TABLE OF CONTENT

ACKNOWLEDGEMENT:

"No work can be completed successfully without the proper guidance & help
Of the trainer and other people."

This project report is also a manifestation of the invaluable guidance, suggestions, support and encouragement extended by various people.

We are very thankful to our project guides **Dr.Siva Reddy**, **Dr. Hina Pande, Dr. Poonam Seth Tiwari, Mr.Raghuvendra S** who helped us in understanding and visualizing the project in a better way. Their regular guidance gave us the direction to work during the project.

Our sincere thanks to our university department head **Dr. Chitralekha** and our project coordinator **DR. Jaya Kumar**. Our special thanks to our internal guides, **Dr. S.Siva Sathiya** and **Dr. V. Uma,** for providing us with this wonderful opportunity and for their valuable guidance and moral support.

We would also like to thank our parents for all the support and strength. We are very grateful for our friends **Mr. Shoubik Biswas,** for helping us to get into IIRS for our Internship and **Ms. Priya,** who is our class representative and has worked throughout to meet all the requirements for the coordination between our University and IIRS**.**

We are thankful to all the other staff members, librarian & other co-workers for their help in the completion of this project.

SYNOPSIS:

1. Introduction
    1.1 About the organization
    1.2 About the project
    1.3 Plan of the report
2. Problem definition and feasibility analysis
3. Software requirements specifications
4. System design
5. System implementation and testing
6. Conclusion and foreseeable enhancements
7. Screenshots.
8. Database design

1. INTRODUCTION:

1.1. ABOUT THE ORGANIZATION:

Indian Institute of Remote Sensing (IIRS)

The Indian Institute of Remote Sensing (IIRS) - an ISO 9001:2008 institute, is a constituent unit of Indian Space Research Organization (ISRO), Department of Space, Govt. of India. Since its establishment in 1966, IIRS is a key player for training and capacity building in geospatial technology and its applications through training, education and research in Southeast Asia. The training, education and capacity building programmes of the Institute are designed to meet the requirements of Professionals at working levels, fresh graduates, researchers, academia, and decision makers. IIRS is also one of the most sought after Institute for conducting specially designed courses for the officers from Central and State Government Ministries and stakeholder departments for the effective utilization of Earth Observation (EO) data.

To widen its outreach, IIRS has started live and interactive Distance Learning Programme (DLP) since 2007. IIRS has also launched e-learning course on Remote Sensing and Geo-information Science since August, 2014.

The Institute has a strong, multi-disciplinary and solution-oriented research agenda that focuses on developing improved methods/ techniques for processing, visualization and dissemination of EO data & Geo-information for various societal applications and better understanding of Earth's system processes. Currently, Microwave, hyper spectral and high-resolution EO data processing and their applications are some of the prime research areas. State-of-the-art laboratory and field-based instrumentation and observatories network help meeting the research goals and objectives.

IIRS hosts headquarters of Centre for Space Science and Technology Education in Asia and the Pacific (CSSTEAP), affiliated to the United Nations and provides support in conducting the Remote Sensing and GIS training and education programmes. IIRS also plays a key role in the activities of Indian Society of Remote Sensing (ISRS), which is one of the largest non-governmental Scientific Societies in the country.

1.2. ABOUT THE PROJECT:

This project is an open source tool for digital documentation of heritage sites which allows the user to perform various task inside the software by authenticating himself to the software with the allowance provided by the administrator. After authentication the project gives the user various functionalities (Point cloud processing, Image processing and Database querying) organized in a simple tool. The project brings forth the point cloud processing procedures that are: visualize point cloud, registration, translate/rotate, registration, unroll and then finally rasterize point cloud to give 2D image. Then the image is processed. The domain of image processing provides unique functionalities, since it only considers an image as valuable entity. Some of its applicability is in Edge Detection, Corner Detection, RGB to HIS, HIS to RGB, Texture Analysis and Change Detection. The project uses python and C++ as a programming language. As python is open sourced, easy to understand, highly readable and has a powerful syntax it has been used. And some functionality is accomplished using C++. Firstly python script is used to call the Qtcreator framework. The libraries are of high importance in the project. This project is intended to be used by any person with elementary knowledge of computing and who knows handling point cloud. The only contribution of user should be of providing point cloud to be analyzed. The working of this project is based on point cloud and image processing concept. Moreover, after converting 3D point cloud to 2D image, the point and image attributes are stored in the PostgreSQL database and can be visualized by the user through interface. The user should be equipped with computing technology required for the working of this project, given the operational load to be handled. With only an input, the machine is left with an algorithm to give an output.

2. PROBLEM DEFINITION:

The heritages all over the world are the properties a country can flaunt about. These heritages need to be maintained in order to sustain them over centuries. Precise documentation of cultural heritage status is essential for its protection and scientific studies carried out during the restoration and renovation process. The maintenance can though be done manually but in this and upcoming digital centuries, where we can doubt the remanence of manpower, we need to depend upon software which in turn raises the need for such a software that can maintain the heritages and their sustainability.

Purpose:

This project deals with development of software that can Analyze, Detect and Report about the Changes and Damages taking place or tending to occur over time, such that the heritages can be maintained well throughout.

The project requires the point cloud data from the Arial LiDAR (Light Detection and Ranging) Laser Scanners for any heritage site. This point cloud data gives a 3D image of the entire site. This data can be analyzed for changes over different time by comparing two or more LiDAR data. The differences to be detected can be the following:

- Edges of the building
- Corners present in the buildings
- Change between two data captured over different time like peel off, etc.,
- Texture Analysis that can analyze and detect the texture and report about issues like algae growth, moisture accumulation on the surface, etc.

