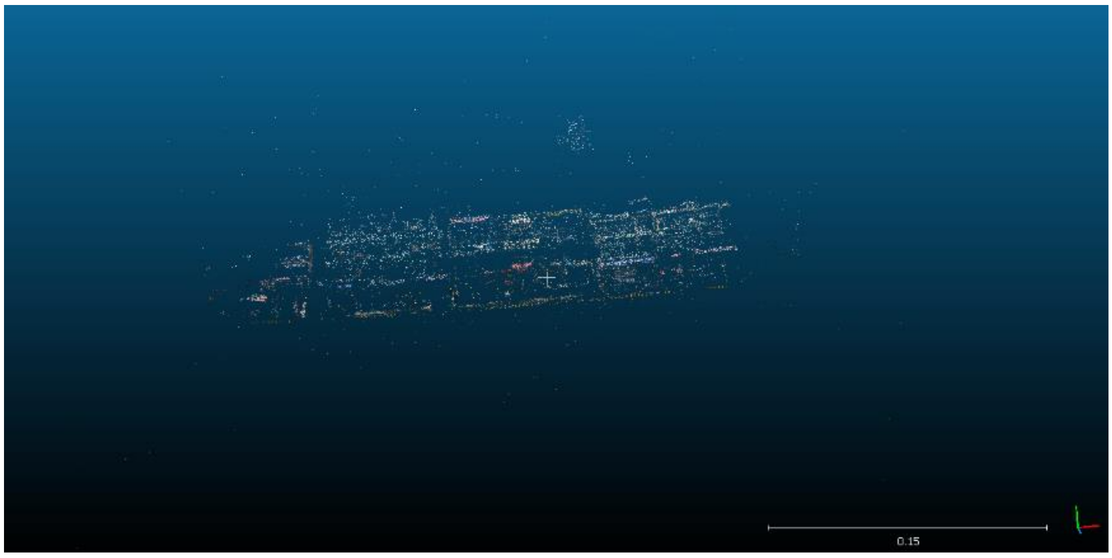


Fig 2.3: Sparse Point Cloud

To reconstruct the scene, we use RunBundler.sh script. Execute this script with set of images in JPEG format and it will run the following steps:

i. Create a list of images using the script '*extract_focal.pl*'. This script extracts focal length information from each image and stores it in an image list.
ii. Generate (SIFT) features for each image. This is done through SIFT method.
iii. The match features between each pair of images. The computed feature matches are stored in a file '*matches.init.txt*'.
iv. Run '*bundler*' with a suitable options file.

Bundler is invoked as:

> Bundler list.txt - -options_file options.txt

The first argument is the list of images to be reconstructed (created with '*extract_focal.pl*' utility). Next, an option file containing settings to be used for the current run created by '*RunBundler.sh*' is passed.

From bundler we get the file bundle.out which contains the estimated scene and camera geometry having the following format:

# Bundle file v0.3

<num_cameras> <num_points>

<camera1>

….

<CameraN>

<point1>

…

<PointM>

Each camera entry contains estimated camera intrinsic and extrinsic, and is in the following format:

<f> <k1> <k2>	[Focal length, 2 radial distortion coefficients]
<R>	[3*3 matrix representing camera rotation]
<t>	[3- vector describing camera translation]

Each point entry has the form:

<Position>	[3-vector describing 3D position of the point]
<Color>	[3*3 vector describing the RGB color of the point]
<View list>	[list of views the point is visible in]

View list contains a quadruplets <camera> <key> <x> <y>, where <camera> is a camera index, <key> the index of the SIFT keypoint where the point was detected in that camera, and <x> and <y> are the detected positions of that key point. The output file bundle.out have sparse point cloud[22].

A utility program for converting bundle files (.out) to the input required by Dr. Yasutaka Furukawa's PMVS multi-view stereo system called Bundle2PMVS is also included. This distribution also includes a program called Radial Undistort for generating undistorted images (based on the undistortion parameters estimated by Bundler).

3. **CMVS (Clustering Views for Multi-view Stereo)** is a software which takes the output of SfM software which is Bundler used here as input and then decomposes the input images into a set of image clusters of manageable size. MVS software can process each cluster independently and in parallel, where the union of reconstructions from all the clusters

should not miss any details that can be otherwise obtained from whole image set. Multi-view stereo software PMVS2 is included in CMVS package[23]. By PMVS2 .ply file is generated which contains dense point cloud in local coordinate system shown in Fig 2.4.

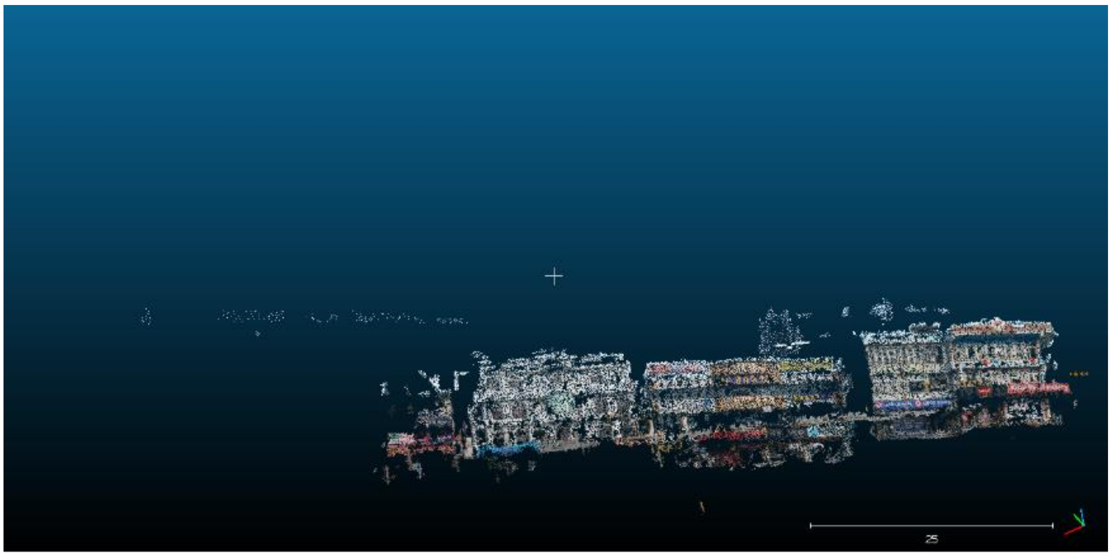


Fig 2.4: Dense Point Cloud

4. **PMVS-2(Patch-based Multi-view Stereo Software)** is software which takes a set of images and camera parameters as input and then reconstructs the 3D structure of an object or scene of the images. Through this, the only rigid structure is constructed. The software automatically ignores non-rigid objects. The software gives set of oriented points as output, where both the 3D coordinate and surface normal are estimated at each oriented point[24]. The output by PMVS is in 3 different files. Let N denote the name of the options file used in the reconstruction. Then we have N.ply, N.patch, N.pset files as output.
    i) **N.ply:** It contains 3D colored points for visualization purposes.
    ii) **N.patch:** It contains full reconstruction information.
    iii) **N.pset:** It contains a list of the 3D locations and estimated surface normal's for all the reconstructed points.

2.5 CLOUD COMPARE

Cloud Compare is an application for managing and comparing 3D point clouds (and, to some extent surface meshes). It takes input .ply file produced by PMVS, and further processing is done on those cloud points [15].The program is not designed for commercial use. It was largely developed by Daniel Giradeau-Montaut, with participation of: Salma Baugacha (Intern at EDF R&D in 2004); Aurelien Bey (PhD student at EDF R&D since the start of December 2008) and Raphael Marc (research engineer at EDF R&D). It can calculate local distance between two dense point cloud, identify individual parts in two similar data sets etc[25] .We use cloud compare to visualize the points generated by Bundler, PMVS/CMVS.

2.6 3D MODEL GENERATION THROUGH PYCOLLADA

Pycollada is a python library for creating, editing and loading COLLADA[15]. COLLADA (collaborative design activity) is an interchange file format for interactive 3D applications. It is managed by, the Khronos Group. The library makes to load a COLLADA file and interact with it as a python object. Pycollada uses lxml for XML loading, construction, and saving. NumPy is used for numerical arrays [15].COLLADA defines an open standard XML schema for exchanging digital assets among various graphics software applications that might otherwise store their assets incompatible file formats. COLLADA documents that describe digital assets are XML files, identified with .dae (Digital Asset Exchange) filename extension[26]. These dae files can then be visualized on Google Earth/Bhuvan. Along with .dae files, we also have KML files for developing 3D models of the building. As per as google documentation, there are several entities which are defined for every model in the corresponding KML files such as[27]:

1. ***<altitudeMode>*** - Specifies how the <altitude> specified in <Location> is interpreted. Possible values are as follows:
    - ***clampToGround*** – It is by default. It indicates to ignore the <altitude> specification and keep the Model on the ground.
    - ***relativeToGround*** - Interprets the <altitude> as a value in meters above the ground.
    - ***absolute*** - Interprets the <altitude> as a value in meters above sea level.
2. ***<gx:altitudeMode>*** - A KML extension in the Google extension namespace, allowing altitudes relative to the sea floor. Values are:
    - ***relativeToSeaFloor*** - Interprets the <altitude> as a value in meters above the sea floor. If the point is above land rather than sea, the <altitude> will be interpreted as being above the ground.
    - ***clampToSeaFloor*** - The <altitude> specification is ignored, and the Model will be positioned on the sea floor. If the point is on land rather than at sea, the Model will be positioned on the ground.
3. ***<Location>*** - Specifies the exact coordinates of the Model's origin in latitude, longitude, and altitude. Latitude and longitude measurements are standard lat-lon

projection with WGS84 datum. Altitude is distance above the earth's surface, in meters, and is interpreted according to <altitudeMode> or <gx: altitudeMode>. For example:

<Location>
<Longitude>78.05830296424719 </longitude>
<Latitude>30.3380113844064</latitude>
<Altitude>5.0</altitude>
</Location>

4. ***<Orientation>*** - Describes rotation of a 3D model's coordinate system to position the object in Google Earth/Bhuvan. It contains the following parameters:

    - ***<heading>*** - Rotation about the z-axis (normal to the Earth's surface). A value of 0 (the default) equals North. A positive rotation is clockwise around the z-axis and specified in degrees from 0 to 360.
    - **<tilt>** - Rotation about the x-axis. A positive rotation is clockwise around the x-axis and specified in degrees from 0 to 180.
    - ***<roll>*** - Rotation about the y-axis. A positive rotation is clockwise around the y-axis and specified in degrees from 0 to 180.

5. ***<Scale>*** - Scales a model along the x, y, and z axes in the model's coordinate space.

    <Scale>

    <x>3.016484140122818</x>

    <y>0.9269862356476752</y>

    <z>1.971735187885247</z>

    </Scale>

6. ***<href>*** - A URL (either an HTTP address or a local file specification).When the parent of <Link> is a Model, <href> is a COLLADA file. In this tag we can use relative URL.[27].

2.7 KML

KML (Keyhole Markup Language) is a file format used to display geographic data in an Earth Browser such as Google Earth, Bhuvan, etc. The tag-based structure is used by the KML with nested elements and attributes and is based on XML standards. All tags are case sensitive (wiki). KML was developed to use with Google Earth, which was originally named Keyhole Earth Viewer. It was created by Keyhole, Inc, which was acquired by Google in 2004. KML became an international standard of the Open Geospatial Consortium in 2008. Google Earth was the first program able to view and graphically edit KML files. The KML file consist of a set of features (placemarks, images, polygons, 3D models, textual descriptions, etc.) for display in Google Earth, Maps and Mobile, or some other geospatial software implementing the KML encoding[28].

2.8 KMZ ARCHIVE

A KMZ file consists of the main KML file with zero or more supporting files that are packaged using a Zip utility into one unit called an archive. The KMZ file can then be stored as a single entity. When the unzipping of the KMZ file is done, the main .kml file and its supporting files are separated into their original formats and directory structure, with their original filenames and extensions. In addition to being an archive format, the Zip format is also compressed, so an archive can include only a single large KML file. Depending on the content of the KML file, this process typically results in 10:1 compression. Google Earth and Bhuvan can read the KML and KMZ files directly, and they can save files as KMZ files. By default, the main KML file is named doc.kml[28].

2.9 PLACEMARK

When KML is generated through pycollada then the placemark on the 3D Model can be created. Placemark marks the position on the earth’s surface. Placemark can be a description, a custom icon or a style map that defines a rollover icon. The basic placemark includes only a <Point> element, in this element we have the location of the placemark shown in Fig 2.5. Information can be added in the placemark like name of the location.

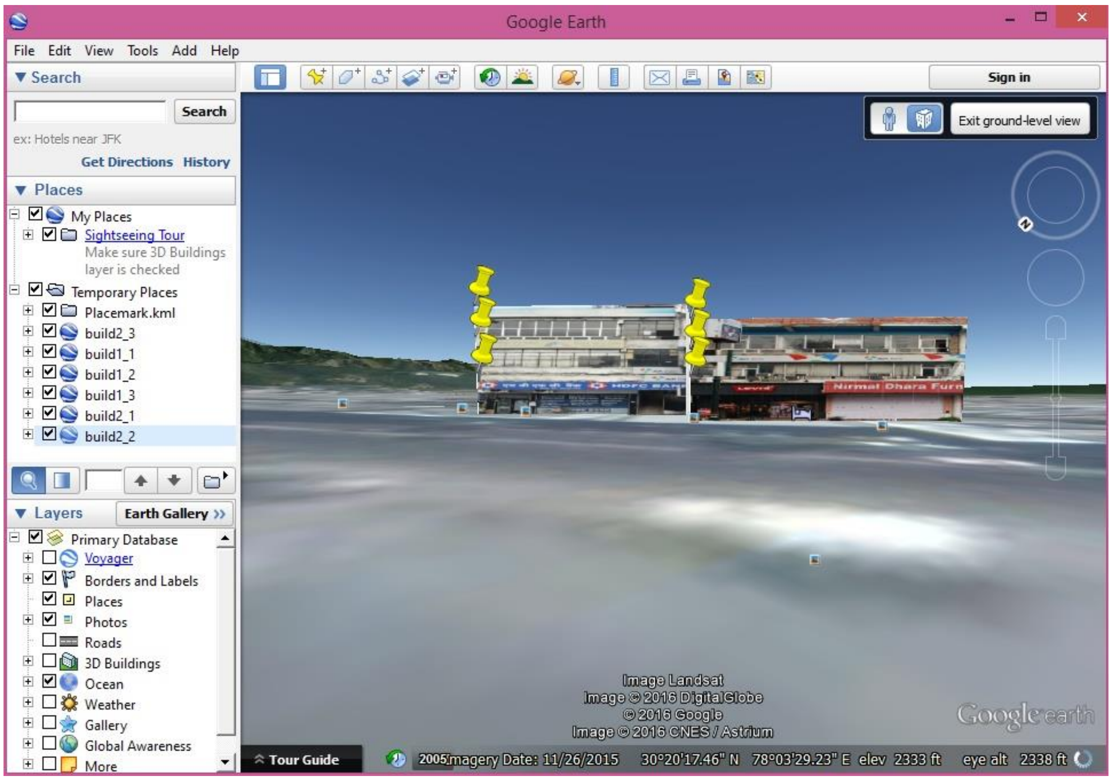


Fig 2.5: Placemark on the Model

KML code for basic placemark is:

```
<? xml version=”1.0” encoding=”UTF-8”? >

<kml xmlns=”http://www.opengis.net/kml/2.2”>

 <Placemark>

 <name>Simple</name>

 <description>Hello</description>

<Point>

<coordinates>78.0583029, 30.338011</coordinates>

</Point>

</Placemark>

</kml>
```

Structure of the code is 1st line contains XML header, 2nd line declares KML namespace. Further the Placemark object contains following elements:

- Name that is used to label Placemark.
- Description contains information
- Point specifies the position where placemark should be placed, it contains longitude, latitude and altitude (optional).

This placemark is at particular point, we can create a marker on whole building by creating polygon. The KML code for point at particular building is:

```
<? xml version="1.0" encoding="UTF-8" standalone="no”?>

<kml xmlns="http://www.opengis.net/kml/2.2"
xmlns:gx="http://www.google.com/kml/ext/2.2">

<Folder>

<Placemark>

    <ExtendedData>

<Data name="Name">

      <value>build1_1</value>

     </Data>

<Data name="Area">

      <value>21</value>

     </Data>

<Data name="Height:">

      <value>3.869741831</value>

     </Data>

</ExtendedData>

    <Point>
```

```
 <coordinates>78.05830296424719, 30.3380113844064, 3.869741831</coordinates>
            <altitudeMode>relativeToGround</altitudeMode>
            <extrude>1</extrude>
      </Point>
</Placemark>
</Folder>
</kml>
```

In the code <extrude> is used to extends the line down to the ground, KML is generated and is used in earth browser such as Google Earth, Bhuvan etc. [28].

2.10 ODK COLLECT:

ODK (Open Data Kit) is an open-source suite of tools that helps organizations author, field, and manage mobile solutions. The goal is to make open-source and standards-based tools which are easy to use, easy to modify and easy to scale. ODK's core developers are the researchers at the Washington's department of computer science and Engineering department and active members of change, a multi-disciplinary group at UW exploring how technology can improve the lives of underserved populations around the world. It is sponsored by google and you can find its source code on GitHub.

It is an app used for data collection, with the help of android supporting mobile phones and later on connected to the server for the transferring the data to common /Personal server and used to obtain information out of the data. Open Data Kit (ODK) is a free and open-source set of tools which help organizations author, field, and manages mobile data collection solutions. ODK provides an out-of-the-box solution for users to:

1. Build a data collection form or survey(XLSForm is recommended for larger forms);
2. Collect the data on a mobile device and send it to a server; and
3. Aggregate the collect data on a server and extract it in useful formats.

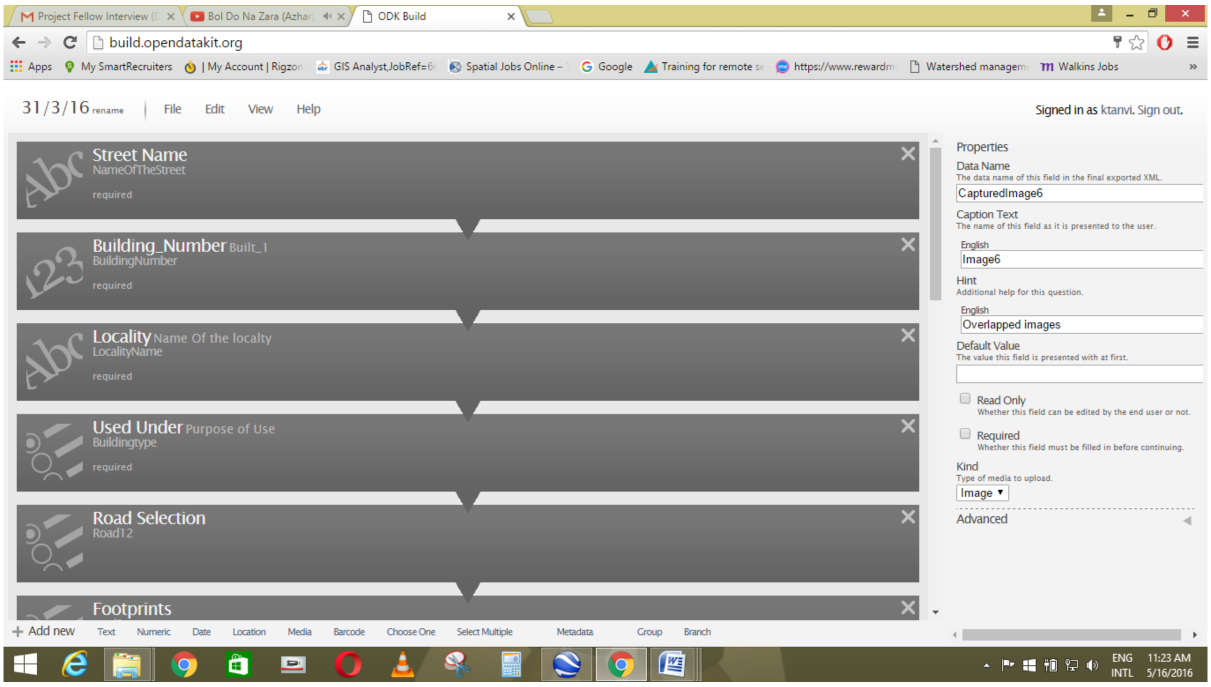
The required form format is built on ODK collect form builder site on web and this form can be uploaded to the server page in form of a XML format. Later on when obtaining the collected data it is gathered in CMV file format.

ODK collect is used under this project for the building of form to collect the data in a proper format and to keep the sequential record of every meta-data that to be displayed at the end with the generated 3D model of the buildings and provide information about them at one instance. A common server has been built and the information is taken care of under the secured server.

The data is collected with the sync of the ODK collect app in android mobiles and worked under different required parameters. And then the filled forms are uploaded to the server, which can be later on/ or side by side downloaded from the server site in the form of CSV's.

From these files data is fetched automatically from CSV files to separate folder including the metadata of each building in specified folder and then finally they are processed under the TRIVIM software. This work of automation of generating files and fetching data from CSV's directly is coded with the help of python programming. Further links are attached with the references.

Screen shots of different working pages are attached:



Street Name
NameOfTheStreet
required

Building_Number Built_1
BuildingNumber
required

Locality Name Of the localty
LocalityName
required

Used Under Purpose of Use
Buildingtype
required

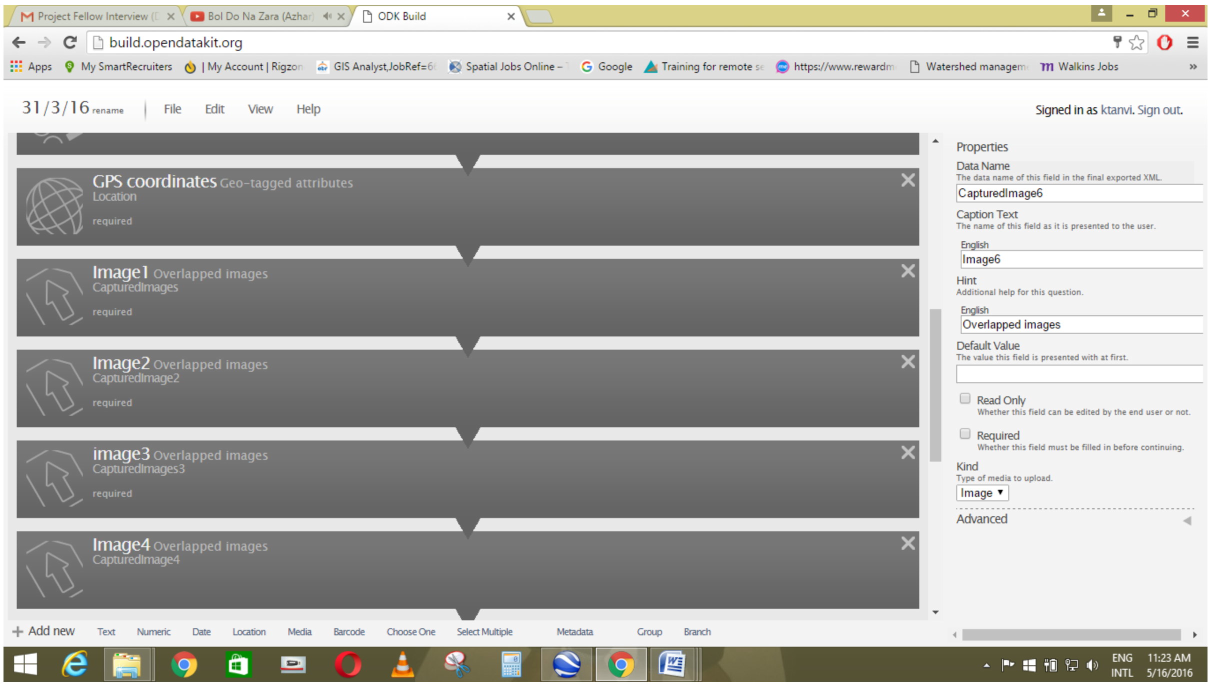
Road Selection
Road12

Footprints

Properties
Data Name
The data name of this field in the final exported XML.
CapturedImage6
Caption Text
The name of this field as it is presented to the user.
English
Image6
Hint
Additional help for this question.
English
Overlapped images
Default Value
The value this field is presented with at first.

Read Only
Whether this field can be edited by the end user or not.
Required
Whether this field must be filled in before continuing.
Kind
Type of media to upload.
Image
Advanced

Fig 2.6: ODK Build showing Form created for data collection Part 1



GPS coordinates Geo-tagged attributes
Location
required

Image1 Overlapped images
CapturedImages
required

Image2 Overlapped images
CapturedImage2
required

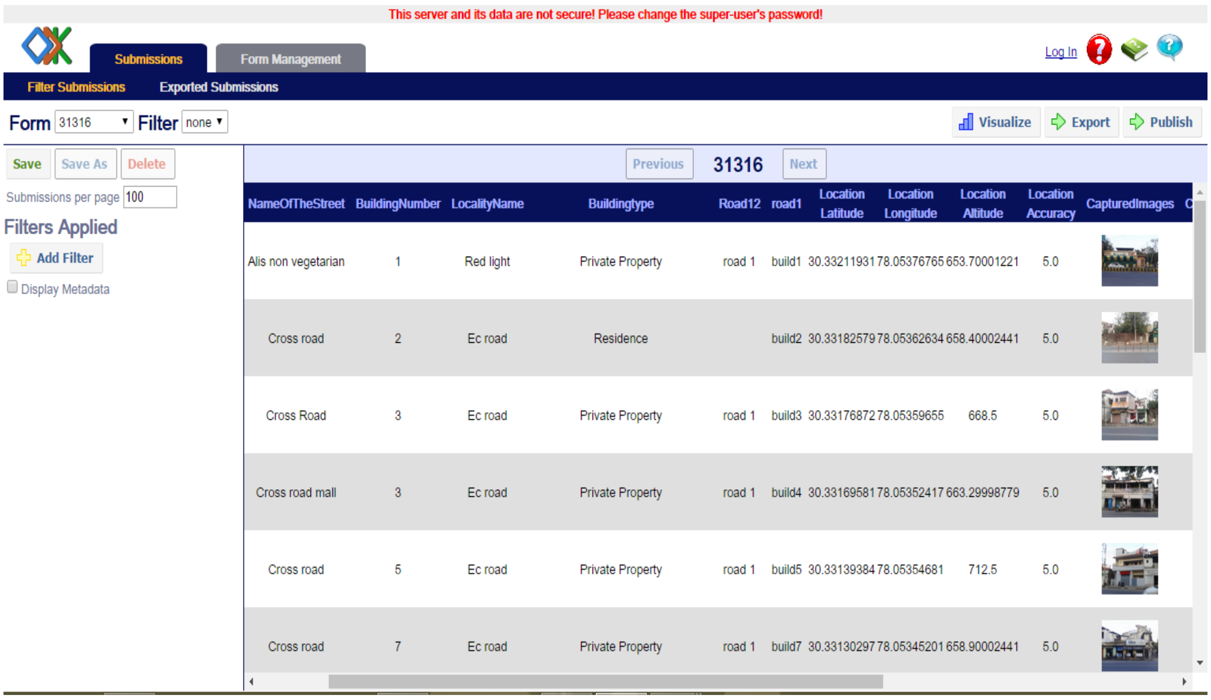
image3 Overlapped images
CapturedImages3
required

Image4 Overlapped images
CapturedImage4

Properties
Data Name
The data name of this field in the final exported XML.
CapturedImage6
Caption Text
The name of this field as it is presented to the user.
English
Image6
Hint
Additional help for this question.
English
Overlapped images
Default Value
The value this field is presented with at first.

Read Only
Whether this field can be edited by the end user or not.
Required
Whether this field must be filled in before continuing.
Kind
Type of media to upload.
Image
Advanced

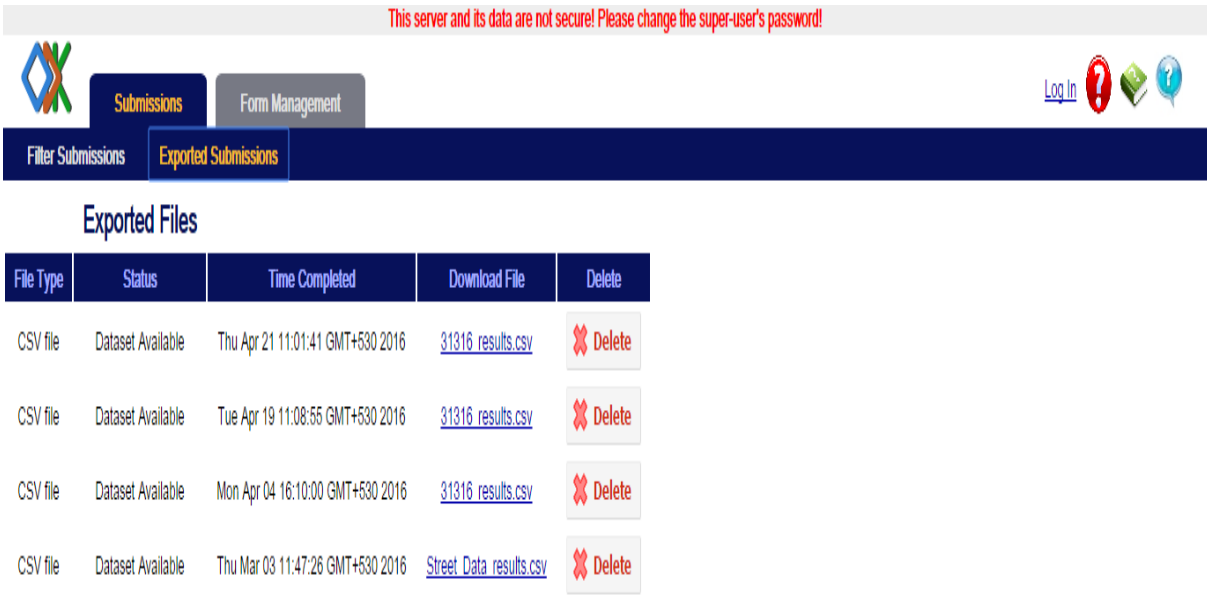
Fig 2.7: ODK Build showing form created for data collection Part 2



This server and its data are not secure! Please change the super-user's password!

NameOfTheStreet	BuildingNumber	LocalityName	Buildingtype	Road12	road1	Location Latitude	Location Longitude	Location Altitude	Location Accuracy	CapturedImages
Alis non vegetarian	1	Red light	Private Property	road 1	build1	30.3321931	78.05376765	653.7001221	5.0	
Cross road	2	Ec road	Residence		build2	30.3318257	978.05362634	658.40002441	5.0	
Cross Road	3	Ec road	Private Property	road 1	build3	30.33176872	78.05359655	668.5	5.0	
Cross road mall	3	Ec road	Private Property	road 1	build4	30.33169581	78.05352417	663.29998779	5.0	
Cross road	5	Ec road	Private Property	road 1	build5	30.33139384	78.05354681	712.5	5.0	
Cross road	7	Ec road	Private Property	road 1	build7	30.33130297	78.05345201	658.90002441	5.0	

Fig 2.8: ODK build Server page Showing Uploaded Entries



This server and its data are not secure! Please change the super-user's password!

Exported Files

File Type	Status	Time Completed	Download File	Delete
CSV file	Dataset Available	Thu Apr 21 11:01:41 GMT+530 2016	31316_results.csv	Delete
CSV file	Dataset Available	Tue Apr 19 11:08:55 GMT+530 2016	31316_results.csv	Delete
CSV file	Dataset Available	Mon Apr 04 16:10:00 GMT+530 2016	31316_results.csv	Delete
CSV file	Dataset Available	Thu Mar 03 11:47:26 GMT+530 2016	Street_Data_results.csv	Delete

Fig 2.9: ODK Build Server Page, Entries are converted in CSV formate

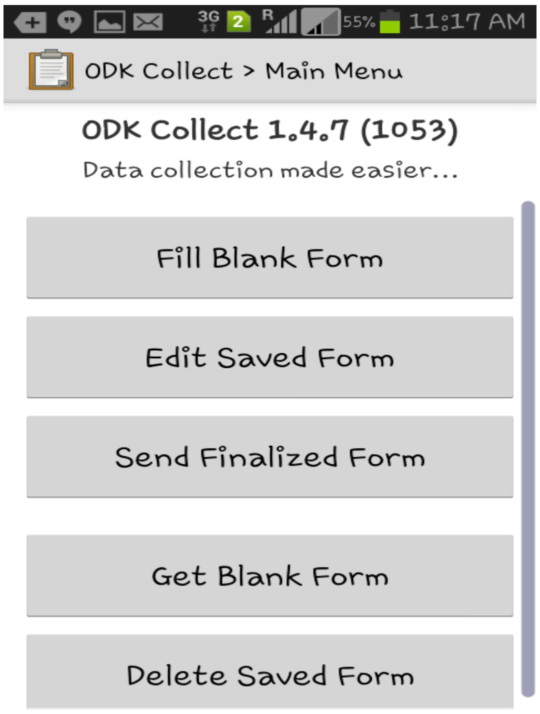
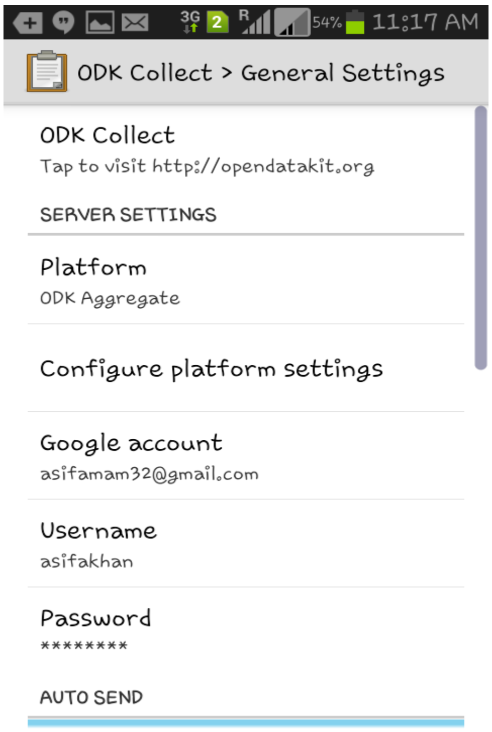


Fig 2.10: ODK Collect Mobile Application Page



ODK Collect > General Settings

ODK Collect
Tap to visit http://opendatakit.org

SERVER SETTINGS

Platform
ODK Aggregate

Configure platform settings

Google account
asifamam32@gmail.com

Username
asifakhan

Password
****

AUTO SEND

Fig 2.11: ODK Collect Mobile APPlication Settings

CHAPTER III

STUDY AREA

The project work is carried out at GID Dept. of IIRS, Dehradun. The study area is also the city Dehradun, Uttrakhand and Field work is carried out on the major streets.

Dehradun is the capital city of the state of Uttarakhand in the northern part of India. Located in the Garhwal region, it lies 236 kilometers (147 miles) north of India's capital New Delhi. Dehradun is in the Doon Valley on the Foothills of the Himalayas nestled between two of the India's mightiest rivers- the Ganges on the east and the Yamuna on the west. The city is famous for its picturesque landscape and the slightly milder climate and provides a gateway to the surrounding region. It is renowned for its natural resources, publishing services and particularly for its educational institutions. It is one of the highest rain receiving areas of North India.

The District contains Rajaji National Park which is home to several elephants, Benog Wildlife Sanctuary at Mussoorie and Asan Conservation Reserve. As the city is located between 30-31^0 N the climate is exclusively classified under Humid Subtropical Climate and is often Continental type. The city is also referred to as Rainy City.

A map of the field work area and considered study area is attached with the thesis.

Also a small project work of Swach Bharat program is involved in our internship for testing the web based application which involved a small survey and testing the application for IIRS.

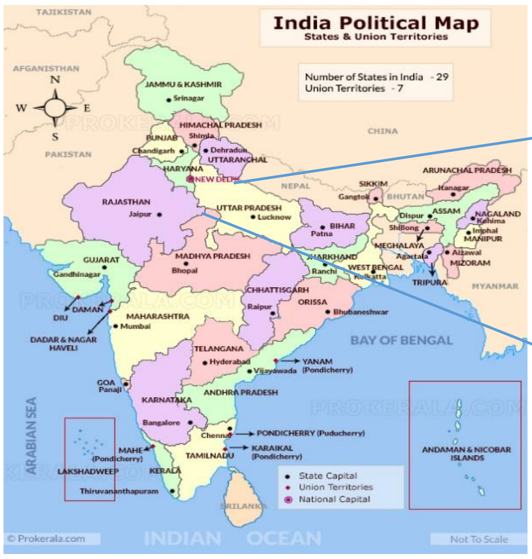


Fig 3.1: Political Map of India

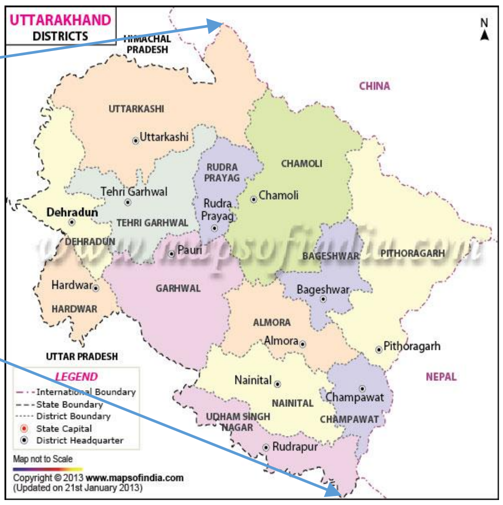


Fig 3.2: Political Map Of Uttarakhand

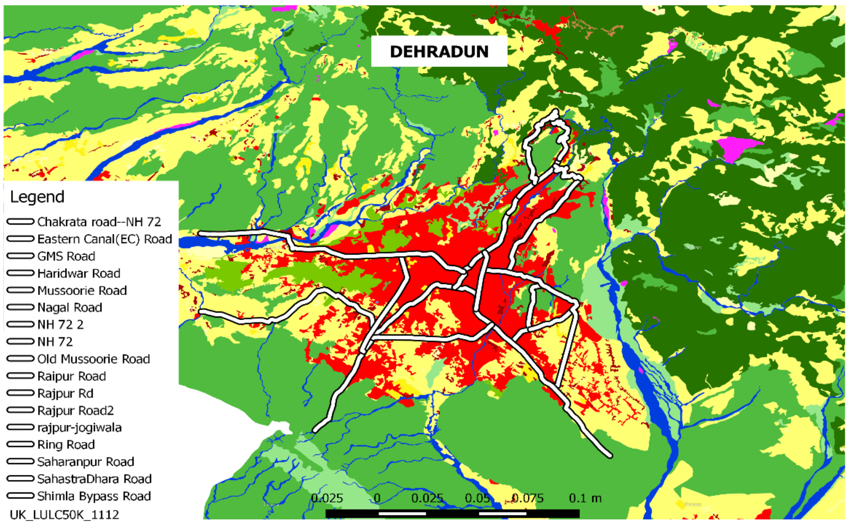


Fig 3.3: LULC Map OF Dehradun, Uttarakhand

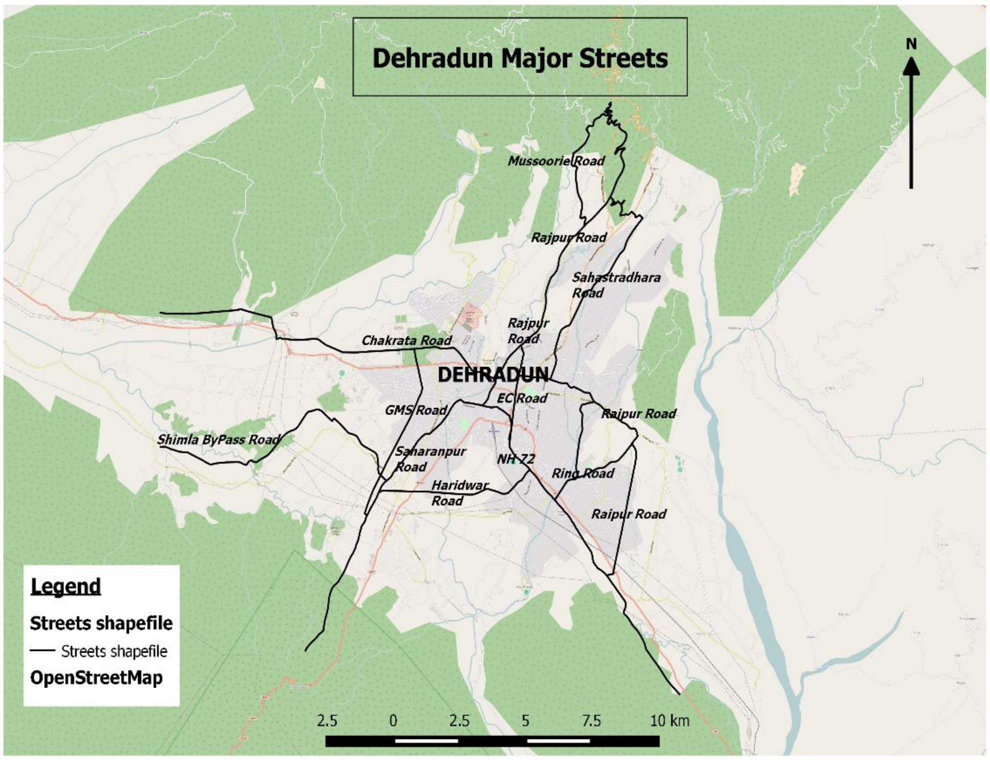


Fig 3.4: Street Map of Study Area of Dehradun, Uttarakhand

CHAPTER IV

TOOLS USED AND METHODOLOGY

4.1 TOOLS USED

4.1.1 HARDWARE TOOLS

Table 4.1 shows the description of the hardware used in the research work:

S. No.	Hardware	Model No.	Used for
1	Cameras	Nikon D-60 SLR, Galaxy Core	Capturing images
2	Laptop	64 bit	Processing work

Table 4.2 shows minimum hardware requirements for execution:

Minimum Requirement	Recommended
Intel Pentium Processor 1GB of RAM 1GB of free Hard Disk Space	Intel core i5 or above 4GB of RAM or more 2GB or more free Space on Hard Disk

4.1.2 SOFTWARE TOOLS

Several software's are used in the execution of the application. Table 4.3 shows the description of the software's used in application.

Table 4.3: Software's used

S. No.	**Software/Package**	**Used for**
1	Microsoft Office Picture Manager 2007	Re-sampling of captured images
2	SIFT v4.0 (compiled binaries files)	Extract key features from images.
3	Bundler v0.3 (open source package)	Features matching, point clouds generation and estimate the exposure station position, orientation parameters. (It contains binary and source code of Sparse Bundler Adjustment (SBA) v1.5 packages).
4	PMVS2 (open source package)	Generate dense point clouds. (It is Patch-based Multi-view Stereo Software v2)
5	Open CV	Calibration of camera
6	Google Earth	Visualization of 3D Model
7	ODK Build	Form creation and server upload
8	ODK collect	Data Collection

4.2 METHODOLOGY

The methodology follows certain steps; firstly image acquisition of the object using a pre calibrated camera occurs. Further captured images are processed for automatic feature extraction using software SIFT and then matching of the sequence of images occur. Further sparse bundle adjustment is applied to generate sparse point cloud. Dense point cloud in local coordinate system is generated by PMVS using camera parameter files. Further it is transformed into global coordinate system by using space photo intersection and similarity transformation with proper scale. Then the generated point cloud and captured images are georeferenced. In next step segments from the images of the floors of the building are taken and while segmentation auto height extraction of the segment takes place. Further kml files for each building are loaded in next step and photo-textured 3D models are generated that can be visualized on geo-portal like Google Earth. Detailed methodology is shown in Fig 5.2.1 and each step is explained in following section.

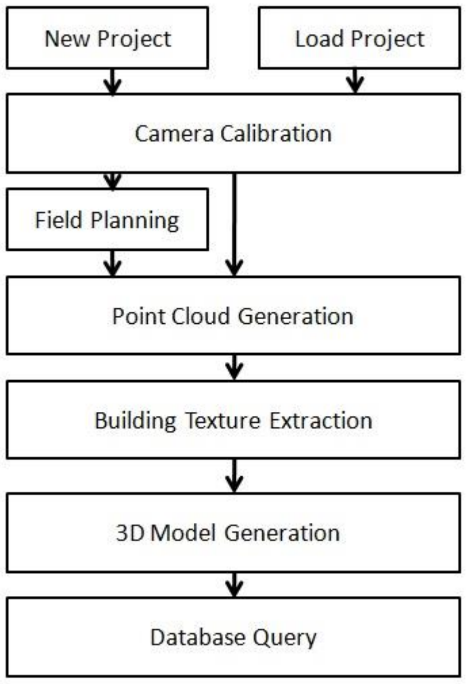


Fig 4.2.1: Steps to Execute for Generation of 3D Model

4.3 EXECUTION OF SOFTWARE

- Execute the NewApplication.py file present in path *Trivim/bin.* Window will pop up of the software which is shown in Fig 4.3.1. If executable file of the software is made then we can directly run the software by double clicking the exe file. The window will pop up of the software.

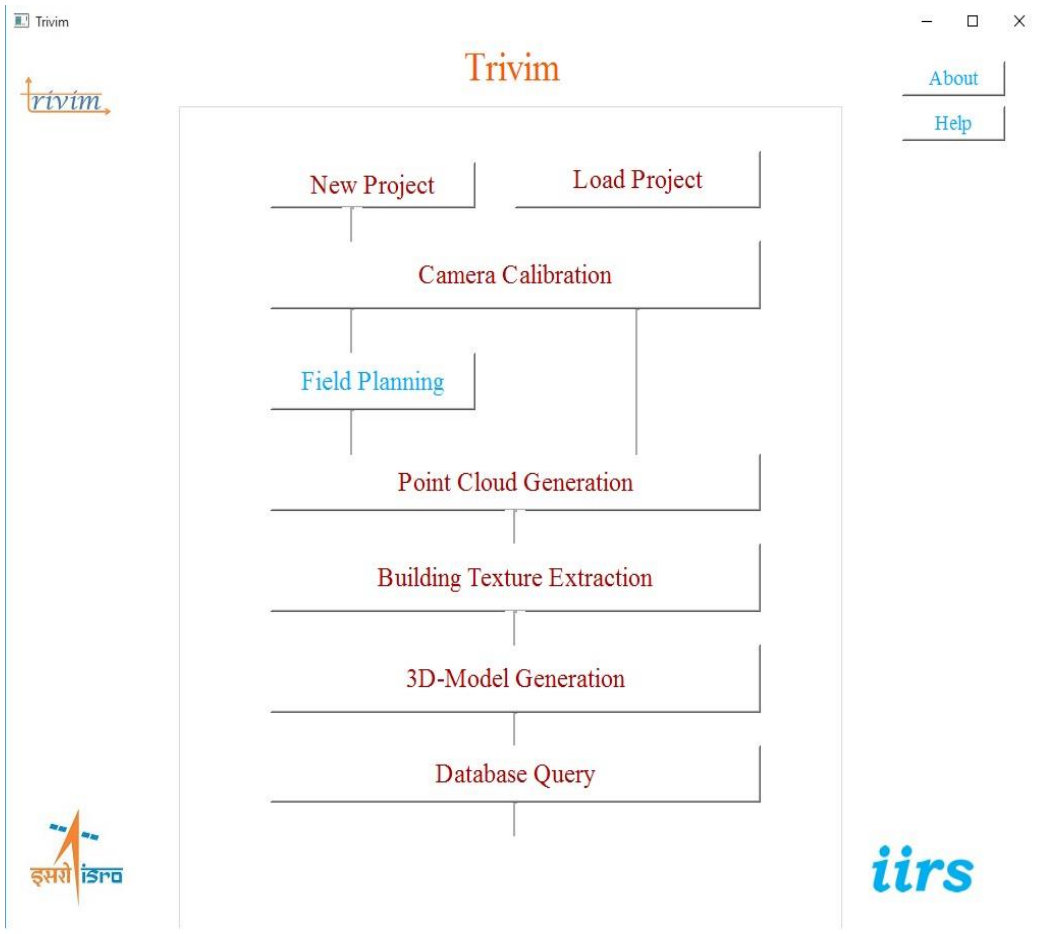
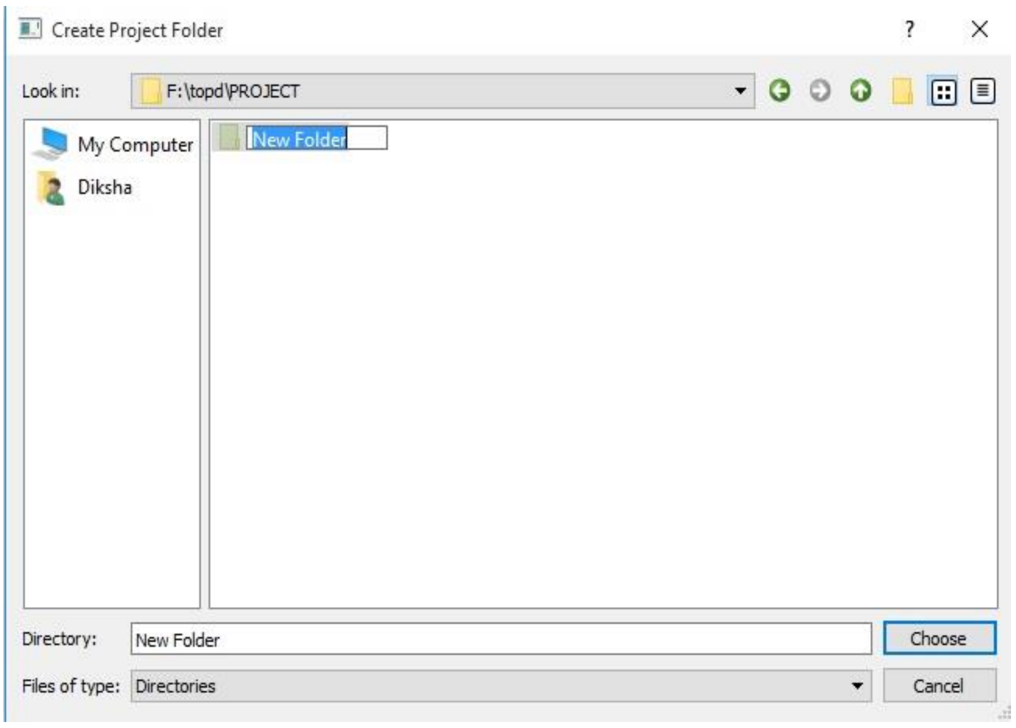


Fig 4.3.1 Main Window

- Buttons present in the window are in red color initially which represent that the process needs to be completed and the blue color shows the optional process (Field Planning Button). When any process is finished, the text color of that button will become green.

4.3.1 CREATING A NEW PROJECT

- Click on the button *New Project* in main window shown in Fig 4.3.1 to create a new project.
- A window will appear asking the name of project folder shown in Fig 4.3.2. Assign name to the folder at required path. The created Folder is the Project folder.



Look in: F:\topd\PROJECT

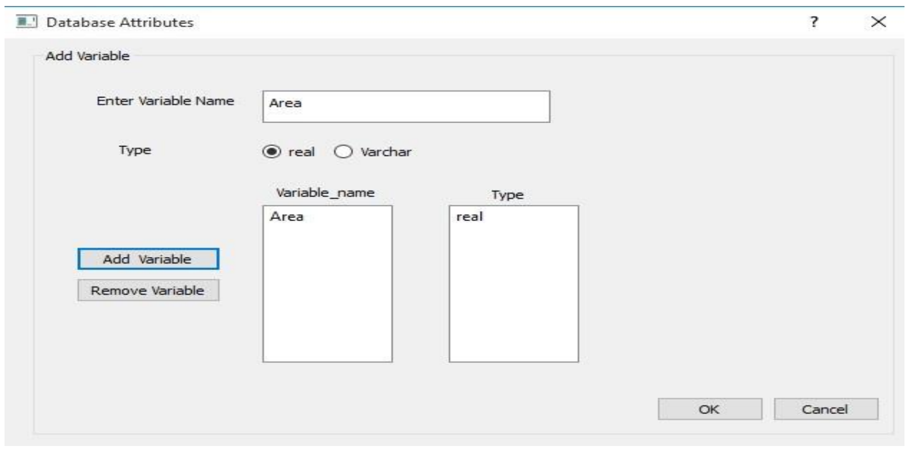
New Folder

Directory: New Folder

Files of type: Directories

Fig 4.3.2: Creating New Project

- After choosing the project another window will appear asking the database attributes shown in Fig



Enter Variable Name: Area

Type: [✓ real] Varchar

Variable_name	Type
Area	real

Fig 4.3.3: Database Attributes

- Enter the database attributes as per user's requirement. (These attributes will be used for Database querying)
- Enter the variable name and choose variable type.
- Click on Add variable to add the attribute to the database.
- Click on Remove variable to delete an attribute from the database.

4.3.2 LOADING AN EXISTING PROJECT

- To load an already existing project. Click on *Load Project* in the main window and choose the Project Folder shown in Fig 4.3.4.

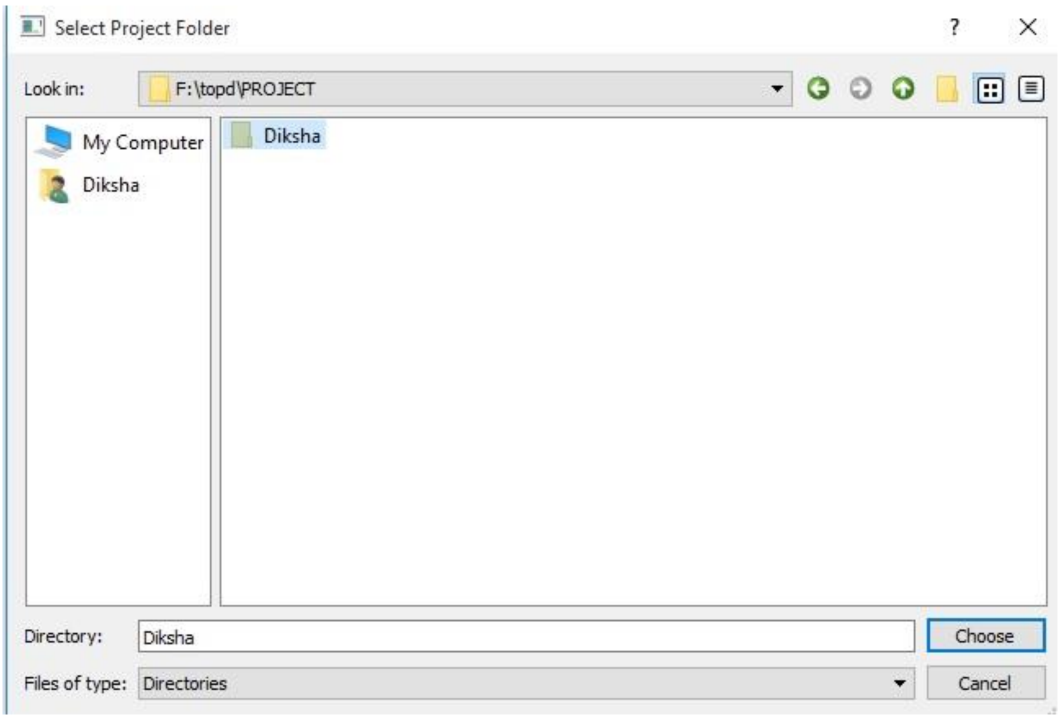


Fig 4.3.4: Loading an Existing Project

4.3.3 CAMERA CALIBRATION

4.3.3.1 Introduction to Calibration sheet

First important step for photogrammetry is camera calibration. Camera calibration is very important in 3D Model generation. Camera calibration is a necessary step to achieve accuracy and remove distortion in images. To generate 3D model we need some camera parameters, these parameters are determined using some calculations, this process of determining parameters through calculation is known as camera calibration [19].We use OpenCV for camera calibration. OpenCV supports three types of objects for calibration:

- Classical black-white chessboard showing in Fig 4.3.5.
- Symmetrical circle pattern
- Asymmetrical circle pattern

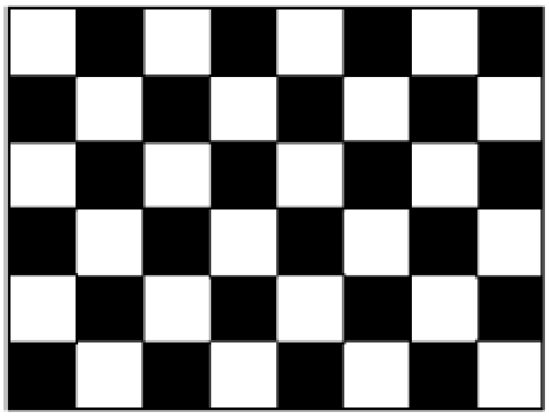


Fig 4.3.5: Classical Black-White Chessboard

Here to achieve camera calibration we use ‘Calibration Sheet’ in the form of chessboard image. The image should be in grey scale and this sheet is printed. The photographs of this sheet are being captured by the camera from different angles or directions i.e., from top, side view with different angles. These captured images of chessboard are used for calculating camera parameters. Camera parameters are of two type’s internal and external parameters. Internal parameters gives the internal optical and geometry characteristics of camera like focal length, image center, lens distortion etc and external parameters are three dimensional position and orientation of camera coordinates system compared with real world coordinates. The camera model of calibration algorithm in OpenCV is based on pinhole model and introduces the radial lens distortion and tangential distortion [19].

4.3.3.2 Capturing Camera Calibration Sheet

We need camera calibration sheet i.e. chessboard sheet for performing camera calibration. To capture set of photographs some preparations are required to provide adequate lighting and background conditions as well as to determine camera positions. Steps need to be followed to capture chessboard images:

- Choosing camera parameters like fixing focal length of the camera.
- Selecting a location to capture photographs, such that sufficient space should be there around calibration sheet, sufficient illumination of calibration sheet and free from any disturbances like wind etc.
- Set-Up of calibration sheet at location.
- Capturing set of photographs of calibration sheet from different angles.

While capturing the photographs of calibration sheet, the sheet should be stable i.e. it should remain stationary at one position and the photos are captured in two different camera orientations - landscape and profile, from four different locations around the sheet showing in Fig 4.3.6 [5].

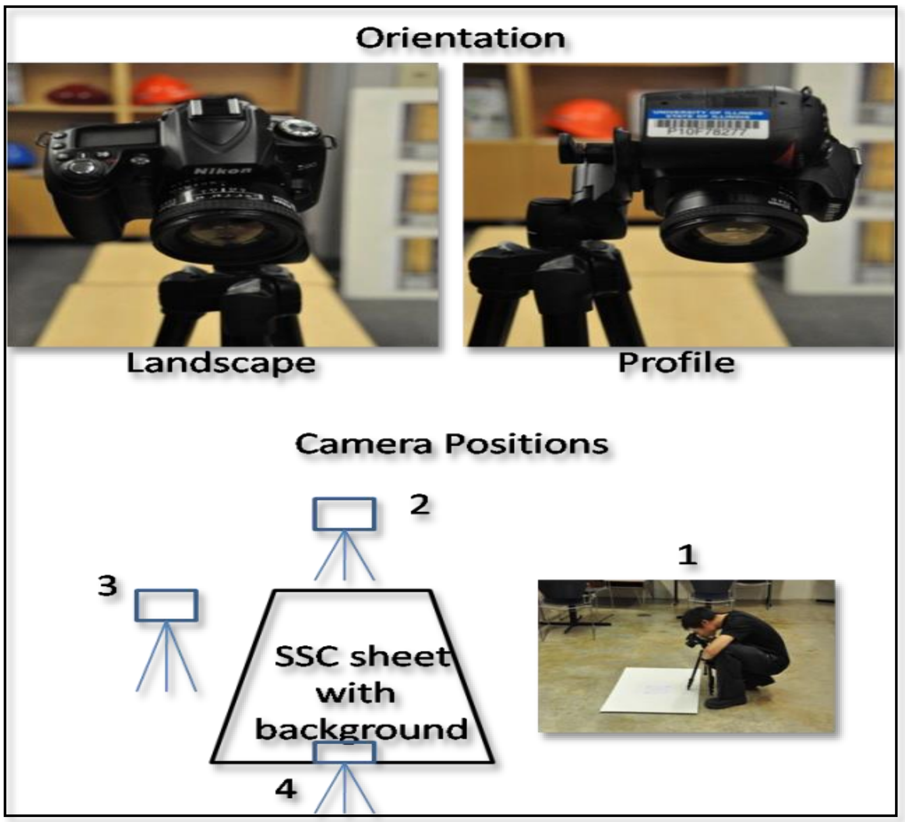


Fig 4.3.6: Camera Orientation and Positions

Captured Grey scale images shown in Fig 4.3.7 of calibration sheet are stored in a separate folder and will be used as an input to the camera calibration step. The processing steps of camera calibration are:

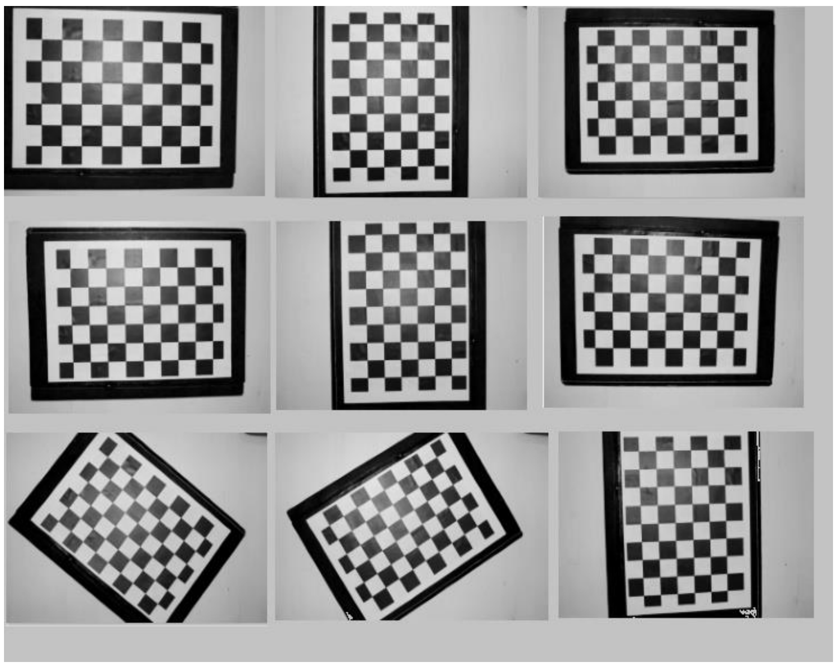


Fig 4.3.7: Captured grey scale images of Chessboard from different angles

- Selection of the camera name from the drop down menu available in select camera name and directory section of the user interface shown in Fig 4.3.8. The camera from which calibration images were taken is selected.
- Click on *Browse* to locate the directory of chessboard images (or directly enter path of directory of chess board images in the provided text field). (*In the sample data, images for calibration are provided in the "gry" folder*)
- Click on the *Calculate Parameters* to perform Camera Calibration.
- If the camera is not listed in the *Camera Type,* then the details of new camera in *Add New Camera* section is entered and after filling the details click on *Add New Camera* button. Perform step 2 and 3 again.
- If the result of Camera Calibration has already been saved in the computer, then click on *Load Camera Parameters* and load the camera file.

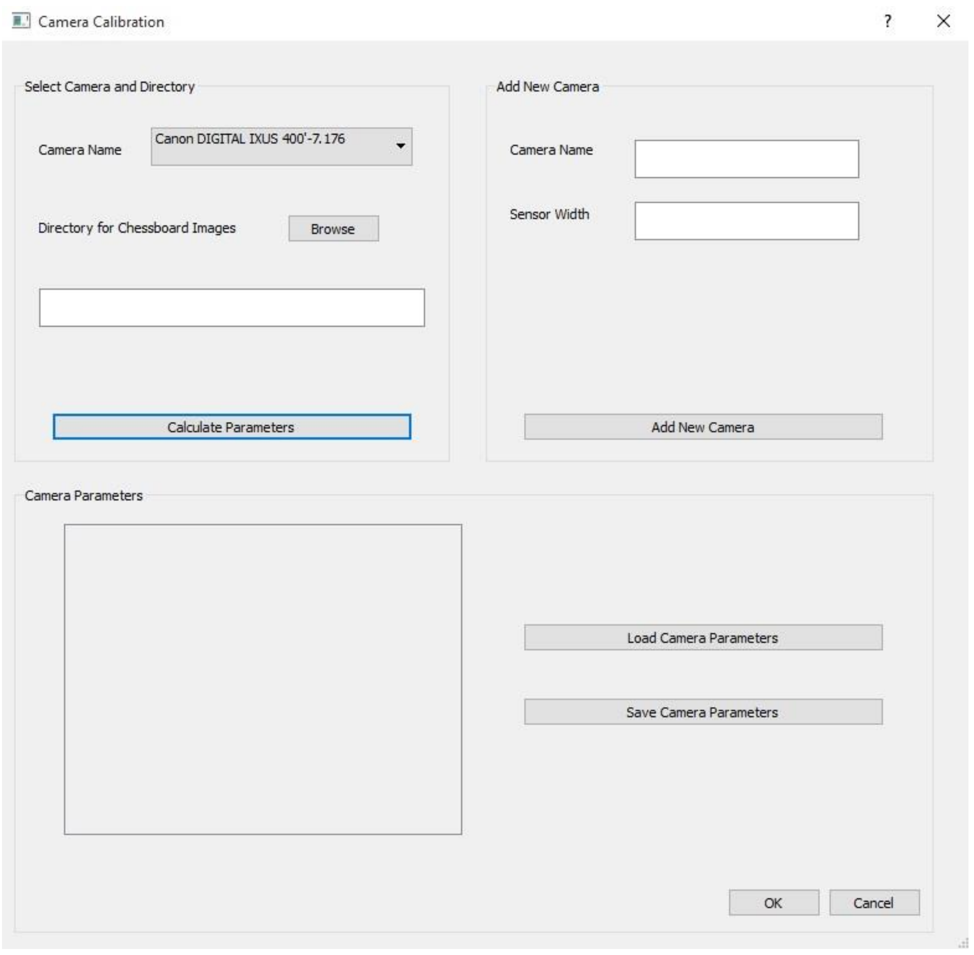


Fig 4.3.8: Camera Calibration Window

- Once the process is finished, results are displayed in the *Camera Parameters* tab. If the results are not satisfactory then click on *Recalculate Parameters* to perform the camera calibration again. (*If required, camera name and chessboard image directory can be chosen again*)
- The camera parameters can also be saved for further use from *Save Camera Parameters*.
- Click on *OK* to complete the process.

4.3.4 IMAGE ACQUISITION

Image acquisition is the step of capturing the 2D images of the object. The field work is done with the pre-calibrated camera. It is necessary to visit the site, take various test photos, estimate time required for preparing, making field notes to study prior to planning. It is necessary to look over all possible working conditions & other site specifics: visibility, weather, equipment, sun/shadows, safety regulations & legal responsibilities [5]. The quality of the captured images should be good as the quality of the output depends on the quality of input provided in the form of images. For good quality input we need photographic skills through which we will be able to take consistently sharp photos in all conditions, for accurate photographs that are suitable for photogrammetry. For suitable photographs certain points should be taken into consideration while acquiring the images [29].

- Use sharpest aperture for lens of the camera (mostly f/8), lens aperture setting can be determined by pixel size of an image.
- The shutter speed should be fast but according to the condition and available light
- Increase ISO as necessary if additional sensitivity is needed in lower lighting conditions
- Use infinity focus of the lens
- Use tripod on stable platform for fixing camera so that no camera movement occur.
- If low light condition occurs, we need to compromise with the optimal exposure settings.

For 3D data acquisition and object reconstruction we need images and for acquiring the image we use close range photogrammetry approach in which the camera is placed closed to the object like building while capturing the photographs. We use the approach of overlapping images. The Overlapped images are shown in Fig 4.3.9.

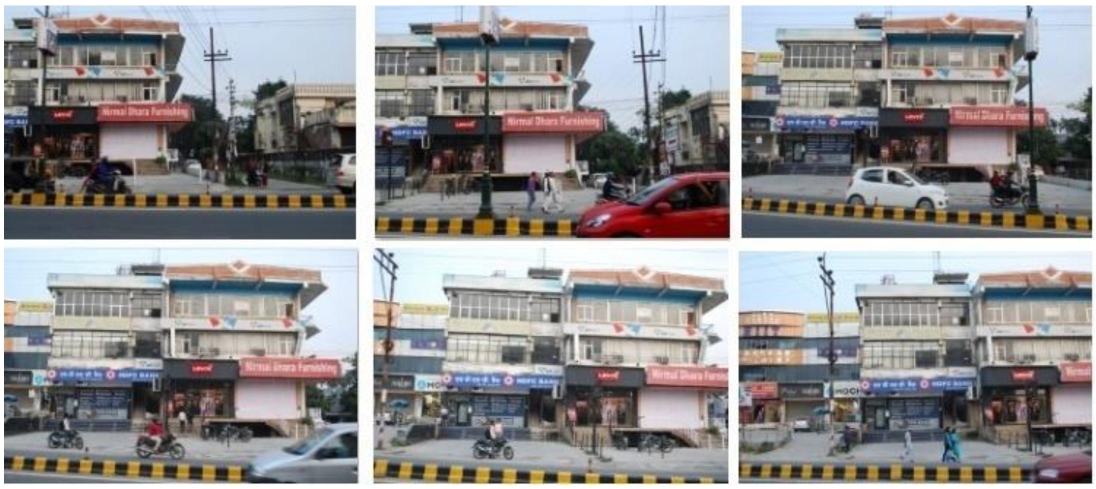


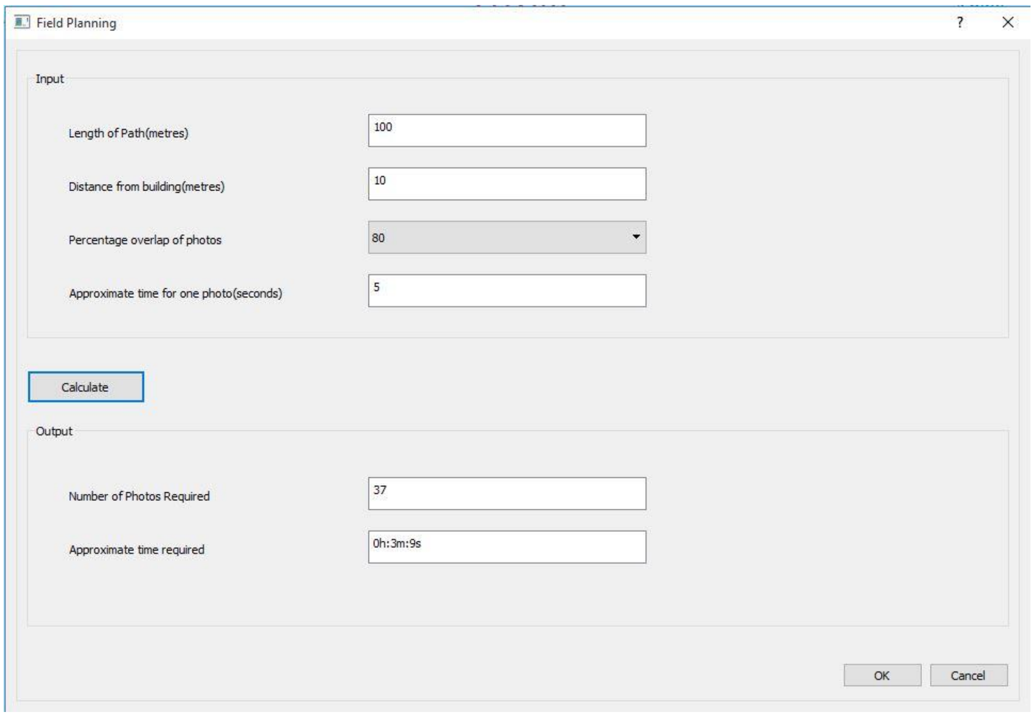
Fig 4.3.9: Overlapped Images

While acquiring through pre-calibrated camera, the captured images should be 60 to 80% overlapped. The images are captured by placing camera on tripod so that camera becomes stable by placing camera closed to the object and the distance is noted. We can use any image capturing devices but preferably D-SLR cameras with over 10 megapixels of resolution for better images and good quality output. We have used Nikon D60 (10.2 Megapixels). The focal length of camera is kept 18mm. The focal length should be kept fixed during entire process. GPS/DGPS device is used for data collection of Georeferencing the point cloud data to real world coordinates.

5.3.5 FIELD PLANNING

Before Image Acquisition, to find how many number of photographs are required of the object and the approximate time required for the photographs, this field planning step can be carried out. To make planning certain steps need to be followed:

- ➢ Click on *Field Planning* button in main menu to open the Field Planning window shown in Fig 4.3.10.



Length of Path(metres) 100
Distance from building(metres) 10
Percentage overlap of photos 80
Approximate time for one photo(seconds) 5
Number of Photos Required 37
Approximate time required 0h:3m:9s

Fig 4.3.10: Field Planning Window

- ➢ Give approximate *Length of Path (meters).*
- ➢ Give approximate *Distance from Building (meters).*
- ➢ Give approximate *Overlap of photos (percentage).*
- ➢ Give approximate *time for one photograph (sec).*
- ➢ Click on *calculate* button.
- ➢ We will receive *Number of Photos Required* and *Approximate Time Required* as output.
- ➢ Click on *OK* to complete the process.

$$n = 1 + \frac{\frac{Length\ of\ the\ Path \times Focal\ Length\ in\ mm}{CCD\ in\ mm\ \times Distance\ From\ Building} - 1}{1 - \frac{Percentage\ of\ Overlap}{100}}$$

Where, n is the number of photographs required.

4.3.6 POINT CLOUD GENERATION

To generate point cloud steps to be followed:

- Click on the *Point Cloud Generation* in the main window to open the Point Cloud Generation window.
- Select the folder which contains images of the object whose point cloud is to be generated. Then click on *Generate Point Cloud* button.
- Processing of generation of point cloud will start.
- First, SIFT will run and it will extract features from the images and then matching of the images will be performed.
- Second, Bundler will run which will create sparse point cloud from the images.
- Thirdly, CMVS will run and decomposes the input images into a set of image clusters of manageable size and then PMVS will reconstruct the 3D structure of an object by creating dense point clouds from images.
- When the button *Generate Point Cloud* will become Green then this step is completed shown in Fig 4.3.11.

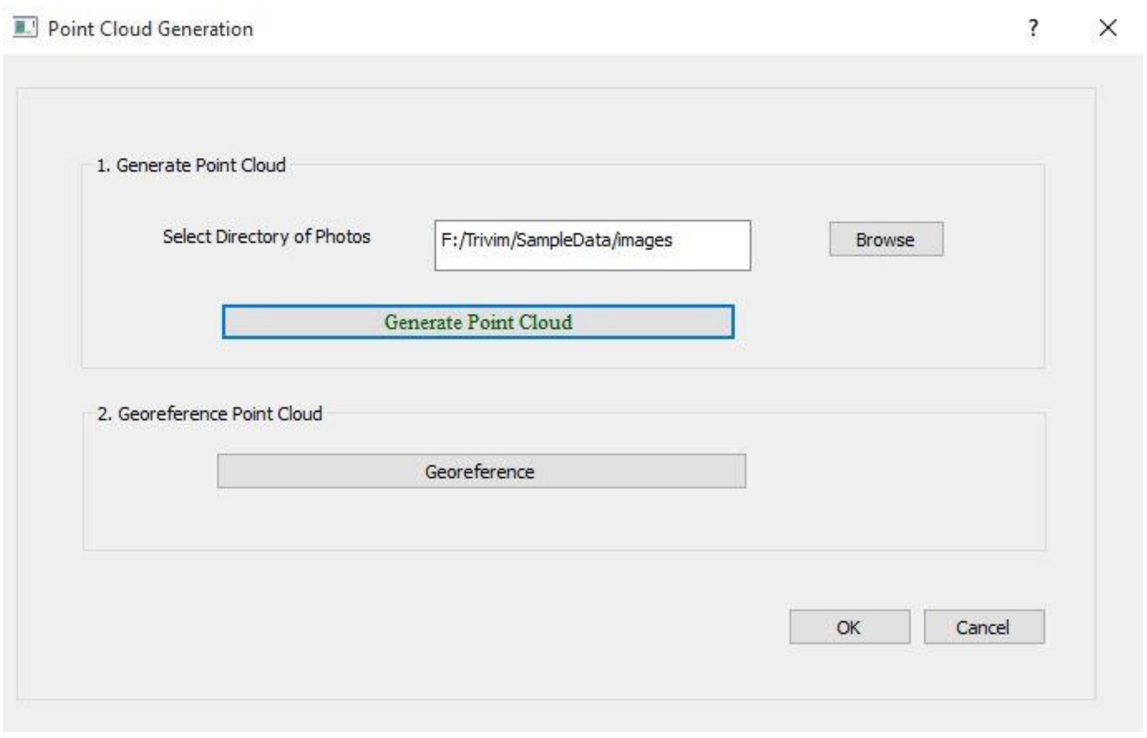


Fig 4.3.11: Point Cloud Generation Window

Note: Processing time of this step depends on number of images and size of images chosen for point cloud generation. All the files of this step of Point Cloud Generation is present in Point Cloud folder inside the project folder. In bundle folder inside Point Cloud folder consist of the files created by bundler and in PMVS folder the files are created by PMVS. These created files can be viewed with the help of Cloud Compare software.

4.3.7 GEOREFERENCING

Geo-referencing explains how position data (e.g., Global Positioning System (GPS) locations) relate to imagery and to a physical location. It helps to find out the relation between the Local Coordinate System and the Global Coordinate system.

To Georeference steps to be followed are:

➢ Click on *Georeference* button for Georeferencing the generated point cloud shown in Fig 4.3.12.

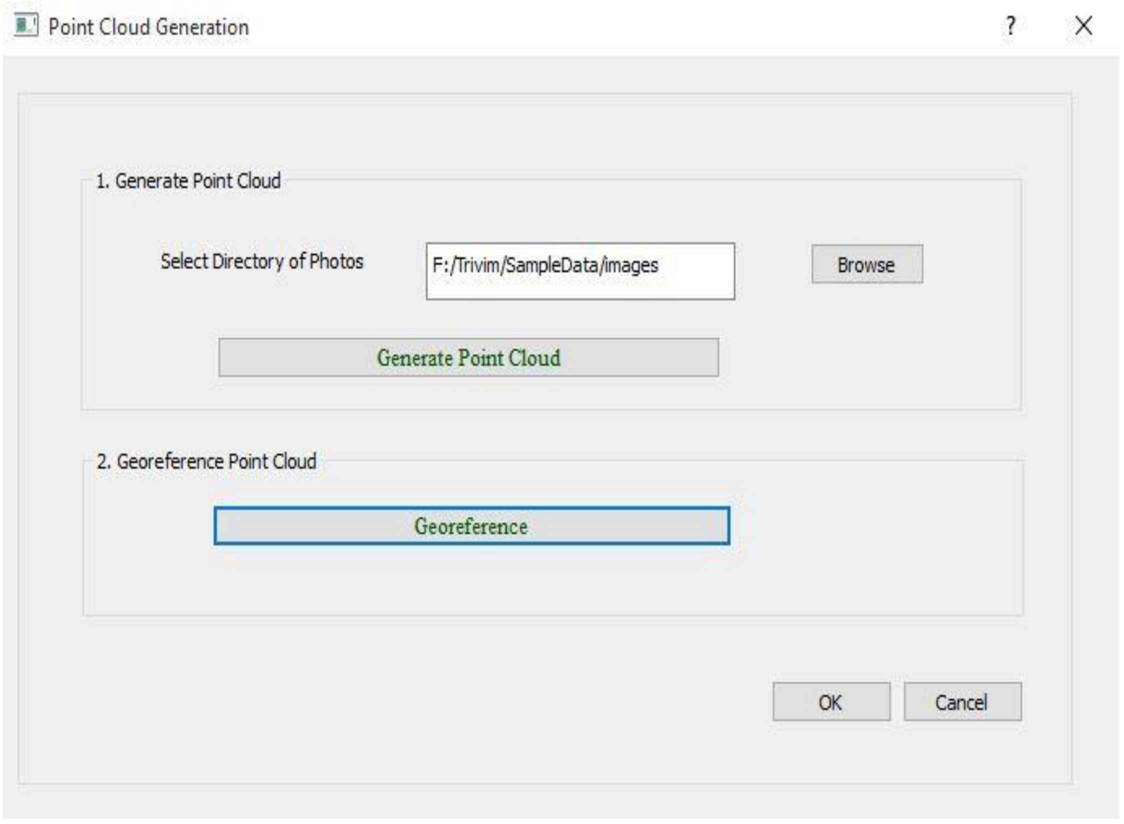


Fig 4.3.12: Georeferencing Window

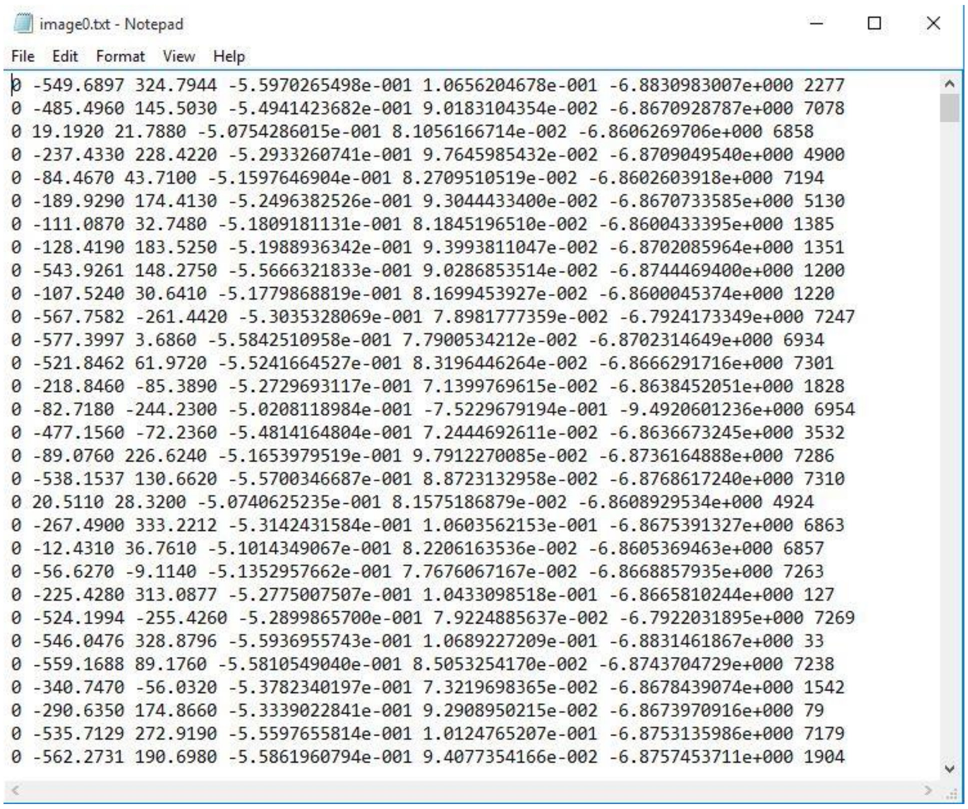
- ➢ If the images are already geo-tagged then a file named *georeffile.txt* will be automatically created in the *Point Cloud* folder inside the project folder otherwise a browsing window will appear asking to browse the file containing the global coordinates of the exposure station points(*User should provide the coordinates file if the images are not geo-tagged*).
- ➢ This process will give *georeffile.txt* as the output file. This georeferenced file will be saved inside the *Point Cloud* folder.
- ➢ In this process GPS coordinates of each image with the pixel values are stored separately in the file named *'imagenewN.tx*t' shown in Fig 4.3.14, where N is the image number in *Image Coor* folder.

This process of Georeferencing each image is done by following certain steps:

- File named *bundle.out* is taken as input. First number of images is extracted from the *bundle.out.*
- Then (Number of photographs*5+1) lines are skipped and then each row is read.
- Row containing RGB is skipped. Rest rows data are stored according to images in respective image files.

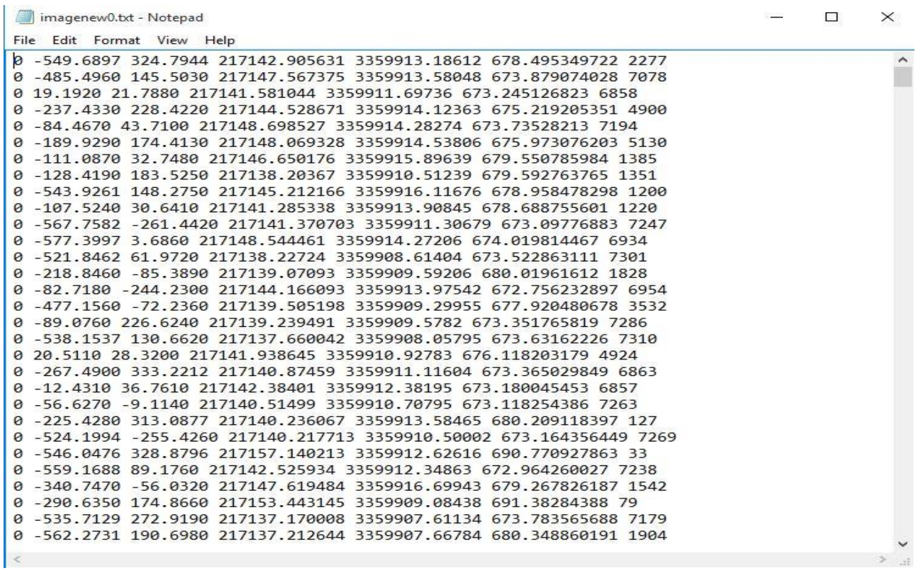
- Files of each image are created and data present in row of corresponding image is stored in its file. Data entered in file is x and y coordinate along with the 3D location of the point.
- Through this process, the files for each image are created from *bundle.out*.

Here in Fig 4.3.13 File of image no 0 is shown.



```
0 -549.6897 324.7944 -5.5970265498e-001 1.0656204678e-001 -6.8830983007e+000 2277
0 -485.4960 145.5030 -5.4941423682e-001 9.0183104354e-002 -6.8670928787e+000 7078
0 19.1920 21.7880 -5.0754286015e-001 8.1056166714e-002 -6.8606269706e+000 6858
0 -237.4330 228.4220 -5.2933260741e-001 9.7645985432e-002 -6.8709049540e+000 4900
0 -84.4670 43.7100 -5.1597646904e-001 8.2709510519e-002 -6.8602603918e+000 7194
0 -189.9290 174.4130 -5.2496382526e-001 9.3044433400e-002 -6.8670733585e+000 5130
0 -111.0870 32.7480 -5.1809181131e-001 8.1845196510e-002 -6.8600433395e+000 1385
0 -128.4190 183.5250 -5.1988936342e-001 9.3993811047e-002 -6.8702085964e+000 1351
0 -543.9261 148.2750 -5.5666321833e-001 9.0286853514e-002 -6.8744469400e+000 1200
0 -107.5240 30.6410 -5.1779868819e-001 8.1699453927e-002 -6.8600045374e+000 1220
0 -567.7582 -261.4420 -5.3035328069e-001 7.8981777359e-002 -6.7924173349e+000 7247
0 -577.3997 3.6860 -5.5842510958e-001 7.7900534212e-002 -6.8702314649e+000 6934
0 -521.8462 61.9720 -5.5241664527e-001 8.3196446264e-002 -6.8666291716e+000 7301
0 -218.8460 -85.3890 -5.2729693117e-001 7.1399769615e-002 -6.8638452051e+000 1828
0 -82.7180 -244.2300 -5.0208118984e-001 -7.5229679194e-001 -9.4920601236e+000 6954
0 -477.1560 -72.2360 -5.4814164804e-001 7.2444692611e-002 -6.8636673245e+000 3532
0 -89.0760 226.6240 -5.1653979519e-001 9.7912270085e-002 -6.8736164888e+000 7286
0 -538.1537 130.6620 -5.5700346687e-001 8.8723132958e-002 -6.8768617240e+000 7310
0 20.5110 28.3200 -5.0740625235e-001 8.1575186879e-002 -6.8608929534e+000 4924
0 -267.4900 333.2212 -5.3142431584e-001 1.0603562153e-001 -6.8675391327e+000 6863
0 -12.4310 36.7610 -5.1014349067e-001 8.2206163536e-002 -6.8605369463e+000 6857
0 -56.6270 -9.1140 -5.1352957662e-001 7.7676067167e-002 -6.8668857935e+000 7263
0 -225.4280 313.0877 -5.2775007507e-001 1.0433098518e-001 -6.8665810244e+000 127
0 -524.1994 -255.4260 -5.2899865700e-001 7.9224885637e-002 -6.7922031895e+000 7269
0 -546.0476 328.8796 -5.5936955743e-001 1.0689227209e-001 -6.8831461867e+000 33
0 -559.1688 89.1760 -5.5810549040e-001 8.5053254170e-002 -6.8743704729e+000 7238
0 -340.7470 -56.0320 -5.3782340197e-001 7.3219698365e-002 -6.8678439074e+000 1542
0 -290.6350 174.8660 -5.3339022841e-001 9.2908950215e-002 -6.8673970916e+000 79
0 -535.7129 272.9190 -5.5597655814e-001 1.0124765207e-001 -6.8753135986e+000 7179
0 -562.2731 190.6980 -5.5861960794e-001 9.4077354166e-002 -6.8757453711e+000 1904
```

Fig 4.3.13: Point Cloud of Images before Georeferencing



```
0 -549.6897 324.7944 217142.905631 3359913.18612 678.495349722 2277
0 -485.4960 145.5030 217147.567375 3359913.58048 673.879074028 7078
0 19.1920 21.7880 217141.581044 3359911.69736 673.245126823 6858
0 -237.4330 228.4220 217144.528671 3359914.12363 675.219205351 4900
0 -84.4670 43.7100 217148.698527 3359914.28274 673.73528213 7194
0 -189.9290 174.4130 217148.069328 3359914.53806 675.973076203 5130
0 -111.0870 32.7480 217146.650176 3359915.89639 679.550785984 1385
0 -128.4190 183.5250 217138.20367 3359910.51239 679.592763765 1351
0 -543.9261 148.2750 217145.212166 3359916.11676 678.958478298 1200
0 -107.5240 30.6410 217141.285338 3359913.90845 678.688755601 1220
0 -567.7582 -261.4420 217141.370703 3359911.30679 673.09776883 7247
0 -577.3997 3.6860 217148.544461 3359914.27206 674.019814467 6934
0 -521.8462 61.9720 217138.22724 3359908.61404 673.522863111 7301
0 -218.8460 -85.3890 217139.07093 3359909.59206 680.01961612 1828
0 -82.7180 -244.2300 217144.166093 3359913.97542 672.756232897 6954
0 -477.1560 -72.2360 217139.505198 3359909.29955 677.920480678 3532
0 -89.0760 226.6240 217139.239491 3359909.5782 673.351765819 7286
0 -538.1537 130.6620 217137.660042 3359908.05795 673.63162226 7310
0 20.5110 28.3200 217141.938645 3359910.92783 676.118203179 4924
0 -267.4900 333.2212 217140.87459 3359911.11604 673.365029849 6863
0 -12.4310 36.7610 217142.38401 3359912.38195 673.180045453 6857
0 -56.6270 -9.1140 217140.51499 3359910.70795 673.118254386 7263
0 -225.4280 313.0877 217140.236067 3359913.58465 680.209118397 127
0 -524.1994 -255.4260 217140.217713 3359910.50002 673.164356449 7269
0 -546.0476 328.8796 217157.140213 3359912.62616 690.770927863 33
0 -559.1688 89.1760 217142.525934 3359912.34863 672.964260027 7238
0 -340.7470 -56.0320 217147.619484 3359916.69943 679.267826187 1542
0 -290.6350 174.8660 217153.443145 3359909.08438 691.38284388 79
0 -535.7129 272.9190 217137.170008 3359907.61134 673.783565688 7179
0 -562.2731 190.6980 217137.212644 3359907.66784 680.348860191 1904
```

Fig 4.3.14: Point Cloud of Images after Georeferencing

- After creating each file for all images like *image0.txt* shown in Fig 4.3.13. Each image is georeferenced.
- A new file named *imagenew0.txt* is created from *image0.txt* by replacing 3D local coordinate of the image to 3D real world coordinate by matching row number with the georeferenced file row number. Through this process each image coordinate is georeferenced.

4.3.8 BUILDING TEXTURE EXTRACTION

This step consists of two steps i.e. Image Segmentation and Height Extraction. While cropping the image we get the height of the image. Sequence of images is captured through pre-calibrated camera of a textured object that is to be reconstructed. Texture on the images plays a vital role in analyzing the object. We are considering 3D buildings as object; buildings have same structure, shape or geometry. This step performs editing, assignment of object textures and real world height extraction of each building segment. This step makes the user to create building segments which can be used as unit for database creation and query and calculate the real world height of each segment. These texture and height are needed for 3D model generation. The height of the building segments are calculated by making the captured 2D images georeferenced. This process of converting the image coordinates into real world coordinates are performed by applying transformation. The rescaled images of original photos, those are present in the Point Cloud folder of the project are chosen, segment of each floor are created and the height of the segments are calculated. To perform this step steps to be followed are:

- ➢ Click on *Building Texture Extraction* button in main window to perform this task.
- ➢ *Image Segmentation* window will pop up.
- ➢ Now to start segmentation select the desired image from the drop down menu *Select Image* shown in Fig 4.3.15.

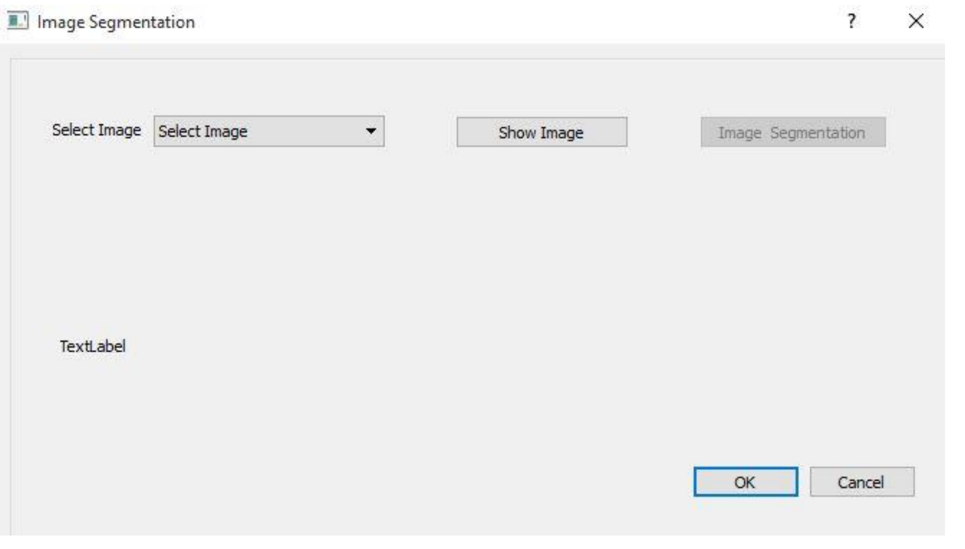


Fig 4.3.15: Image Segmentation Window

- ➢ After selecting the image, click on *Show Image* button to view the selected image. The selected image will be displayed on the *Image Preview* tab.
- ➢ Click on Image Segmentation button to perform building texture extraction and height extraction. A dialog with image pops up shown in Fig 4.3.16.

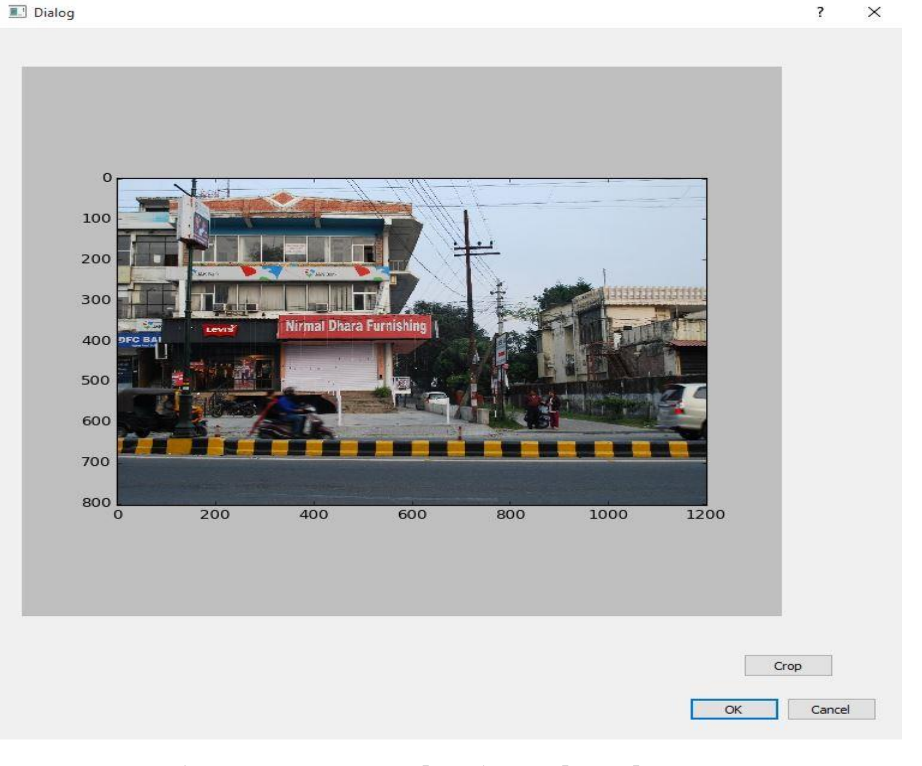


Fig 4.3.16: Popup showing Selected Image

- ➢ With this click on one corner of the desired segment and draw complete segment while holding the left click button (don't release the button before completing the quadrant and ensure that the segment is closed). The segment is drawn in 'green' color. This has been done using *matplotlib* library.

- When segment is drawn we get 2 coordinates of the box drawn i.e 1st coordinate of the rectangle and the third coordinate of the rectangle shown in Fig 4.3.17.

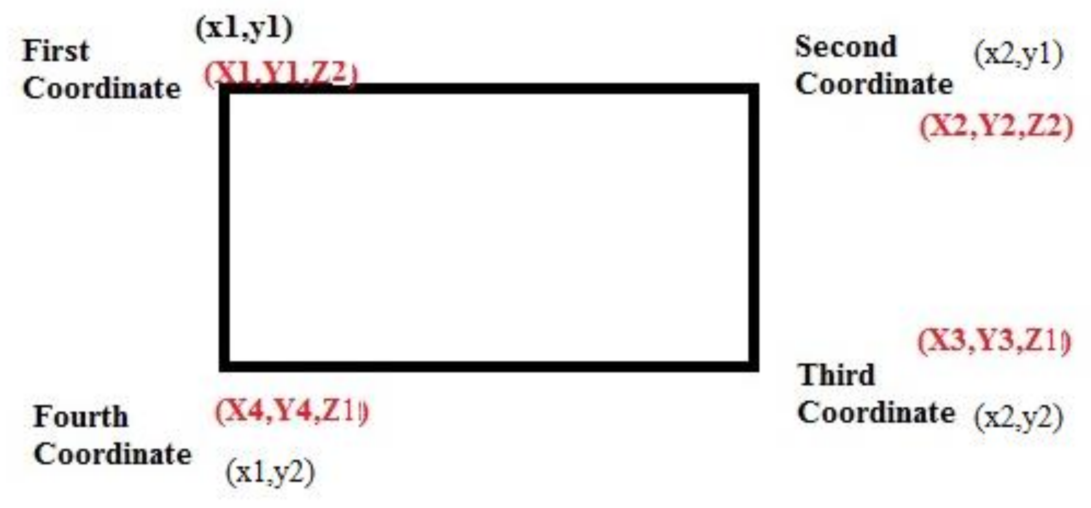


Fig 4.3.17: Segment drawn on the image

- To calculate the height we need the difference between the real world coordinate of the two points i.e. difference between the altitude of the second and third point. But the points are pixel values and they should be georeferenced.

Height = (Z2 – Z1)

- The points of the particular image are georeferenced through transformation. To perform transformation we need 2d matrix of the pixels and 3d matrix of the 3d location of the image selected in image segmentation. We get these matrixes from *imagenewN.txt* of the corresponding image.

  Suppose we have data from file:

  2d matrix =

```
[     [-549.6897 -485.496   19.192  ..., -408.461   161.154   -1.994]
      [ 324.7944  145.503   21.788  ...,  184.059   240.371   42.697 ]
      [   1.         1.        1.     ...,    1.        1.        1.    ]                        ]
```

  3d matrix =

```
[         [2.17148407e+05          2.17145195e+05          2.17136595e+05   ...,
2.17144134e+05
   2.17129088e+05   2.17136822e+05]
                 [3.35991607e+06       3.35991455e+06       3.35990680e+06  ...,
3.35991357e+06
   3.35991274e+06   3.35990731e+06]
            [6.81254477e+02         6.75466729e+02         6.73629096e+02   ...,
6.76745496e+02
   6.72361978e+02   6.73965639e+02]                         ]
```

- After receiving 2d and 3d matrix, we apply superimposition matrix function in python on both matrix and we get the matrix Tr as output.

Tr = trans.superimposition_matrix (**2d matrix**, **3d matrix**)

Tr =

```
[[-9.58620914e-02 -1.36532855e-01 -3.79162243e-02   2.17129152e+05]
 [-1.36413148e-01   1.01308061e-01 -1.99130867e-02   3.35986091e+06]
 [3.83446682e-02   1.90750493e-02 -1.65632817e-01   6.85601747e+02]
 [0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00]]
```

- To make the pixel of the image georeferenced, we multiply the output matrix with the pixel to make it georeferenced.

Georeferenced coordinate at pixel = Pixel * Tr

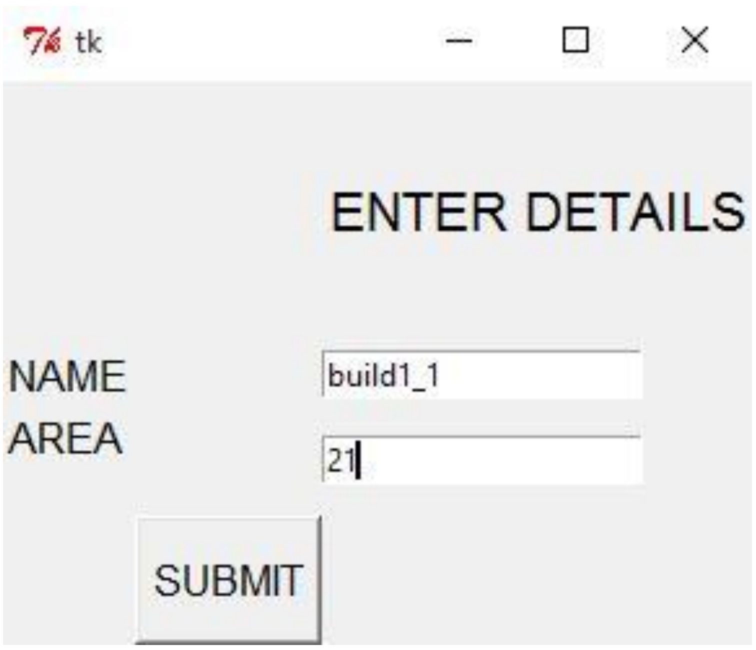
X3, Y3, Z1 = (x2, y2) * Tr

X2, Y2, Z2 = (x2, y1) * Tr

- Both coordinates are georeferenced and the difference in the coordinate is the height of the building.

Height = (Z2 – Z1)

➢ When the segment is drawn, the real world height is extracted of the segment. After drawing the segment, click on *Crop* button in the Segmentation window to save the drawn segment. A window will open to enter the database details of the building segments shown in Fig 4.3.18.


tk

ENTER DETAILS

NAME build1_1

AREA 21

Fig 4.3.18: Saving a segment in Database

- The nomenclature is fix and it should be follow for naming the segment:
  buildM_N
  Where, M is the number of building and N is the floor number of that particular building taking N=1 for ground floor and increase the value of N for upper floors.
  e.g.: for building no: 1 and first floor will be build1_2.
  Then click Submit button after each entry.
- Repeat the same procedure for all the required segments in the image.
- After performing all the segmentation and height extraction in an image, click *OK* button to complete the segmentation and height extraction process for the selected image.
- Repeat the above procedure for others image also if required.
- Click *OK* to complete the process.

From this step we will have the folders created in input folder with the name of buildings (build1, build2 …) with the files of segment image, height extracted file of the building.

4.3.9 3D MODEL GENERATION

This step compiles the planimetry, elevation and attributes information related to the building segment and creates a 3D model. To generate 3D model certain steps need to be follow:

- Click on '*3D Model Generation*' in main window to open the 3D Model Generation window shown in Fig 4.3.19.

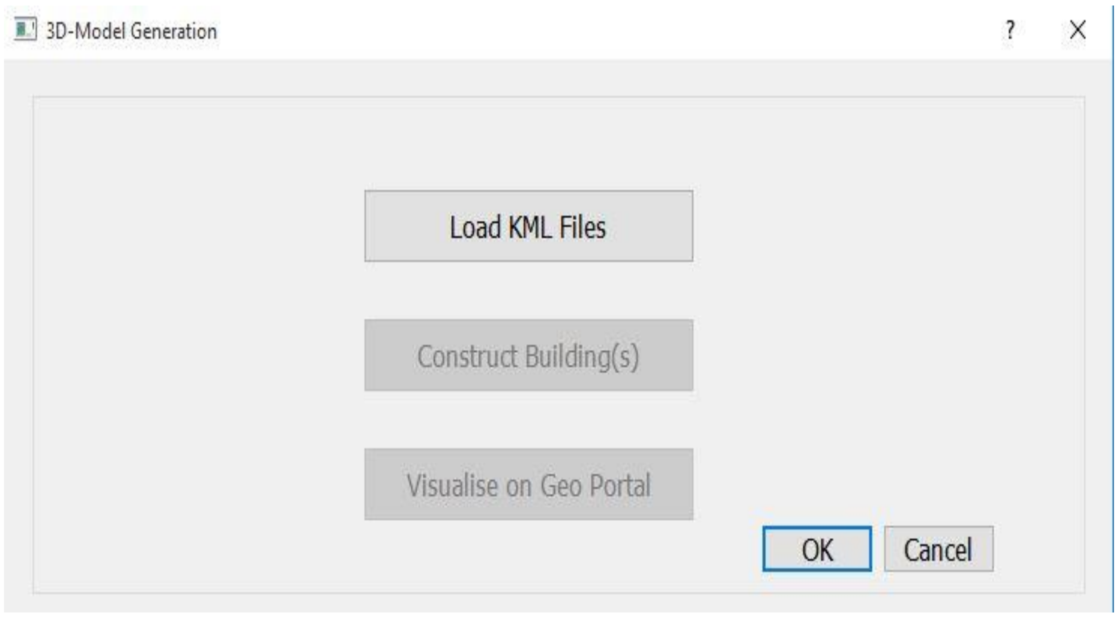


Fig 4.3.19: 3D Model Generation Window

- Now first click on '*Load KML Files*' button to import the footprint (in KML format) of the building to the project folder. A window will open which will allow the user to select the footprint files (in .kml format) for respective buildings shown in Fig 4.3.20. Click Ok after selecting the KML files.

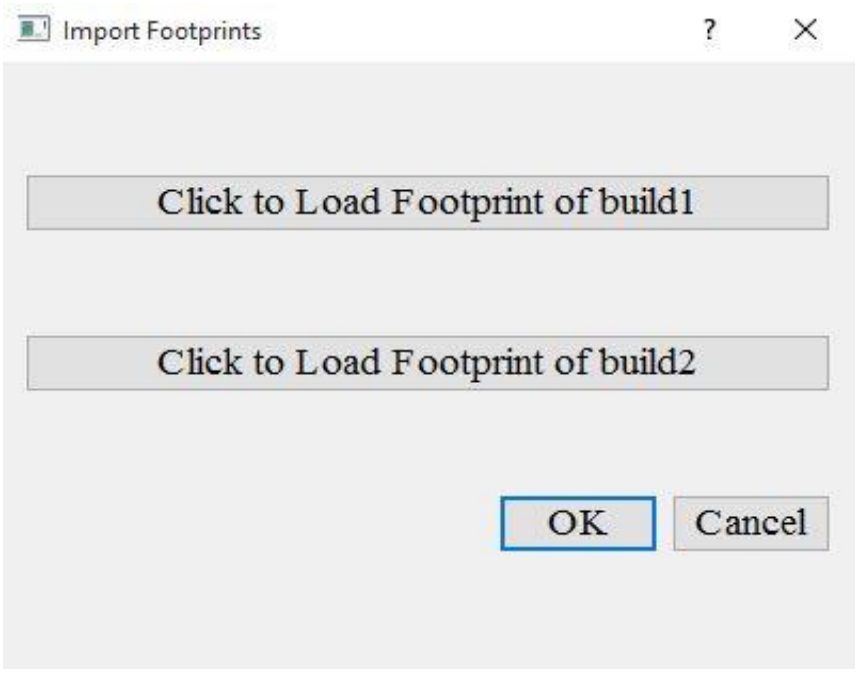

Import Footprints
?
Click to Load Footprint of build1
Click to Load Footprint of build2
OK
Cancel

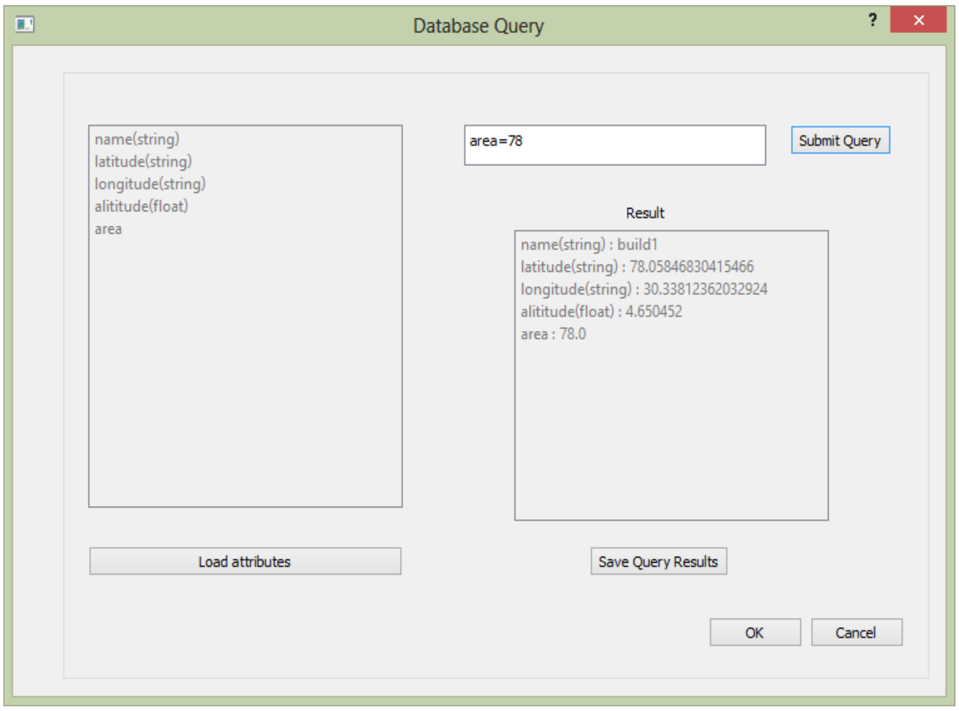
Fig 4.3.20: Import Footprints Model

    - Here footprint is the polygon of the area where the building is located. It should be in the ".kml" file format. The name of the footprint files should be as per the name of the building. For example: The footprint of the first building should be names as "build1.kml", "build2.kml" for second building.

- After loading KML files click on '*Construct Building*' button. It constructs the building by making the dae model and kml files.
- When '*Construct Building*' button becomes green then click on '*Visualize on Geo Portal*' button to visualize the 3D Model. Here for visualization Google Earth is used.
- Click *OK* to complete the step.

4.3.10 DATABASE QUERY

Database query tab is used for querying the database containing the information of all the floors of the buildings shown in Fig 4.3.21.



```
name(string)
latitude(string)
longitude(string)
alititude(float)
area
```

```
area=78
```

```
name(string) : build1
latitude(string) : 78.05846830415466
longitude(string) : 30.33812362032924
alititude(float) : 4.650452
area : 78.0
```

Fig 4.3.21: Database Query Tab

In database query certain tasks are performed:

- *Load Attributes* button will display the attributes of the database of the current loaded project.
- *Submit Query* button will process the query entered by the user and display the results.
- *Save Query Results* button will enable the user to save the query results in a ".txt" file in a directory.
- Click on *OK* to complete the process.

CHAPTER V

RESULTS AND EVALUATION

5.1 INTRINSIC CAMERA PARAMETERS OF CAMERA CALIBRATION

Camera Calibration was done by using an application. Galaxy Core (geotagged) and Nikon D60 SLR (single lens reflex) digital camera was used here. It is a non-metric camera and has to be calibrated for accurate data extraction and to remove distortion in images. For calibration we have use 'Calibration Sheet' in the form of chessboard image. The image is in grey scale and this sheet is printed. The photographs of this sheet are being captured by the camera with fixed distance and focal length of camera with different orientation angles. These captured grey scale images of chessboard are used as input for the camera calibration step. The output file of the two cameras containing camera parameters is:

Table 5.1: Calibration Result of Galaxy Core

Camera Name	Galaxy Core
Sensor Width	5.04
Root Mean Square(RMS) value	1.09963578238
Distortion Coefficients	6.36913262552 -1411.38802704 -0.065163696599 0.0150520938288 8863.38461556
Camera Matrix	[5181.14701843 0. 415.15944591] [0. 4699.91724059 294.12677157] [0. 0. 1.]

Focal Length(mm)	6.74405500332

Table 5.2: Calibration Result of Nikon D60 SLR

Camera Name	NIKON D60
Sensor Width	23.6
Root Mean Square(RMS) value	1.35122690714
Distortion Coefficients	-0.0847703958879 -0.0168466542622 -0.000599890045026 -0.000693130784012 0.49197728215
Camera Matrix	[3242.82321693 0. 1861.58401616] [0. 3242.49381842 1291.96309646] [0. 0. 1.]
Focal Length(mm)	19.7651415082

5.2 IMAGE ACQUISITION AND GPS POINT ACQUISITION

We have used two types of camera for image acquisition. Galaxy Core is geo-tagged camera that automatically stores the GPS coordinates of the image capturing positions. Nikon D60 DSLR camera of focal length 18mm and 10.2 megapixels is used for capturing images and Trimble R7 DGPS is used to measure position of exposure station. As the camera Nikon D60 DSLR is not geo-tagged, so the GPS points have to be manually collected of each image capturing position. Captured overlapped images by the camera Nikon D60 SLR are shown in Fig 5.1.

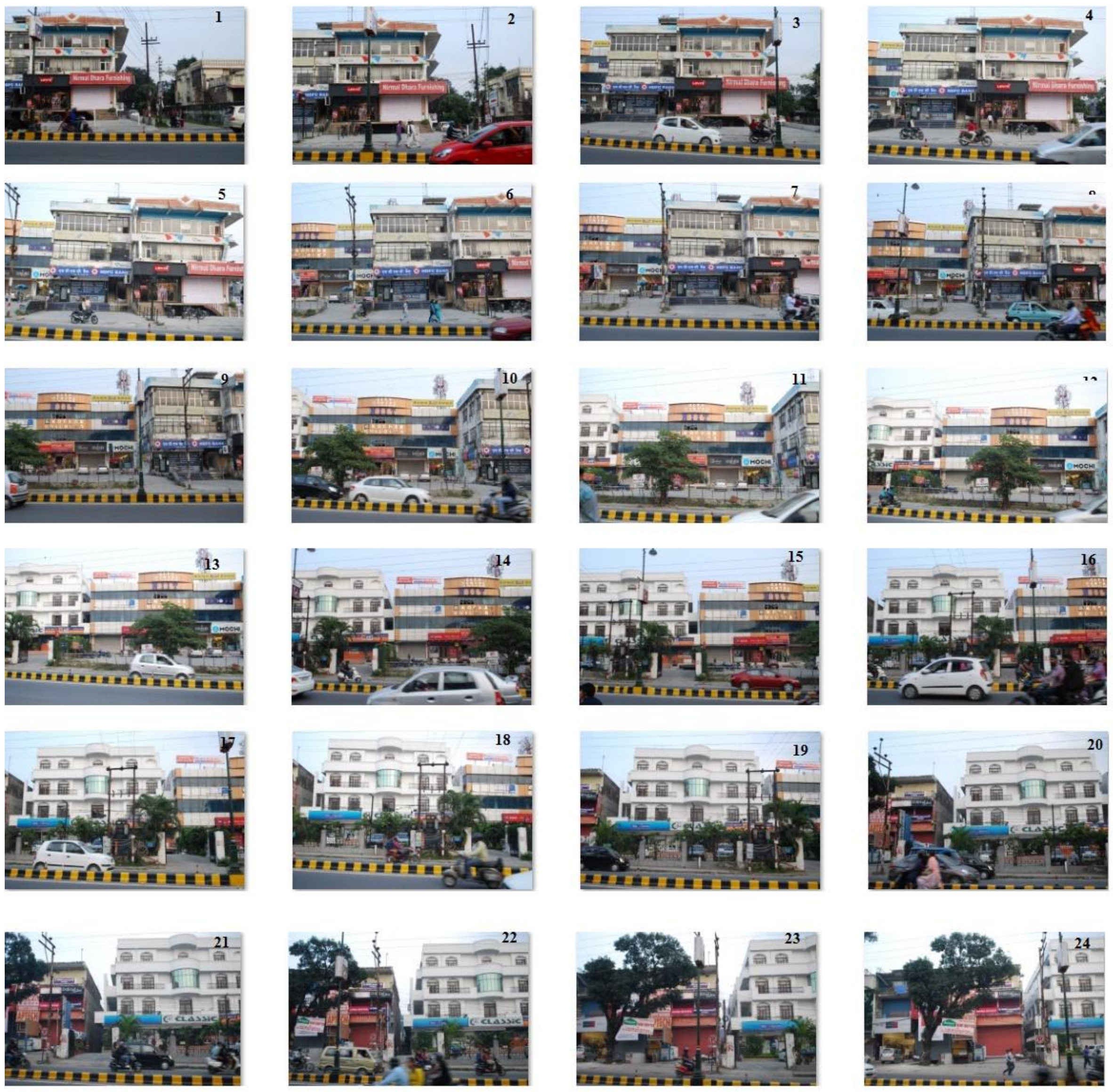
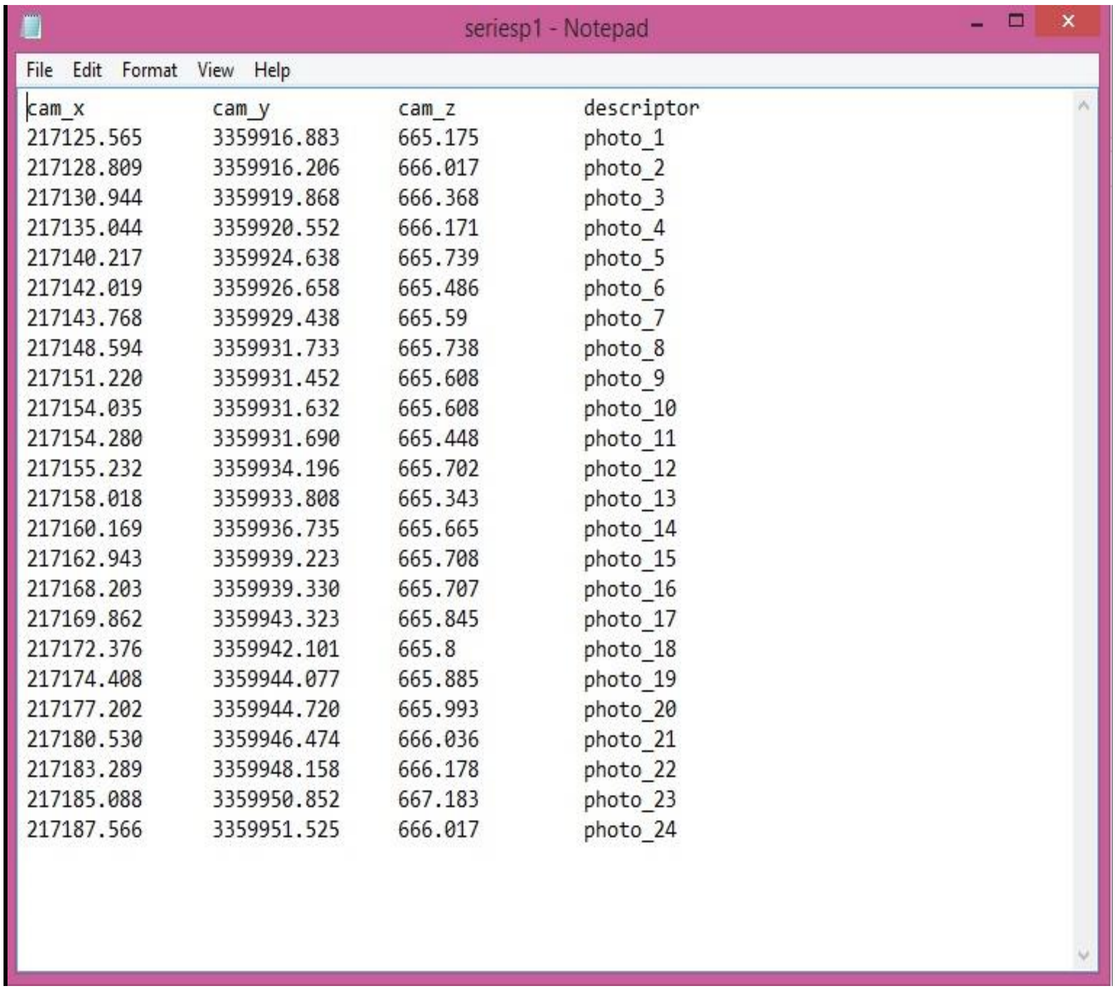


Fig 5.1: Overlapped Images Captured by Nikon D60 SLR

24 images were captured by the camera of the site location. These captured images are 80 percent overlapped and the distance between the two points of capturing the images is about 1 meter. To capture accurate photograph camera is placed on tripod at a distance of 10 meters from the building.

The file containing global position coordinates of the images captured is created. Here global position coordinates are collected manually shown in Fig 5.2.



```
cam_x           cam_y           cam_z           descriptor
217125.565      3359916.883     665.175         photo_1
217128.809      3359916.206     666.017         photo_2
217130.944      3359919.868     666.368         photo_3
217135.044      3359920.552     666.171         photo_4
217140.217      3359924.638     665.739         photo_5
217142.019      3359926.658     665.486         photo_6
217143.768      3359929.438     665.59          photo_7
217148.594      3359931.733     665.738         photo_8
217151.220      3359931.452     665.608         photo_9
217154.035      3359931.632     665.608         photo_10
217154.280      3359931.690     665.448         photo_11
217155.232      3359934.196     665.702         photo_12
217158.018      3359933.808     665.343         photo_13
217160.169      3359936.735     665.665         photo_14
217162.943      3359939.223     665.708         photo_15
217168.203      3359939.330     665.707         photo_16
217169.862      3359943.323     665.845         photo_17
217172.376      3359942.101     665.8           photo_18
217174.408      3359944.077     665.885         photo_19
217177.202      3359944.720     665.993         photo_20
217180.530      3359946.474     666.036         photo_21
217183.289      3359948.158     666.178         photo_22
217185.088      3359950.852     667.183         photo_23
217187.566      3359951.525     666.017         photo_24
```

Fig 5.2 GPS Points Collected for Nikon D60 SLR

5.3 FIELD PLANNING

Field planning is done in the application to find how many numbers of photographs are required of the object and the approximate time required for the photographs. To perform field planning we need some prior information given in Table 5.3. These are:

- Give approximate *Length of Path (meters).*
- Give approximate *Distance from Building (meters).*
- Give approximate *Overlap of photos (percentage).*
- Give approximate *time for one photograph (sec).*

Table 5.3: Input given for field planning

Length of the Path	100 meters
Distance from Building	10 meters
Overlap of photos	70 percent
Time for one photograph	5 minutes or 300 seconds

Following calculation is applied in the software to estimate the number of photographs and time required to capture those photographs.

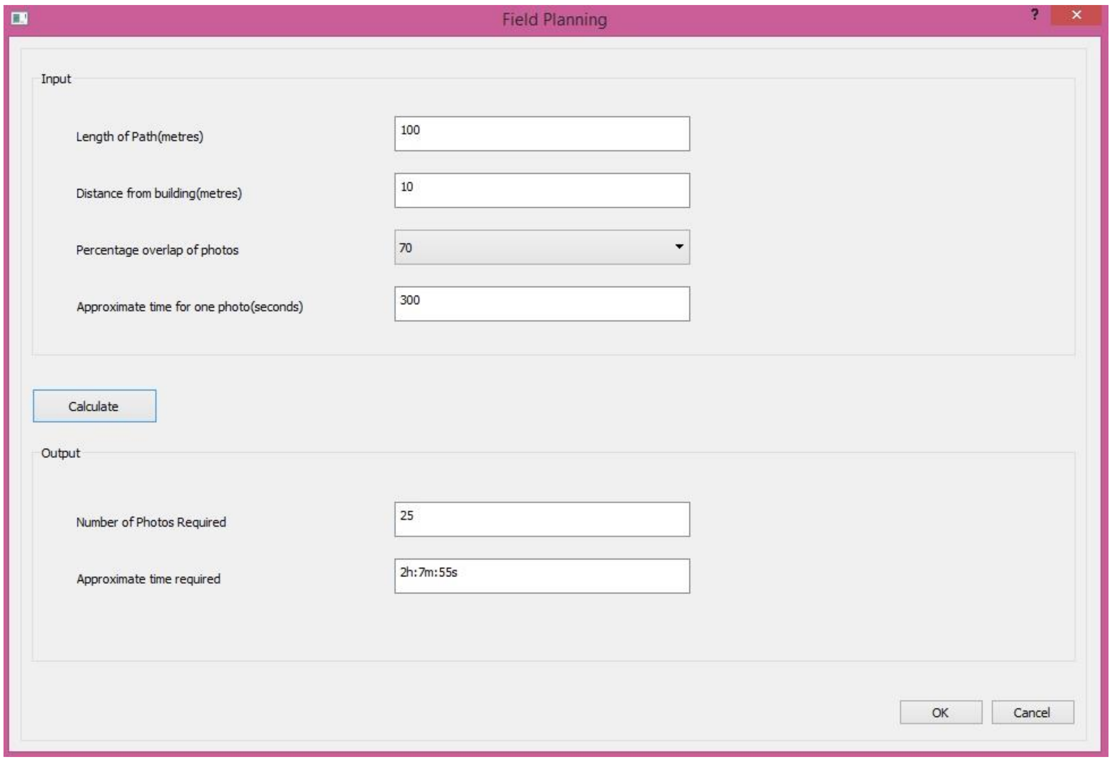
$$n = 1 + \frac{\frac{Length\ of\ the\ Path \times Focal\ Length\ in\ mm}{CCD\ in\ mm\ \times Distance\ From\ Building} - 1}{1 - \frac{Percentage\ of\ Overlap}{100}}$$

Where n is the number of photographs required.

Output of the field planning calculation is shown in Table 5.4 and Fig 5.3 contains the Field Planning window of the application.

Table 5.4: Output of Field Planning

Camera Name	Nikon D60 SLR
Number of Photos Required	25
Approximate Time Required	2 hours 7 minutes 55 seconds


Field Planning

Input

Length of Path(metres): 100

Distance from building(metres): 10

Percentage overlap of photos: 70

Approximate time for one photo(seconds): 300

Output

Number of Photos Required: 25

Approximate time required: 2h:7m:55s

Fig 5.3: Result of Field Planning

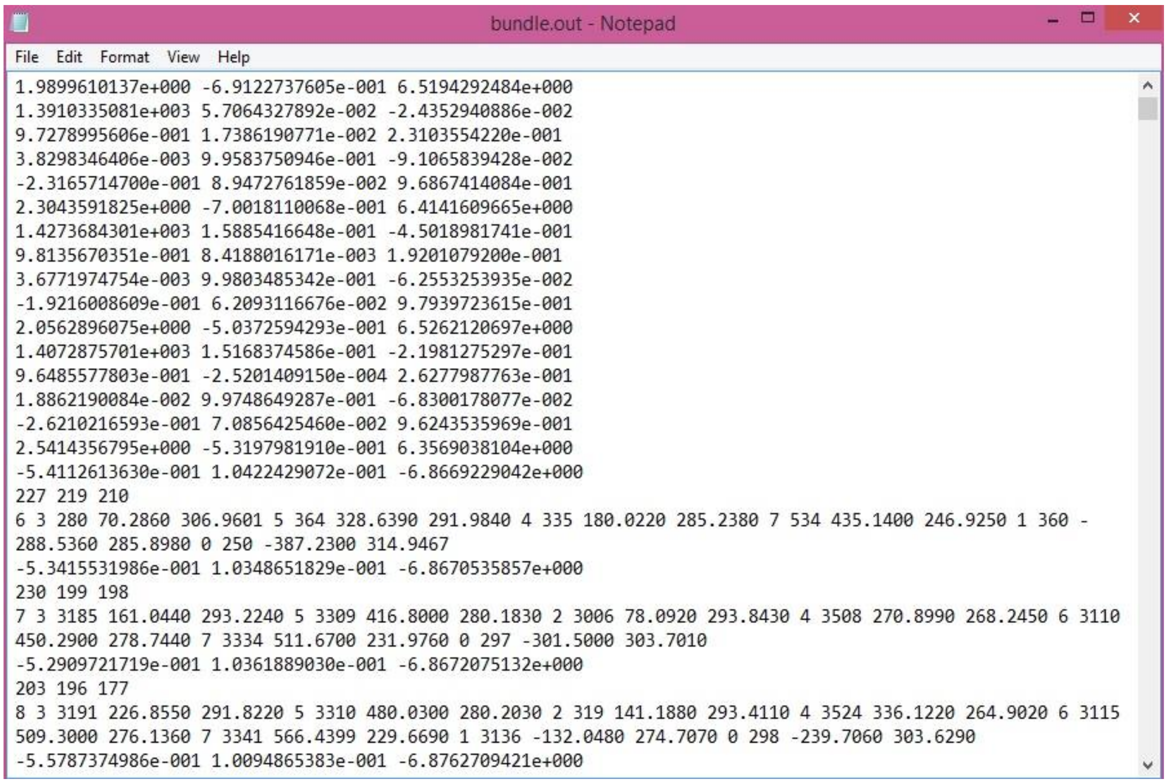
5.4 POINT CLOUD GENERATION

In the Point Cloud Generation step, the system reconstructs the scene incrementally, a few images at a time, using a modified version of the *Sparse Bundle Adjustment* package mentioned above as the underlying optimization engine. Thus, Bundler which is a structure-from-motion (SfM) system for unordered image collections takes a set of images, image features, and image matches as input, and produces a 3D reconstruction of camera and creates a file bundle.out which contains sparse scene geometry as output. This .out file can be visualized in cloud compare software shown in Fig 5.5. The file has the format shown in Fig 5.4 . Initially rows with three columns contains estimated camera intrinsics and extrinsics and has the form:

- In first row first entry is focal length and second, third are the two radial distortion coeffecients.
- In second row we have 3x3 matrix representing the camera rotation.
- In third row we have a 3-vector describing the camera translation.

Further rows has point entry which has the form:

- In first row we have 3-vector describing the 3D position of the point.
- In second row we have 3-vector describing the RGB color of the point.
- In third row we have a view list of the points visible.
    - The view list begins with length of the list. The list is then given as list of quadruplets i.e camera index, index of the SIFT keypoint, x position of that keypoint, y position of that keypoint.



```
1.9899610137e+000 -6.9122737605e-001 6.5194292484e+000
1.3910335081e+003 5.7064327892e-002 -2.4352940886e-002
9.7278995606e-001 1.7386190771e-002 2.3103554220e-001
3.8298346406e-003 9.9583750946e-001 -9.1065839428e-002
-2.3165714700e-001 8.9472761859e-002 9.6867414084e-001
2.3043591825e+000 -7.0018110068e-001 6.4141609665e+000
1.4273684301e+003 1.5885416648e-001 -4.5018981741e-001
9.8135670351e-001 8.4188016171e-003 1.9201079200e-001
3.6771974754e-003 9.9803485342e-001 -6.2553253935e-002
-1.9216008609e-001 6.2093116676e-002 9.7939723615e-001
2.0562896075e+000 -5.0372594293e-001 6.5262120697e+000
1.4072875701e+003 1.5168374586e-001 -2.1981275297e-001
9.6485577803e-001 -2.5201409150e-004 2.6277987763e-001
1.8862190084e-002 9.9748649287e-001 -6.8300178077e-002
-2.6210216593e-001 7.0856425460e-002 9.6243535969e-001
2.5414356795e+000 -5.3197981910e-001 6.3569038104e+000
-5.4112613630e-001 1.0422429072e-001 -6.8669229042e+000
227 219 210
6 3 280 70.2860 306.9601 5 364 328.6390 291.9840 4 335 180.0220 285.2380 7 534 435.1400 246.9250 1 360 -
288.5360 285.8980 0 250 -387.2300 314.9467
-5.3415531986e-001 1.0348651829e-001 -6.8670535857e+000
230 199 198
7 3 3185 161.0440 293.2240 5 3309 416.8000 280.1830 2 3006 78.0920 293.8430 4 3508 270.8990 268.2450 6 3110
450.2900 278.7440 7 3334 511.6700 231.9760 0 297 -301.5000 303.7010
-5.2909721719e-001 1.0361889030e-001 -6.8672075132e+000
203 196 177
8 3 3191 226.8550 291.8220 5 3310 480.0300 280.2030 2 319 141.1880 293.4110 4 3524 336.1220 264.9020 6 3115
509.3000 276.1360 7 3341 566.4399 229.6690 1 3136 -132.0480 274.7070 0 298 -239.7060 303.6290
-5.5787374986e-001 1.0094865383e-001 -6.8762709421e+000
```

Fig 5.4: Bundle.out file generated by Bundler

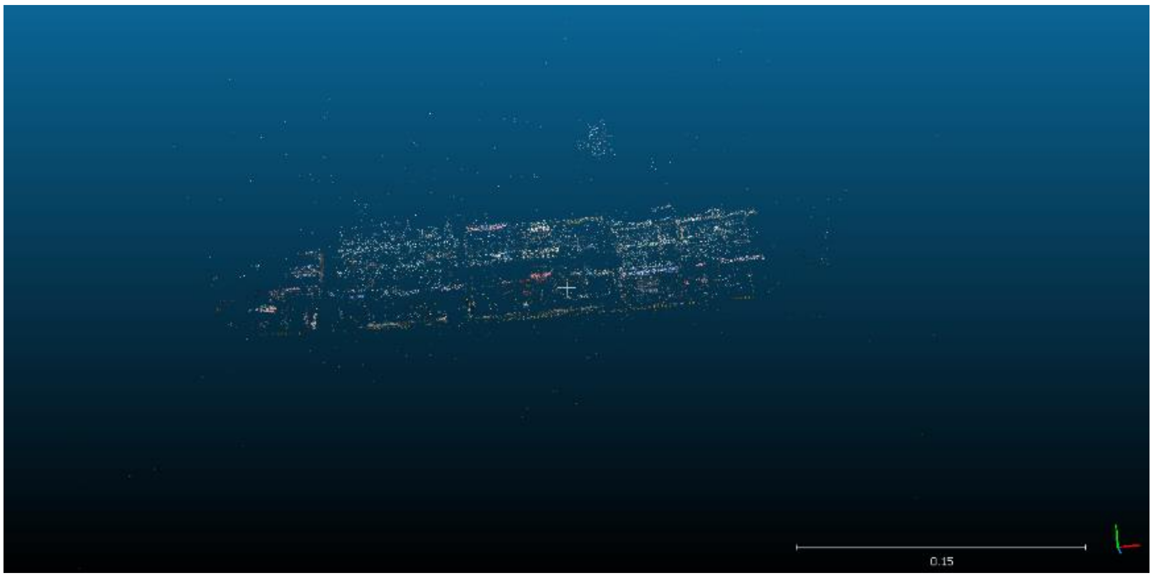
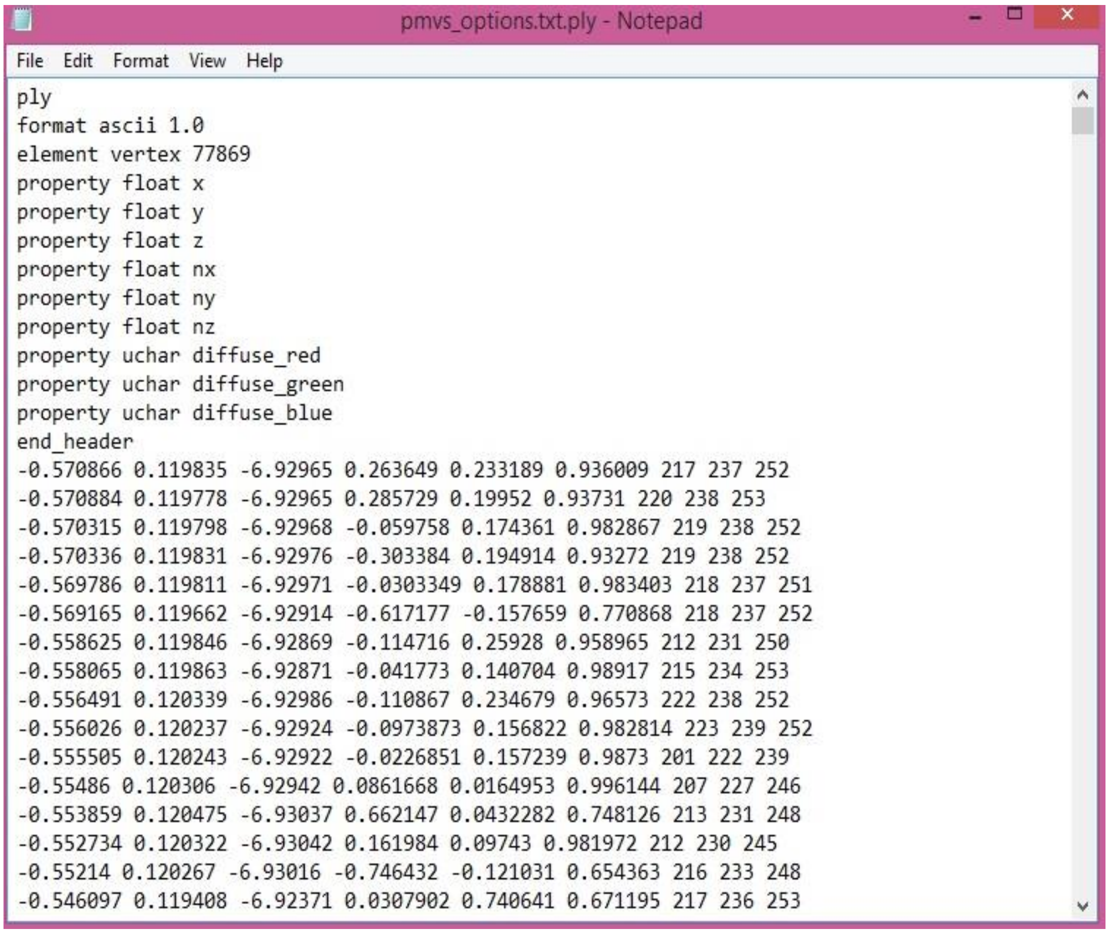


Fig 5.5: Sparse Point Cloud generated by Bundler

After running Bundler next software CMVS, PMVS are used to generate model file ‘*pmvs_options.txt_transformed.ply’* as output which contains dense point cloud this file has the format shown in Fig 5.6 .

- The first three columns shows the 3D location.

- Fourth, fifth and sixth contains estimated surface normals for all the reconstructed points.
- Last three columnns have 3d colored points i.e, RGB value of the location.



```
ply
format ascii 1.0
element vertex 77869
property float x
property float y
property float z
property float nx
property float ny
property float nz
property uchar diffuse_red
property uchar diffuse_green
property uchar diffuse_blue
end_header
-0.570866 0.119835 -6.92965 0.263649 0.233189 0.936009 217 237 252
-0.570884 0.119778 -6.92965 0.285729 0.19952 0.93731 220 238 253
-0.570315 0.119798 -6.92968 -0.059758 0.174361 0.982867 219 238 252
-0.570336 0.119831 -6.92976 -0.303384 0.194914 0.93272 219 238 252
-0.569786 0.119811 -6.92971 -0.0303349 0.178881 0.983403 218 237 251
-0.569165 0.119662 -6.92914 -0.617177 -0.157659 0.770868 218 237 252
-0.558625 0.119846 -6.92869 -0.114716 0.25928 0.958965 212 231 250
-0.558065 0.119863 -6.92871 -0.041773 0.140704 0.98917 215 234 253
-0.556491 0.120339 -6.92986 -0.110867 0.234679 0.96573 222 238 252
-0.556026 0.120237 -6.92924 -0.0973873 0.156822 0.982814 223 239 252
-0.555505 0.120243 -6.92922 -0.0226851 0.157239 0.9873 201 222 239
-0.55486 0.120306 -6.92942 0.0861668 0.0164953 0.996144 207 227 246
-0.553859 0.120475 -6.93037 0.662147 0.0432282 0.748126 213 231 248
-0.552734 0.120322 -6.93042 0.161984 0.09743 0.981972 212 230 245
-0.55214 0.120267 -6.93016 -0.746432 -0.121031 0.654363 216 233 248
-0.546097 0.119408 -6.92371 0.0307902 0.740641 0.671195 217 236 253
```

Fig 5.6 : Model file containing Dense Point Cloud

These dense point cloud can be viewed in cloud compare software shown in Fig 5.7 .

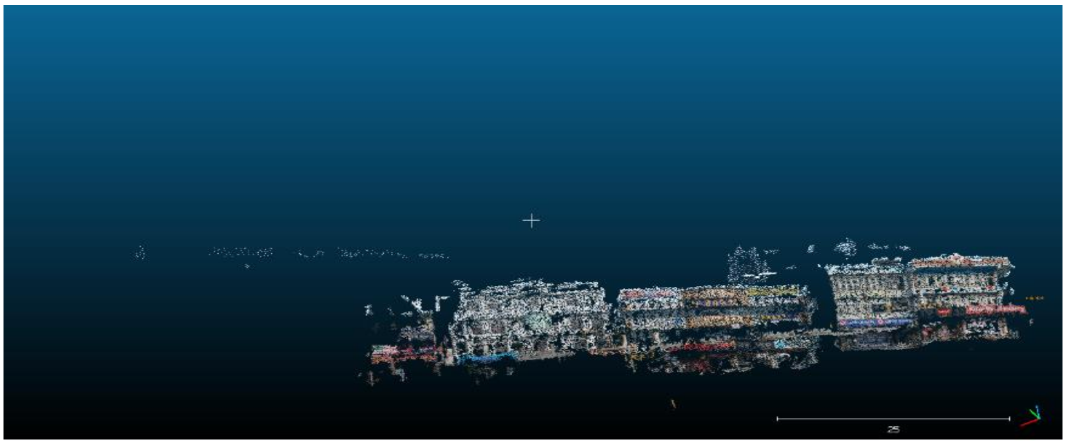


Fig 5.7 : Dense Point Cloud

Different point cloud generated from different devices is shown in Fig 5.8 and Fig 5.9.

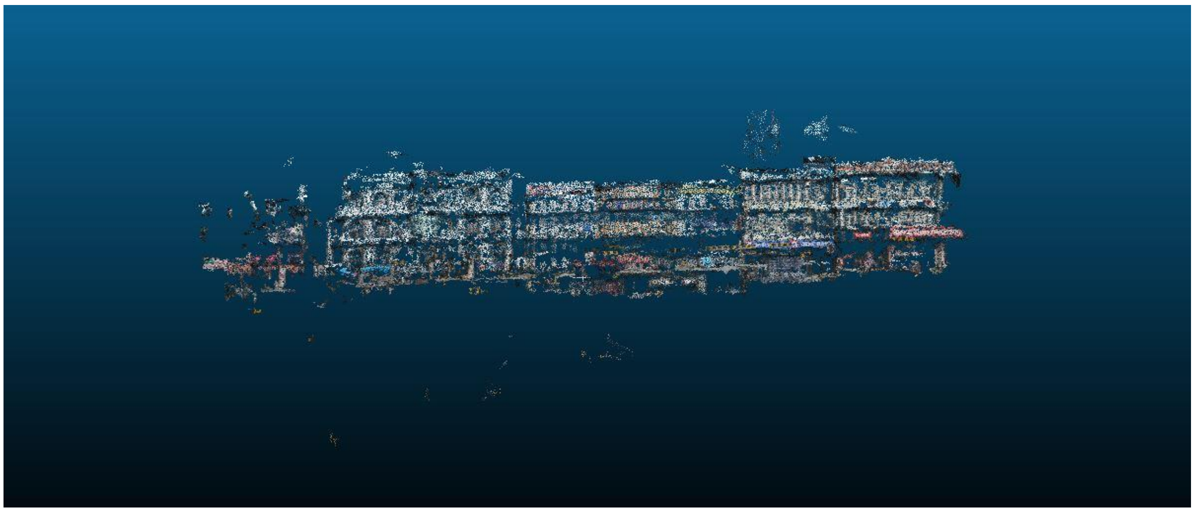


Fig 5.8: Point cloud generated by Nikon D60

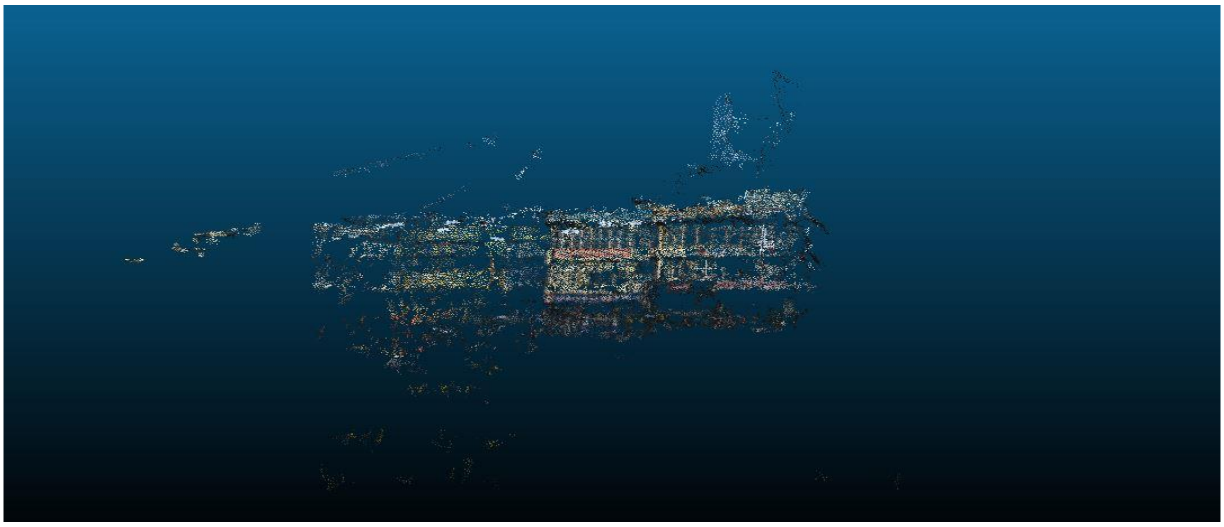
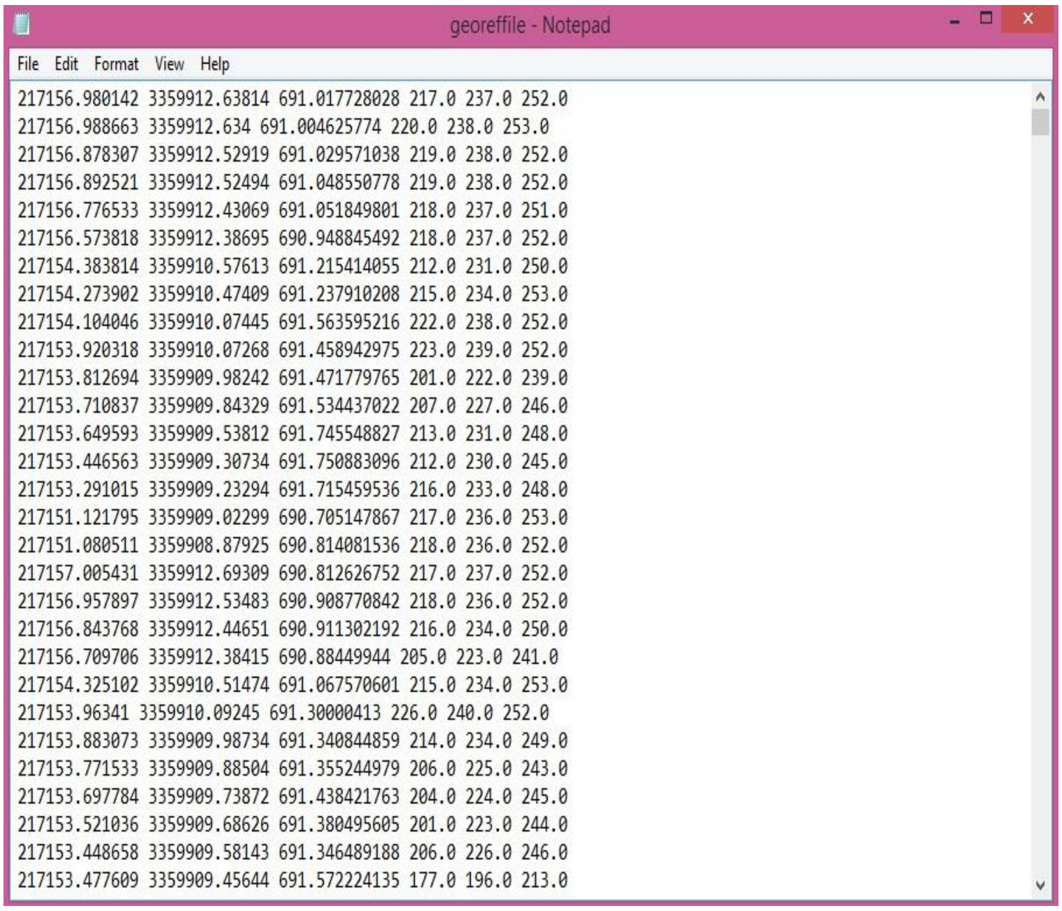


Fig 5.9: Point cloud generated by Samsung Galaxy core

5.5 GEOREFERNCING

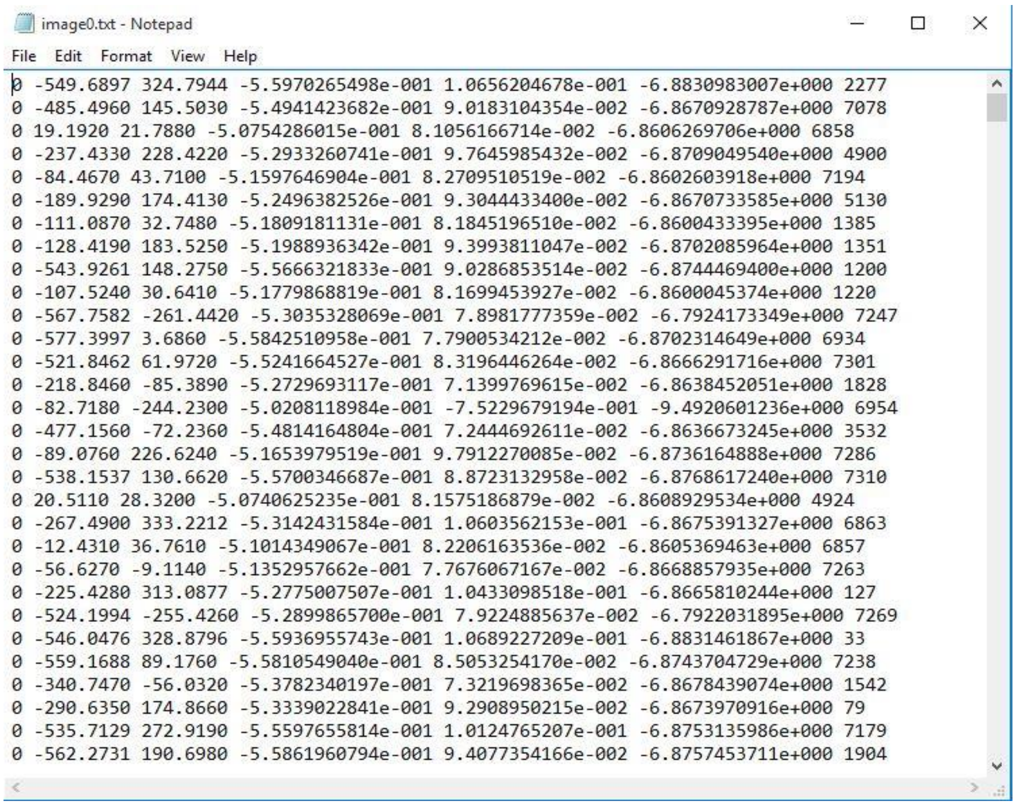
Through this application georeferencing of the data can be performed. Georeferencing finds out the relation between the Local Coordinate System and the Global Coordinate system. It creates a file ‘georeffile.txt’. The file has the format shown in Fig 5.10 which contains location x,y,z in real world coordinate with rgb values.



```
217156.980142 3359912.63814 691.017728028 217.0 237.0 252.0
217156.988663 3359912.634 691.004625774 220.0 238.0 253.0
217156.878307 3359912.52919 691.029571038 219.0 238.0 252.0
217156.892521 3359912.52494 691.048550778 219.0 238.0 252.0
217156.776533 3359912.43069 691.051849801 218.0 237.0 251.0
217156.573818 3359912.38695 690.948845492 218.0 237.0 252.0
217154.383814 3359910.57613 691.215414055 212.0 231.0 250.0
217154.273902 3359910.47409 691.237910208 215.0 234.0 253.0
217154.104046 3359910.07445 691.563595216 222.0 238.0 252.0
217153.920318 3359910.07268 691.458942975 223.0 239.0 252.0
217153.812694 3359909.98242 691.471779765 201.0 222.0 239.0
217153.710837 3359909.84329 691.534437022 207.0 227.0 246.0
217153.649593 3359909.53812 691.745548827 213.0 231.0 248.0
217153.446563 3359909.30734 691.750883096 212.0 230.0 245.0
217153.291015 3359909.23294 691.715459536 216.0 233.0 248.0
217151.121795 3359909.02299 690.705147867 217.0 236.0 253.0
217151.080511 3359908.87925 690.814081536 218.0 236.0 252.0
217157.005431 3359912.69309 690.812626752 217.0 237.0 252.0
217156.957897 3359912.53483 690.908770842 218.0 236.0 252.0
217156.843768 3359912.44651 690.911302192 216.0 234.0 250.0
217156.709706 3359912.38415 690.88449944 205.0 223.0 241.0
217154.325102 3359910.51474 691.067570601 215.0 234.0 253.0
217153.96341 3359910.09245 691.30000413 226.0 240.0 252.0
217153.883073 3359909.98734 691.340844859 214.0 234.0 249.0
217153.771533 3359909.88504 691.355244979 206.0 225.0 243.0
217153.697784 3359909.73872 691.438421763 204.0 224.0 245.0
217153.521036 3359909.68626 691.380495605 201.0 223.0 244.0
217153.448658 3359909.58143 691.346489188 206.0 226.0 246.0
217153.477609 3359909.45644 691.572224135 177.0 196.0 213.0
```

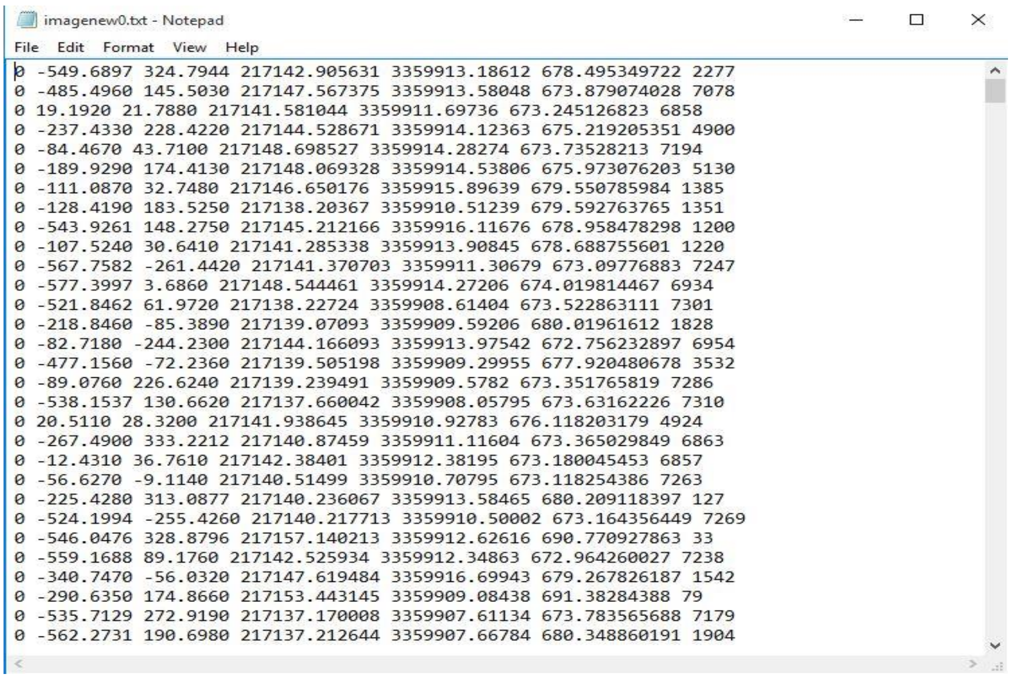
Fig 5.10: Georeffile Format

In this step we also perform Georeference of each image by executing a python code.



```
0 -549.6897 324.7944 -5.5970265498e-001 1.0656204678e-001 -6.8830983007e+000 2277
0 -485.4960 145.5030 -5.4941423682e-001 9.0183104354e-002 -6.8670928787e+000 7078
0 19.1920 21.7880 -5.0754286015e-001 8.1056166714e-002 -6.8606269706e+000 6858
0 -237.4330 228.4220 -5.2933260741e-001 9.7645985432e-002 -6.8709049540e+000 4900
0 -84.4670 43.7100 -5.1597646904e-001 8.2709510519e-002 -6.8602603918e+000 7194
0 -189.9290 174.4130 -5.2496382526e-001 9.3044433400e-002 -6.8670733585e+000 5130
0 -111.0870 32.7480 -5.1809181131e-001 8.1845196510e-002 -6.8600433395e+000 1385
0 -128.4190 183.5250 -5.1988936342e-001 9.3993811047e-002 -6.8702085964e+000 1351
0 -543.9261 148.2750 -5.5666321833e-001 9.0286853514e-002 -6.8744469400e+000 1200
0 -107.5240 30.6410 -5.1779868819e-001 8.1699453927e-002 -6.8600045374e+000 1220
0 -567.7582 -261.4420 -5.3035328069e-001 7.8981777359e-002 -6.7924173349e+000 7247
0 -577.3997 3.6860 -5.5842510958e-001 7.7900534212e-002 -6.8702314649e+000 6934
0 -521.8462 61.9720 -5.5241664527e-001 8.3196446264e-002 -6.8666291716e+000 7301
0 -218.8460 -85.3890 -5.2729693117e-001 7.1399769615e-002 -6.8638452051e+000 1828
0 -82.7180 -244.2300 -5.0208118984e-001 -7.5229679194e-001 -9.4920601236e+000 6954
0 -477.1560 -72.2360 -5.4814164804e-001 7.2444692611e-002 -6.8636673245e+000 3532
0 -89.0760 226.6240 -5.1653979519e-001 9.7912270085e-002 -6.8736164888e+000 7286
0 -538.1537 130.6620 -5.5700346687e-001 8.8723132958e-002 -6.8768617240e+000 7310
0 20.5110 28.3200 -5.0740625235e-001 8.1575186879e-002 -6.8608929534e+000 4924
0 -267.4900 333.2212 -5.3142431584e-001 1.0603562153e-001 -6.8675391327e+000 6863
0 -12.4310 36.7610 -5.1014349067e-001 8.2206163536e-002 -6.8605369463e+000 6857
0 -56.6270 -9.1140 -5.1352957662e-001 7.7676067167e-002 -6.8668857935e+000 7263
0 -225.4280 313.0877 -5.2775007507e-001 1.0433098518e-001 -6.8665810244e+000 127
0 -524.1994 -255.4260 -5.2899865700e-001 7.9224885637e-002 -6.7922031895e+000 7269
0 -546.0476 328.8796 -5.5936955743e-001 1.0689227209e-001 -6.8831461867e+000 33
0 -559.1688 89.1760 -5.5810549040e-001 8.5053254170e-002 -6.8743704729e+000 7238
0 -340.7470 -56.0320 -5.3782340197e-001 7.3219698365e-002 -6.8678439074e+000 1542
0 -290.6350 174.8660 -5.3339022841e-001 9.2908950215e-002 -6.8673970916e+000 79
0 -535.7129 272.9190 -5.5597655814e-001 1.0124765207e-001 -6.8753135986e+000 7179
0 -562.2731 190.6980 -5.5861960794e-001 9.4077354166e-002 -6.8757453711e+000 1904
```

Fig 5.11: Point Cloud of Images before Georeferencing



```
0 -549.6897 324.7944 217142.905631 3359913.18612 678.495349722 2277
0 -485.4960 145.5030 217147.567375 3359913.58048 673.879074028 7078
0 19.1920 21.7880 217141.581044 3359911.69736 673.245126823 6858
0 -237.4330 228.4220 217144.528671 3359914.12363 675.219205351 4900
0 -84.4670 43.7100 217148.698527 3359914.28274 673.73528213 7194
0 -189.9290 174.4130 217148.069328 3359914.53806 675.973076203 5130
0 -111.0870 32.7480 217146.650176 3359915.89639 679.550785984 1385
0 -128.4190 183.5250 217138.20367 3359910.51239 679.592763765 1351
0 -543.9261 148.2750 217145.212166 3359916.11676 678.958478298 1200
0 -107.5240 30.6410 217141.285338 3359913.90845 678.688755601 1220
0 -567.7582 -261.4420 217141.370703 3359911.30679 673.09776883 7247
0 -577.3997 3.6860 217148.544461 3359914.27206 674.019814467 6934
0 -521.8462 61.9720 217138.22724 3359908.61404 673.522863111 7301
0 -218.8460 -85.3890 217139.07093 3359909.59206 680.01961612 1828
0 -82.7180 -244.2300 217144.166093 3359913.97542 672.756232897 6954
0 -477.1560 -72.2360 217139.505198 3359909.29955 677.920480678 3532
0 -89.0760 226.6240 217139.239491 3359909.5782 673.351765819 7286
0 -538.1537 130.6620 217137.660042 3359908.05795 673.63162226 7310
0 20.5110 28.3200 217141.938645 3359910.92783 676.118203179 4924
0 -267.4900 333.2212 217140.87459 3359911.11604 673.365029849 6863
0 -12.4310 36.7610 217142.38401 3359912.38195 673.180045453 6857
0 -56.6270 -9.1140 217140.51499 3359910.70795 673.118254386 7263
0 -225.4280 313.0877 217140.236067 3359913.58465 680.209118397 127
0 -524.1994 -255.4260 217140.217713 3359910.50002 673.164356449 7269
0 -546.0476 328.8796 217157.140213 3359912.62616 690.770927863 33
0 -559.1688 89.1760 217142.525934 3359912.34863 672.964260027 7238
0 -340.7470 -56.0320 217147.619484 3359916.69943 679.267826187 1542
0 -290.6350 174.8660 217153.443145 3359909.08438 691.38284388 79
0 -535.7129 272.9190 217137.170008 3359907.61134 673.783565688 7179
0 -562.2731 190.6980 217137.212644 3359907.66784 680.348860191 1904
```

Fig 5.12: Point Cloud of Images after Georeferencing

Here in Fig 5.11 we have point cloud of image before georeference. It contains pixel position with corresponding local coordinates and after performing georeferencing we have coordinates with real world coordiantes shown in Fig 5.12.

5.6 BUILDING TEXTURE EXTRACTION

For generation of 3D model we need texture of the building. Through this step in application we can get texture of the building and while extracting the texture of the building auto height extraction of the building is performed.

Image Segmentation is performed for texture extraction which creates a database file with the segment name containing the attribute values and a cropped segment of image in jpeg format. While creating segment of the image the height is extracted automatically of each segment.This is performed by executing a code in python which maps the pixels to real world coordiante and finding the difference between the altitude.

5.6.1 EVALUATION OF HEIGHT

The software generates 3D model of the building of approximately same height as real height. For checking the accuracy in height, images from different cameras i.e Nikon D60 (10.2 mp), Samsung Galaxy Core(5 mp) are captured and 3D is generated from the software. The original height of each building is known by manually recording it via Laser Distance Meter. The height is compared of each device. Table 6.5 shows the original height of building 1,2,3 and its averages and differences.

Table 5.5 : Height Evaluation

BUILDING	**ORIGINAL HEIGHT(in meters)**	**Nikon D60 (10.2 mp)**	**Samsung Galaxy Core(5 mp)**
BUILDING 1	9.898	9.02	8.04
BUILDING 2	10.220	8.24	9.74
BUILDING 3	12.126	10.80	8.45
AVERAGES	10.745	9.3533	8.7433
DIFFERENCES		**1.39**	**2.0019**

5.7 3D MODEL GENERATION

Last step is creation of 3dimensional model of building. Segments created of building can be visualized over which textured image can be visible. These model are created through pycollada and can be visualized on geo-portal i.e, google earth. These models can be seen on the footprint provided in the form of kml file.

Here we have 3dimensional model generated from two cameras i.e, Nikon D60 and Samsung Galaxy core. The corresponding 3D Models are shown below in Fig 5.13 and Fig 5.14:

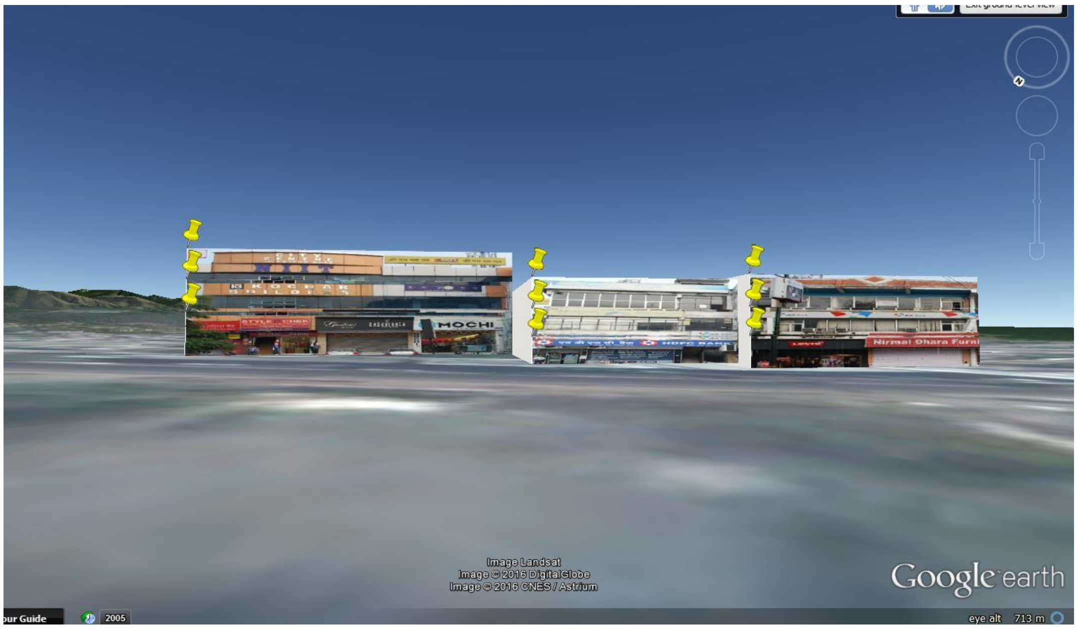


Fig 5.13: 3D view of point cloud near Madhuban, Rajpur Road using Nikon D60

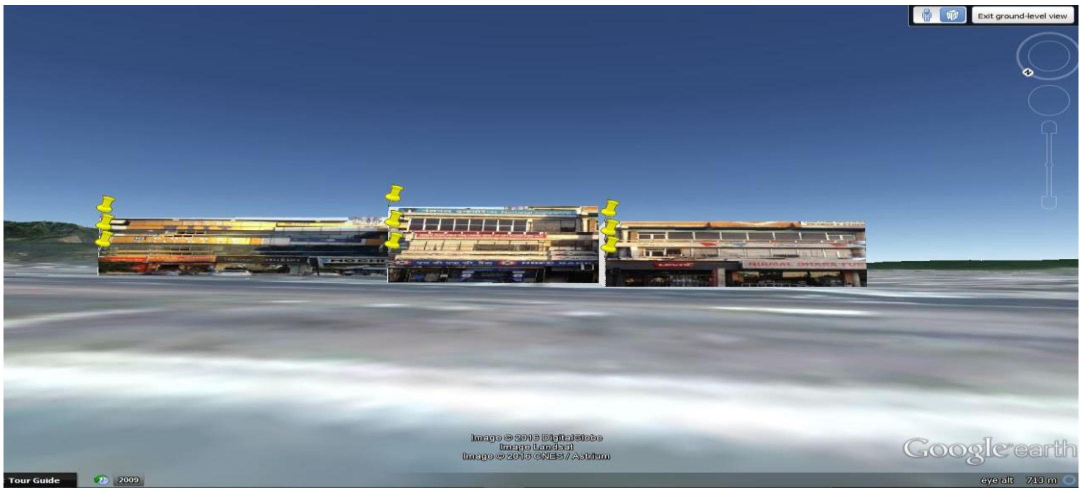
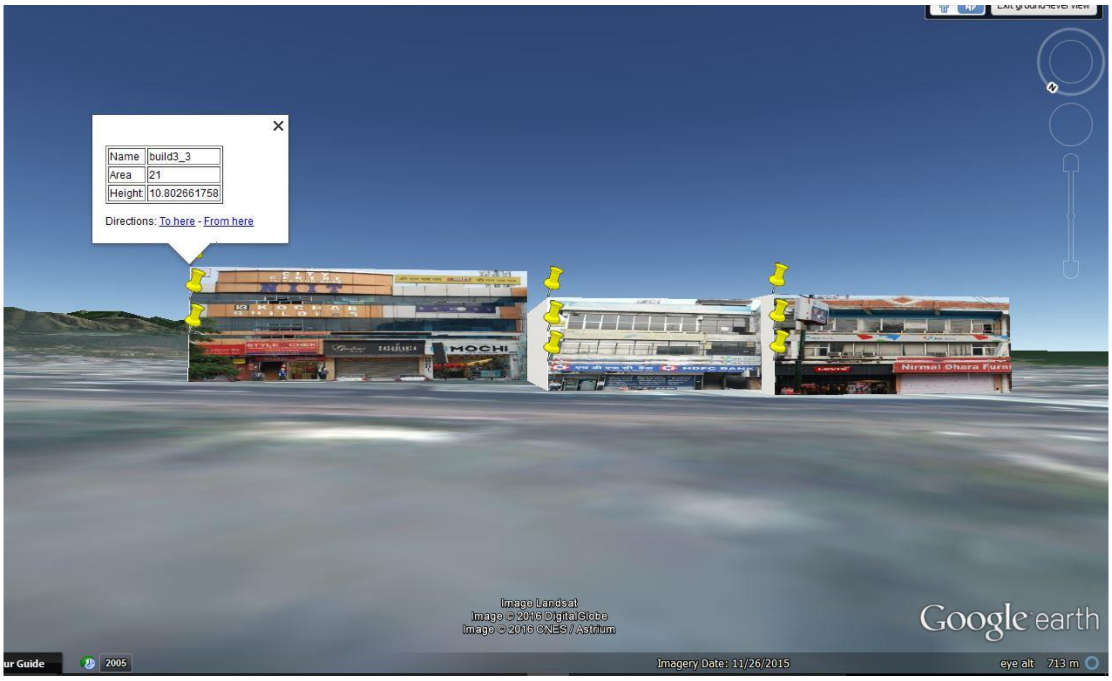


Fig 5.14: 3D view of point cloud near Madhuban, Rajpur Road using Samsung Galaxy Core

The placemark are the yellow pins attached with each segment containing specific geo–information stored uniquely in corresponding files. On clicking any of the pin shows the information about that segment. Fig 5.15 shows the placemark containing information.



Name	build3_3
Area	21
Height:	10.802661758

Directions: To here - From here

Fig 5.15: Placemark showing the geo-spatial information

5.8 QUERY RESULTS

In last step user can ask a query. This can be done either by segment name or manually entered attributes. The result of the query is shown in Table 5.6.

Table 5.6 : Query Result

Name (string)	build3
Latitude (string)	78.05830296424719
Longitude (string)	30.3380113844064
Altitude (string)	4.650452
Area	21.0

Images of the Field work carried out and the final 3D view of buildings on Eastern Canal Road of Dehradun, Uttarakhand

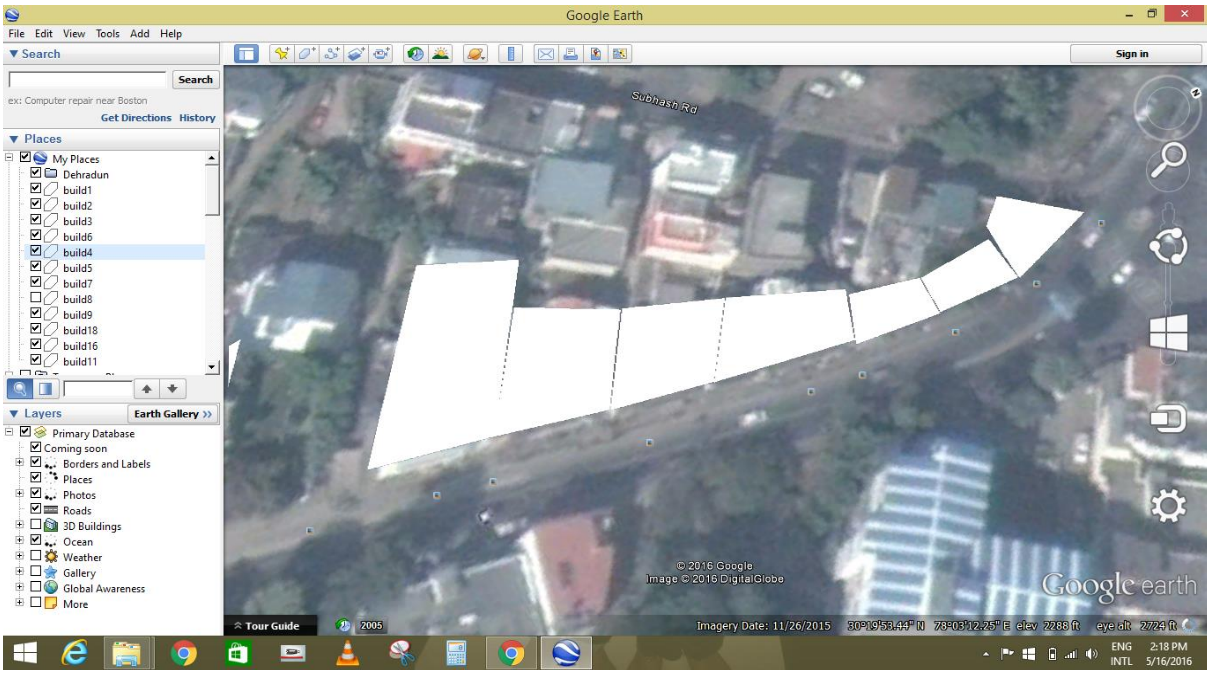


Fig 5.16: KML of buildings of study area

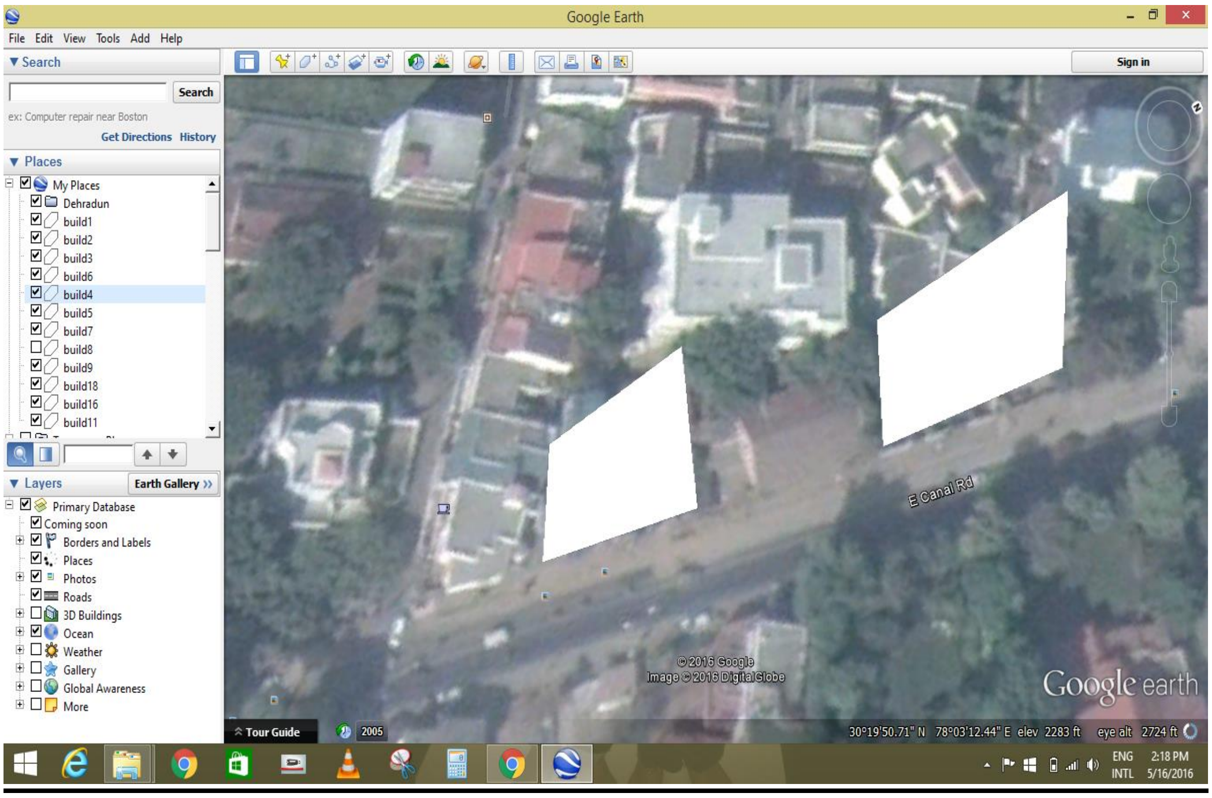


Fig 5.17: KML of Buildings Of study area

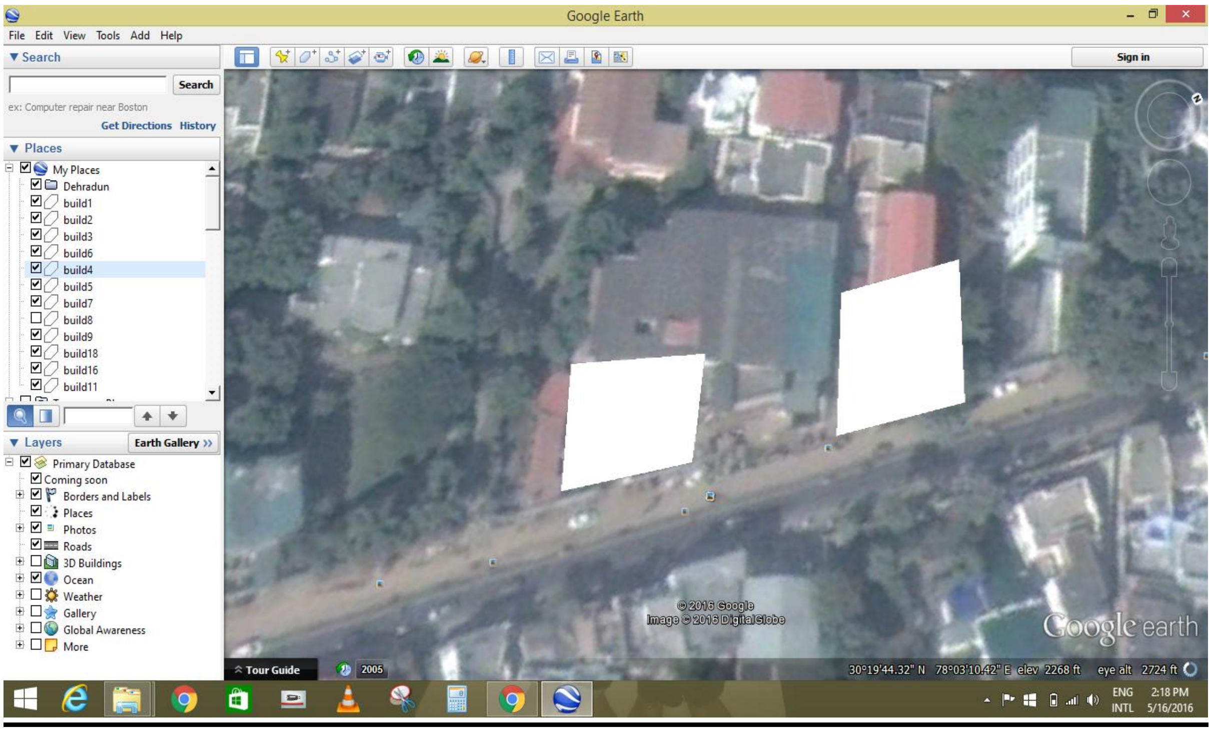


Fig 5.18: KML Of the buildings

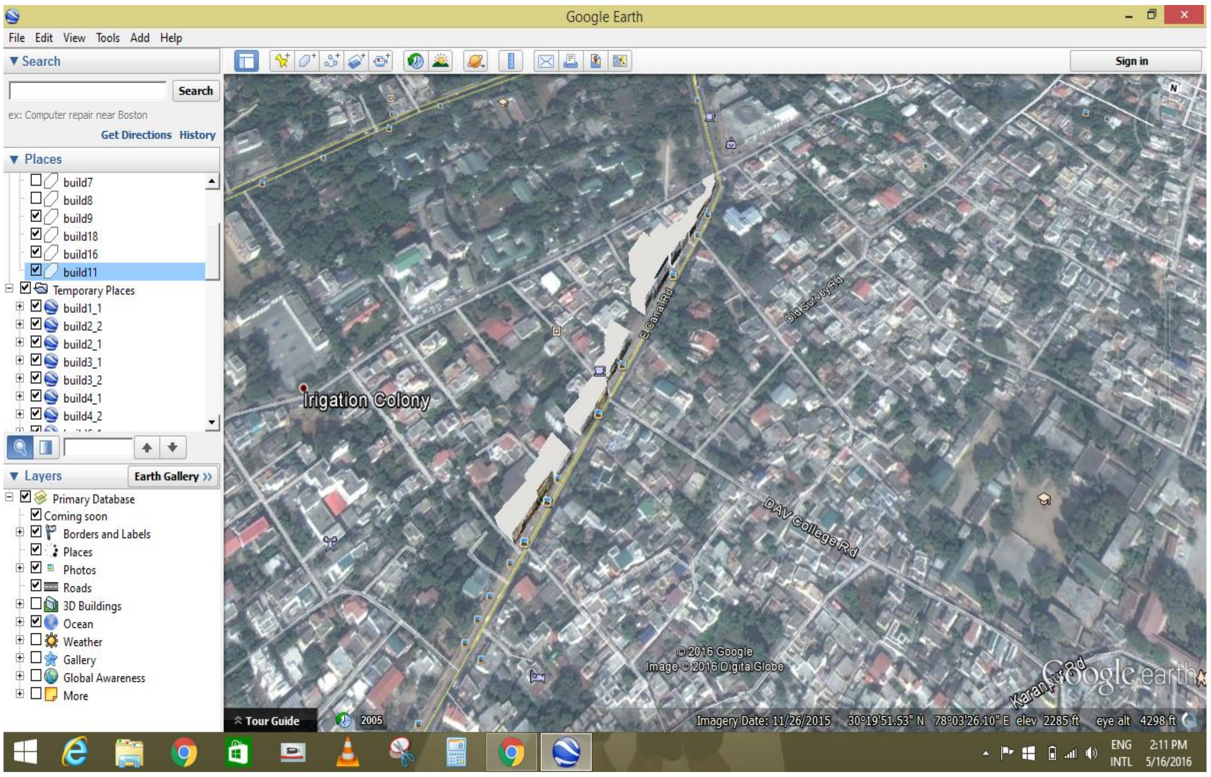


Fig 5.19: 3D View of the buildings in Google earth from Distance

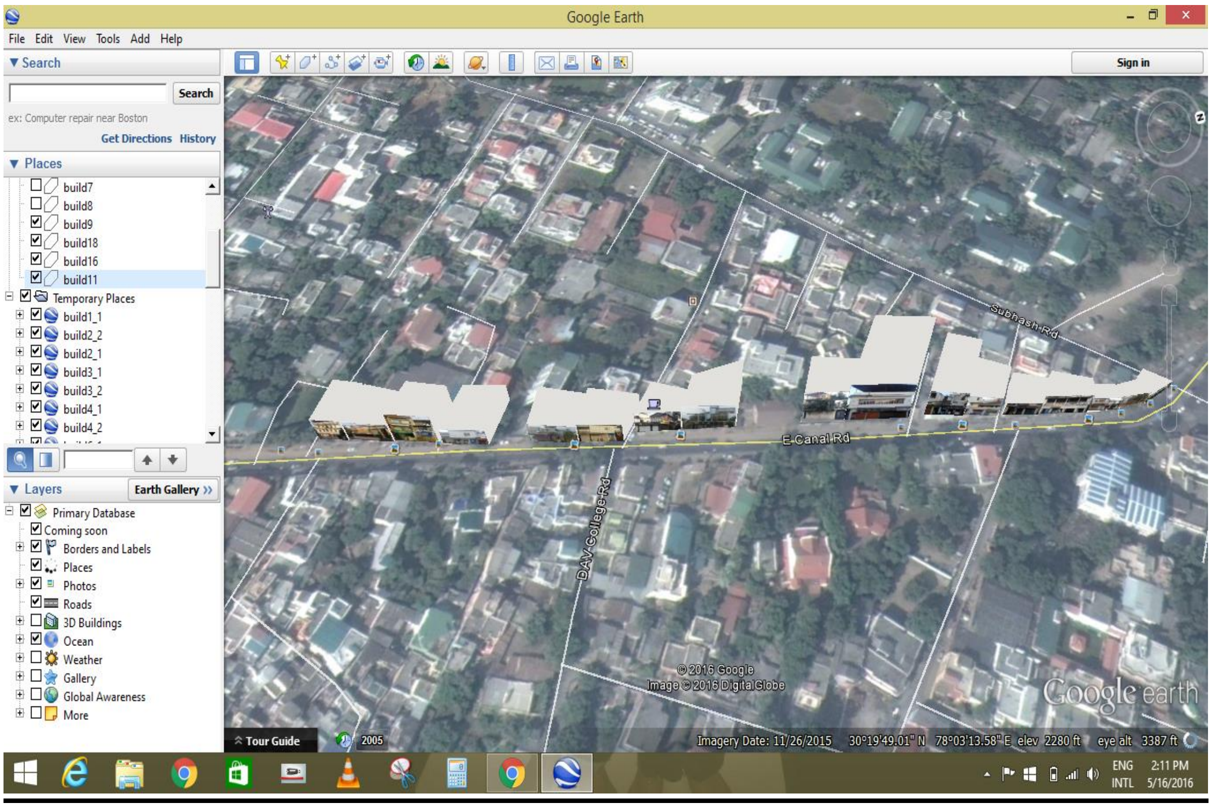


Fig 5.20: 3D view Of the buildings on Google earth

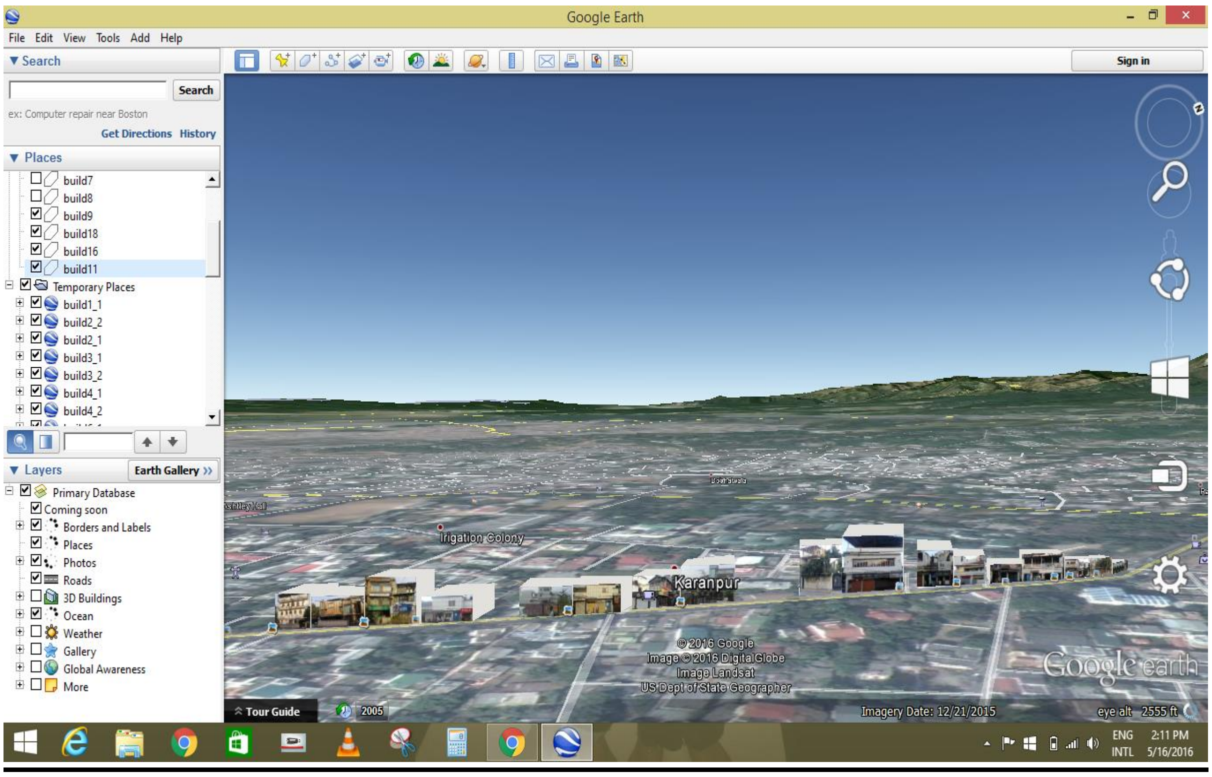


Fig 5.21: 3D view Of the buildings on Google earth

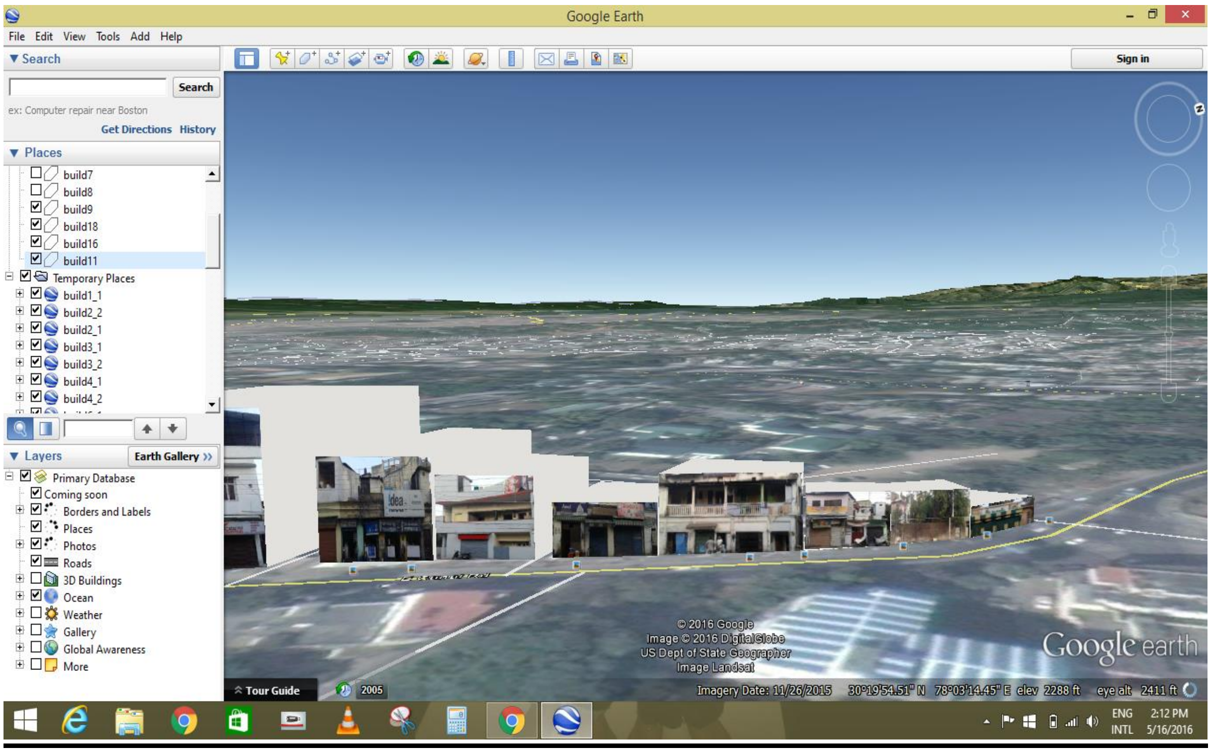


Fig 5.22: 3D view Of the buildings on Google earth

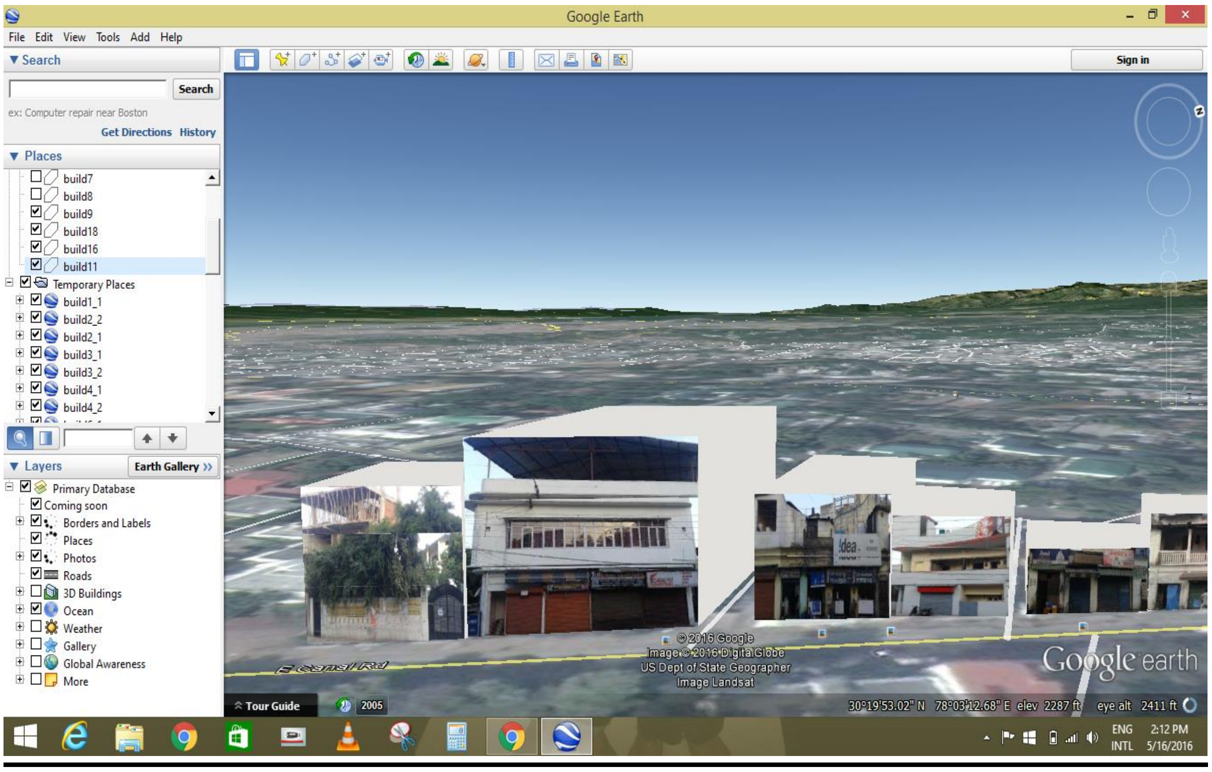


Fig 5.23: 3D view Of the buildings on Google earth

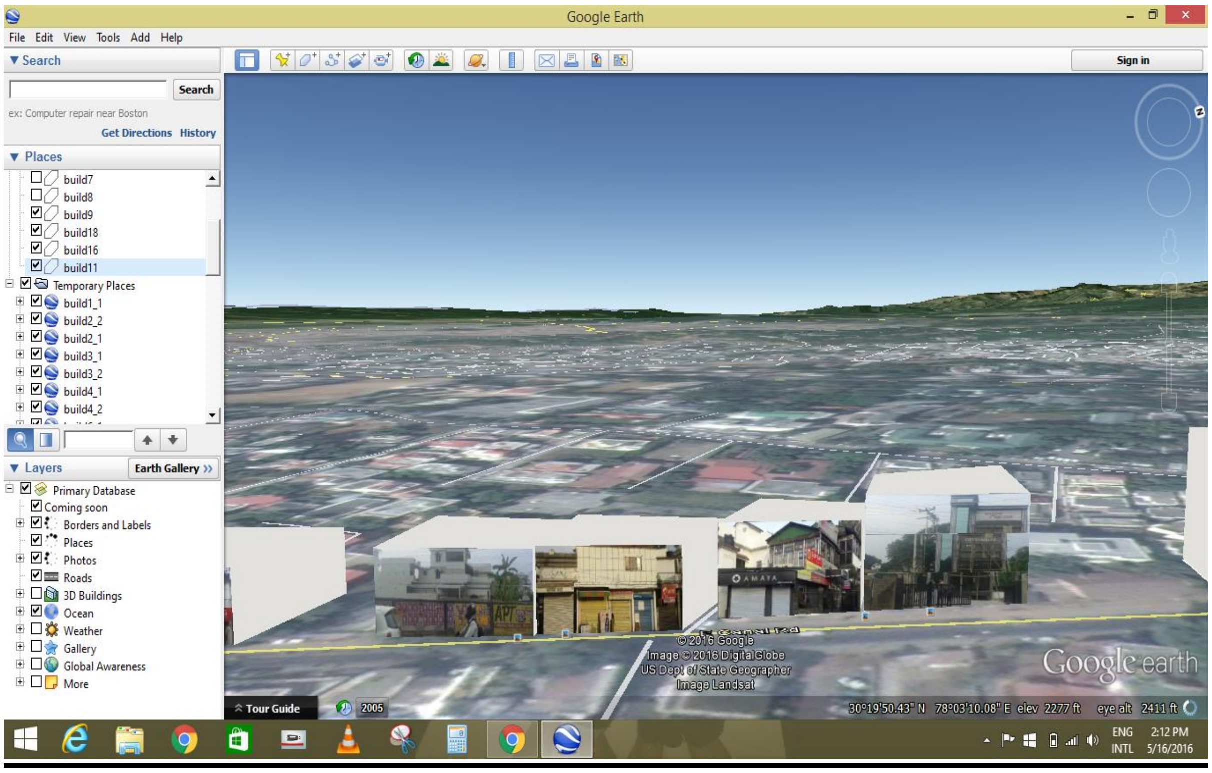


Fig 5.24: 3D view Of the buildings on Google earth

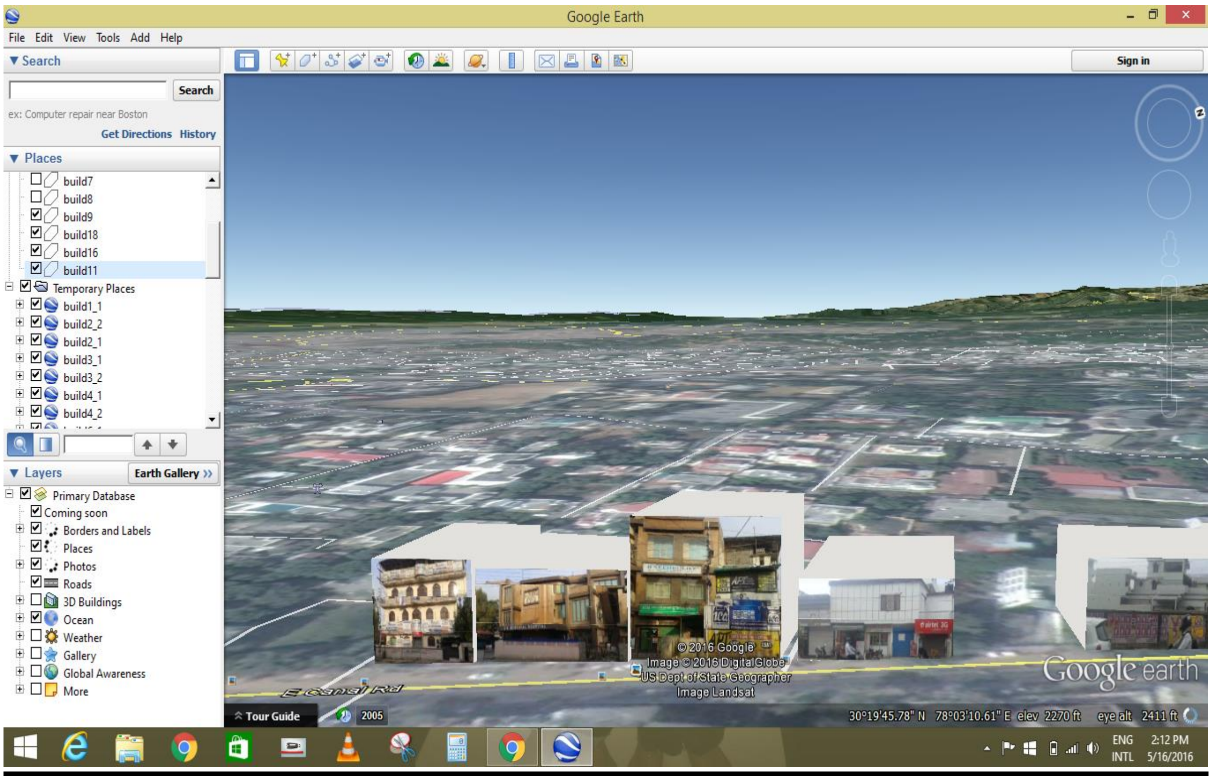


Fig 5.25: 3D view Of the buildings on Google earth

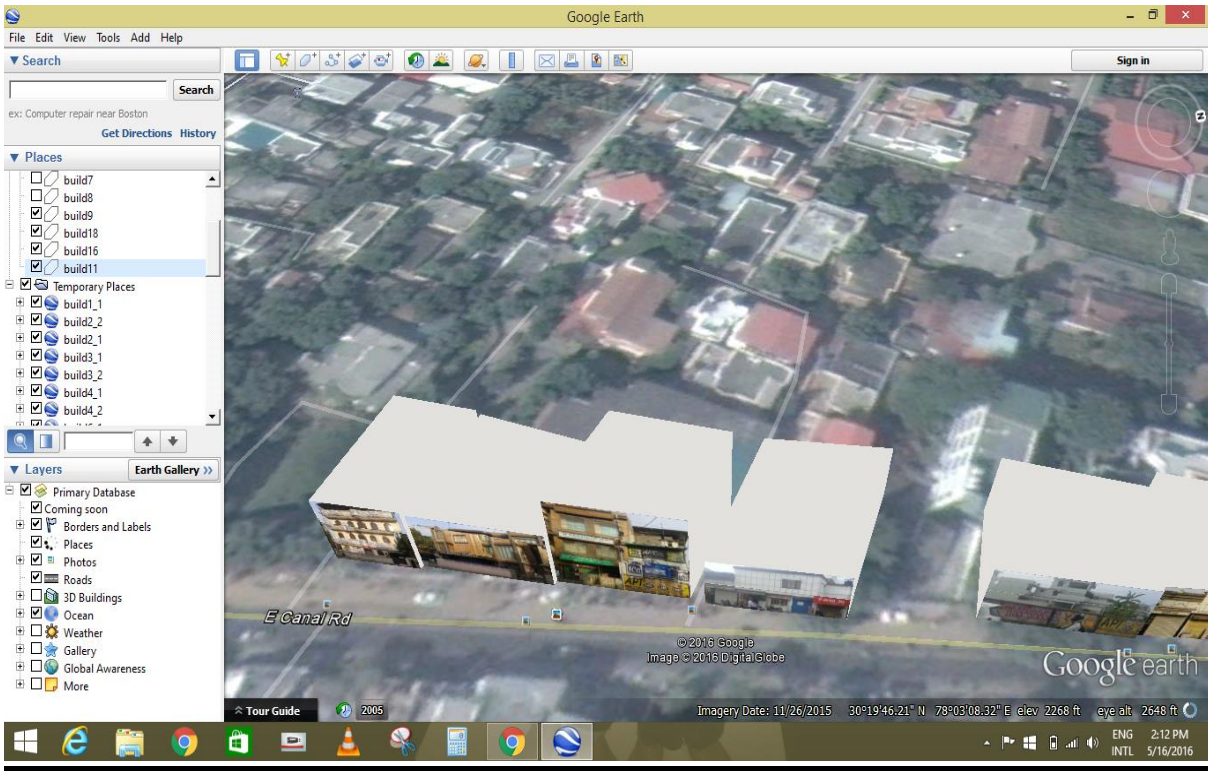


Fig 5.26: 3D view Of the buildings on Google earth

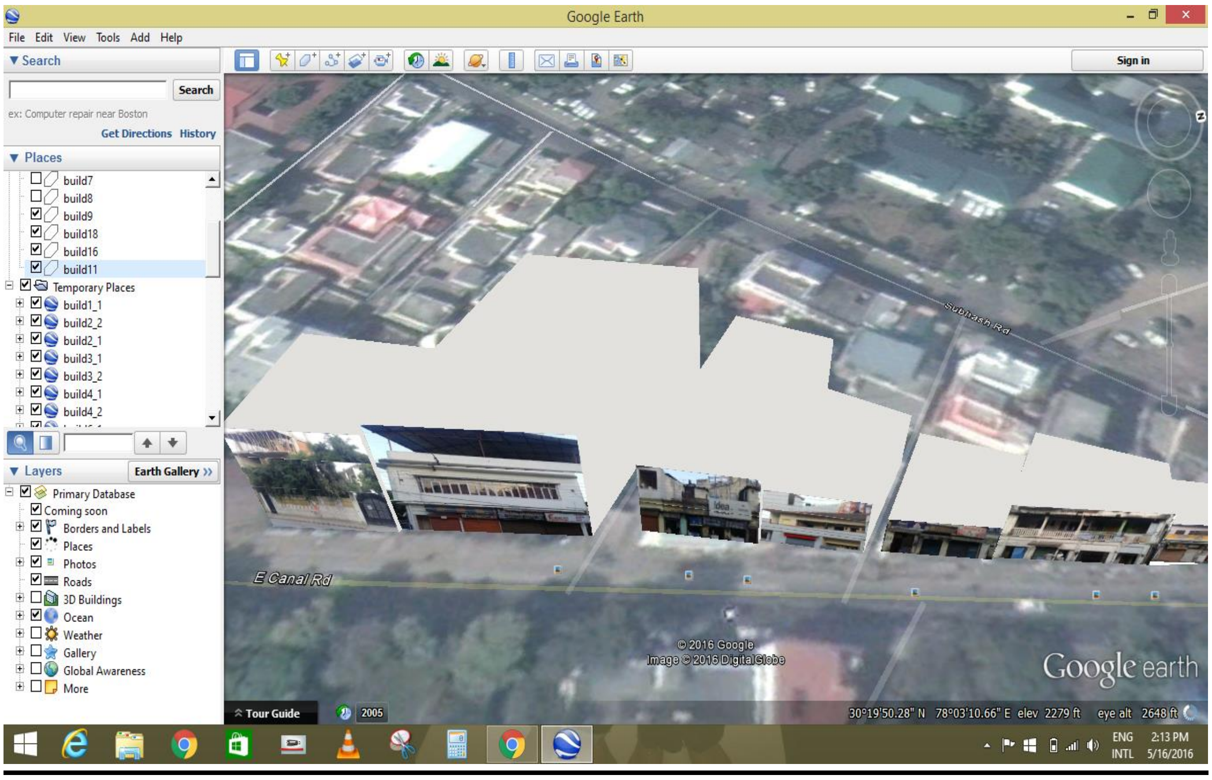


Fig 5.27: 3D view Of the buildings on Google earth

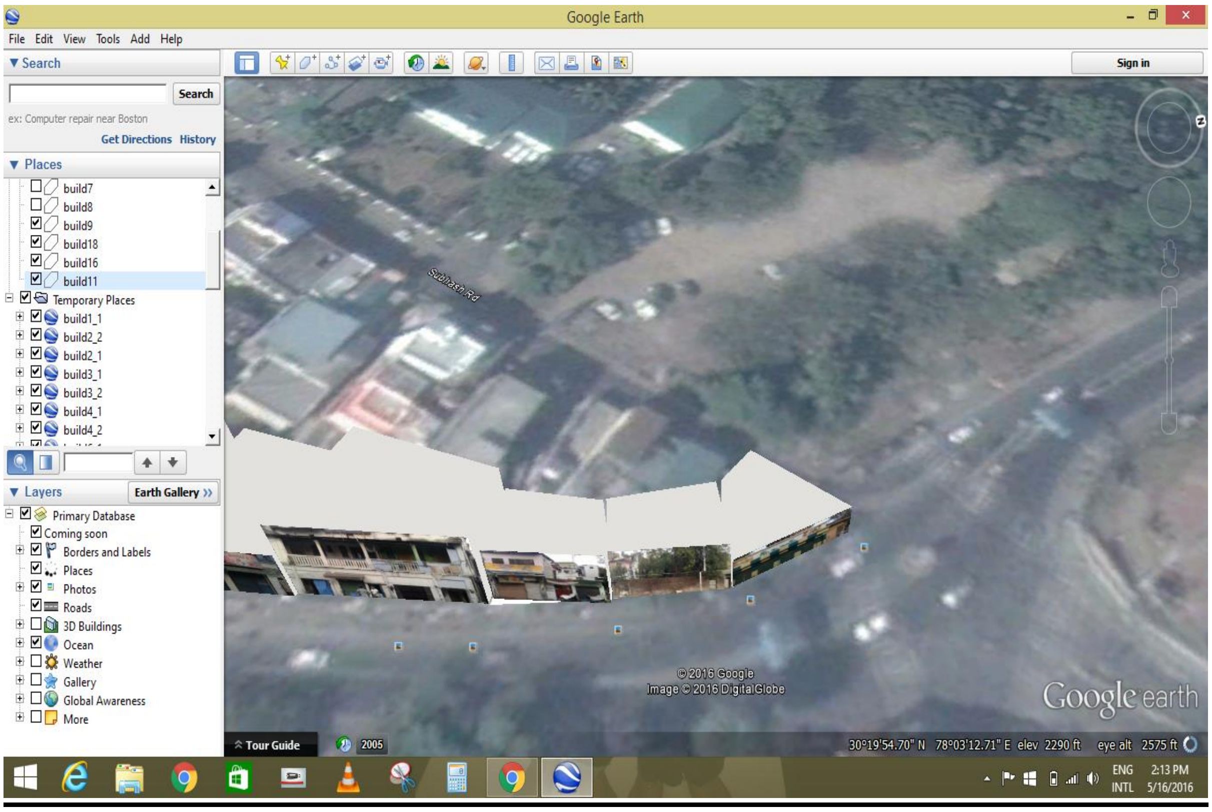


Fig 5.28: 3D view Of the buildings on Google earth

CHAPTER VI

DISCUSSION

Digital city is an important part of digital earth, which is the basis of digital and intelligent urban management achievement. The 3D visualization technology is one of the core technologies of digital city. With the deveolpment of the computer technology, remote sensing technology, virtual reality technology, and GIS technology, fast access to surface information, reconstruction of the real world in computer has become a reality. In comparison to the advancement in 3D visualization, relatively appropriate work has been accomplished in the realization of practical 3D GIS. The obvious reason remains; the transition to 3D means an even greater diversity of object type ad spatial relationship as well as very large data volumes. A 3D GIS deals with volumes. Consider a cube, Instead of looking at just its faces, there must be information about what lies inside the cube. To work, a 3D GIS requires this information to be complete and continous. It could prove a very persuasive tool in the hand of city planners, urban designers, and traffic engineers. Possibly even, they could use it to bring absract project variables like visuals impact analysis into the cost-benefit equation. The potential is definetly there to do a lot more than intersting prespective views of remotely sensed data. Perpetual improvement in hardware and software technology will ensure that a 3D GIS becomes eaasier to implement and finds some unique uses elsewhere in the discipline.

Based on the classification of the urban landscape main modeling objects, this work established a 3D surface data structure, to achieve the topography and surface feature model 3D reconstruction algorithms. And using the accelarated rendering algorithm and special effects to enhance the 3D landscape renderinng, then through 3D roaming engine, a web –based 3D model browsing in digital city is achieved.

The Future prospectives increses with the demand of increse in digital india.

CHAPTER VII

CONCLUSION

The objective of this research was to support decision for the development of the virtual city models by presenting a structures overview of 3D GIS analyses that are likely to be applied in 3D city modelling. This work highlights 3D analyses in a structured form with the objective to illustrate the wide range of 3D analyses. The overview can support the decision if a specific application requires 3D representation of urban objects. These results can contribute in the design of a 3D city model in working processes of communities. The provided results can be validated with the software testing and field work manual surveying techniques. This Obtained model of the buildings on street is in need to improve with the replacement of the placemark and easily fetching the attribute data with one click that could be anywhere on the building. It will provide an appropriate format of the final outcome. Our approach illustrate 3D city modelling and urban planning in a broad sense. It presented an approach for integrated modelling for 3D GIS, aiming at 3D model which supports 3D topology and is coupled with a user interface for querying and visualization on the Web. By linking the model design to procedures for data collection we completed the chain of the 3D modelling process. The proposed system architecture demonstrate that a “working” 3D GIS model of the city and its related attributes can be obtained by using existing technology achievements.

We expect the benefits of the designed system for “urban users”, because of

- Possibilities to query graphically objects in a virtual environment
- Visualization of the 3D spatial queries in a virtual environment
- Virtual reality navigation and exploration of 3D world
- Possibilities to visualize outcomes of queries in different file formats
- Access to the information from any office via the Internet
- Provision of moderate means to edit data over the Web and visualize the changes.

REFERENCES

- [1] “Photogrammetry,” 2007. .
- [2] A. A. Behrouzi, R. Li, and D. A. Kuchma, “Instruction Manual: Photogrammetry As a Non-Contact Measurement System in Large Scale Structural Testing,” p. 103, 2012.
- [3] T. Luhmann, S. Robson, S. Kyle, and J. Boehm, *Close-Range Photogrammetry and 3D Imaging*. De Gruyter, 2014.
- [4] T. Schenk, “Introduction to Photogrammetry,” *Dep. Civ. Environ. Eng. Geod. Sci. Ohio State Univ.*, pp. 79–95, 2005.
- [5] S. Beniwal, “‘TRIVIM’ A CLOSE RANGE PHOTOGRAMMETRY APPROACH FOR 3D MODEL GENERATION,” 2015. .
- [6] I. M. A. Review, “NRC Publications Archive Archives des publications du CNRC Image-Based 3D Modeling : A Review *.”
- [7] X. Version, “Introduction to 3D Modelling,” no. December, 2014.
- [8] www.maplandia.com/india/andhra-pradesh/chittoor/tirumala/
- [9] M. I. A. Lourakis and A. A. Argyros, “Sba,” *ACM Trans. Math. Softw.*, vol. 36, no. 1, pp. 1–30, 2009.
- [10] D. Lowe, “Object Recognition fromLocal Scale-Invariant Features,” *IEEE Int. Conf. Comput. Vis.*, pp. 1150–1157, 1999.
- [11] F. Biljecki, *The concept of level of detail in 3D city models*, no. 62. 2013.
- [12] J. Döllner, H. Buchholz, M. Nienhaus, and F. Kirsch, “Internal Version (Early Draft),” *Citeseer*, vol. 2, pp. 107–112, 2007.
- [13] N. a. Matthews, “Aerial and Close-Range Photogrammetric Technology,” *Technical Note 428*, 2008.

- [14] Y. Takase, N. Sho, a Sone, and K. Shimiya, “Automatic generation of 3D city models and related applications,” *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 34, no. Table 1, p. 5, 2003.
- [15] R. Y. Tsai, “A Versatile Camera Calibration Techniaue for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses,” no. 4, 1987.
- [16] Z. Zhang and S. Member, “A Flexible New Technique for Camera Calibration æ,” vol. 22, no. 11, pp. 1330–1334, 2000.
- [17] “Introduction — OpenCV 2.4.12.0 documentation.” [Online]. Available: http://docs.opencv.org/2.4/modules/core/doc/intro.html. [Accessed: 17-Dec-2015].
- [18]Y. M. Wang, Y. Li, and J. B. Zheng, “A Camera Calibration Technique Based on OpenCV,” pp. 403–406.
- [19]S. E. Detection and K. Localistaion, “SIFT Image Features,” pp. 2–3, 2015.
- [20]D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” vol. 60, no. 2, pp. 91–110, 2004.
- [21]I. Mitsugami, “Bundler: Structure from motion for unordered image collections,” *Kyokai Joho Imeji Zasshi/Journal Inst. Image Inf. Telev. Eng.*, vol. 65, no. 4, pp. 479–482, 2011.
- [22]Y. F. Union, A. Mvs, Y. Furukawa, B. Curless, S. Seitz, R. Szeliski, Y. Furukawa, J. Ponce, E. N. Sup, Y. Furukawa, I. Light, W. Digital, and C. Wu, “Clustering Views for Multi - view Stereo (CMVS),” no. Cvpr 2010, pp. 2–4, 2015.
- [23]Y. Furukawa, J. Ponce, E. N. Sup, C. Views, Y. Furukawa, J. Ponce, E. N. Sup, Y. Furukawa, I. Light, W. Digital, and C. Wu, “Patch - based Multi - view Stereo Software (PMVS - Version 2),” pp. 2–4, 2015.
- [24]D. Girardeau-Montaut, “CloudCompare Manual,” vol. version 2., 2013.
- [25]E. L. Finch, “COLLADA – Digital Asset Schema Release 1 . 4 . 1 Specification (

2 nd Edition),” *Elements*, no. March, 2008.

- [26]K. Lancaster, “simplekml Documentation,” 2015.
- [27]Google, “Keyhole Markup Language.” [Online]. Available: https://developers.google.com/kml/documentation/kmzarchives.
- [28] Bird, S., Hogan, D. and Schwab, J., 2010. Photogrammetric monitoring of small streams under a riparian forest canopy. Earth Surface Processes and Landforms, 35(8): 952-970. http://onlinelibrary.wiley.com/doi/10.1002/esp.2001/abstract

WEBLINKS

- http://www.maps-streetview.com/India/Dehra-Dun/roadmap.php
- http://cdodoon.gov.in/files/Dehradun_At_A_Glance.pdf
- http://www.mapsofindia.com/maps/uttaranchal/dehradun.htm#
- http://www.qgistutorials.com/en/docs/working_with_wms.html(WMS layer Working in QGis)
- https://www.google.com/earth/outreach/tutorials/odk_gettingstarted.html(odk collect app work)
- https://developers.google.com/ad-exchange/rtb/open-bidder/google-app-guide(api Manager and credential settings)
- http://www.gadgetcluster.com/2014/07/download-images-from-urls-using-python/
- http://www.diveintopython.net/html_processing/extracting_data.html
- https://youtu.be/jU3P7qz3ZrM