These things in total help us in a complete maintenance of the heritages as far as possible through 3D Digital Documentation.

The data collected and analyzed can be stored in the database in multiple forms for future references.

Functionalities:

The project gives the user three major functionalities, organized in a simple tool:

- Point cloud processing,
- Image processing
- Database querying.

The project brings forth the *point cloud processing* procedures that are:

- Visualize point cloud
- Translation and Rotation
- Registration
- Unroll
- Rasterize point cloud to give 2D image

Then the image is processed. The domain of image processing provides unique functionalities, since it only considers an image as valuable entity. Some of its applicability is:

- Edge Detection,
- Texture Analysis,
- Change detection.

The database querying portion includes functionalities as follows:

- User Registration
- User Login
- Administration maintenance of Users
- Storage of data in three formats,
  - Raster
  - Matrix
  - Image
- Retrieval of data for its processing

Languages used in the project are:

- *C++*: QtCreator is an integrated development environment that provides tools to design and develop application with the Qt application framework.
- *Python:* python is open sourced, easy to understand, highly readable and has a powerful syntax which made us chose it.
- *PostgreSQL*: The database parts with its queries have been written using PostgreSQL.
- *CSS*: The UI designs and its style sheets have been accomplished using CSS.

The libraries are of high importance in the project. This project is intended to be used by any person with elementary knowledge of computing and who knows handling point cloud. The only contribution of user should be of providing point cloud to be analyzed. The working of this project is based on point cloud and image processing concept. Moreover, after converting 3d point cloud to 2d image, the point and image attributes are stored in the PostgreSQL database and can be visualized by the user through interface. The user should be equipped with computing technology required for the working of this project, given the operational load to be handled. With only an input, the machine is left with an algorithm to give an output.

FEASIBILITY ANALYSIS:

Technical Feasibility:

This project is technically feasible as all technologies used in this project are all easily available; all the python and C++ libraries used in this project are open source libraries easily available over the internet. So the project is technically feasible.

Schedule Feasibility:

This project is feasible as this project was completed in available time, the schedule that was decided before the start of the project has been strictly followed and hence the project was completed in the given time.

Economic Feasibility:

This project was economically feasible because we have used only freely available open source libraries which were available at free of cost, hence economically not much amount has been spent for this project.

Technical requirements:

The hardware and software requirements are very less. It occupies very less space. Python, QtCreator and its libraries being open sourced was easily be accessed. The language is easy to understand and its libraries make it more powerful. The user can also be a non- programmer and can easily access the tool due to the Graphical user interface which was made available to him.

Profitability:

The tool is a culmination of point cloud processing and image processing techniques which can be easily accessed, the graphical user interface provides a flow chart of the steps that makes it much easy for the user to understand and use the software. The project aims to do a digital documentation of any monument, to do so it provides many point cloud processing and image processing tools. The extended objective of the project is to maintain the heritage sites which helps in detecting the damages in those sites due to any factor (time, nature etc.) by making comparison between two different images captured at different time.

Need:

All the functionalities available in this tool are provided by many tools but the highlight is that No software exist which provides all in one plate. The project gives the user all the functionalities as a single application.

While working on other software packages we have to switch from one software to other, all the point cloud processing task could be carried out in one software and then to proceed further we have to move the output data from that software to other, which causes inconvenience and loss of data.

In this software the GUI provides a flow chart, referencing which user can easily understand the steps he needs to carry to proceed in a systematic way without any confusion.

3. SOFTWARE REQUIREMENT SPECIFICATION (SRS):

The tool requires the following hardware, software requirements at the developer as well as the user's end and the following user characteristics:

Hardware Requirement:

- **RAM:** At least 4 GB
- **HDD:** 5 GB or more (Free space excluding data size)
- **Processor:** Intel(R) Core(TM) i3(or equivalent) @ 2.70GHz

Software Requirement:

SOFTWARE	TYPE / PLATFORM	VERSION
OS	Windows OS	Windows 7 and above
Qt Creator	Microsoft Visual Studio 2013	Qt Creator 4.0.1 Based on Qt 5.6.1(MSVC 2013, 32 bit)
OpenCV	Microsoft Visual Studio 2014	OpenCV 3.2.0-vc14
Cmake	Qt 5.6.0	Cmake 3.7.2
Python	-	Python 2.7
PostgreSQL	-	PostgreSQL 9.3
PostGIS	PostgreSQL 9.3	PostGIS 2.2
GDAL	-	GDAL 1.1

QT Creator:

Qt Creator is an integrated development environment (IDE) that provides you with tools to design and develop applications with the Qt application framework. Qt is designed for developing applications and user interfaces once and deploying them to several desktop, embedded, and mobile operating systems. Qt Creator provides you with tools for accomplishing your tasks throughout the whole application

development life-cycle, from creating a project to deploying the application to the target platforms.

OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. It has C++, C, and Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available.

Cmake:

Cmake is an extensible, open-source system that manages the build process in an operating system and in a compiler-independent manner. Unlike many cross-platform systems, Cmake is designed to be used in conjunction with the native build environment. Simple configuration files placed in each source directory (called CMakeLists.txt files) are used to generate standard build files (e.g., make files on UNIX and projects/workspaces in Windows MSVC) which are used in the usual way. Cmake can generate a native build environment that will compile source code, create libraries, generate wrappers and build executables in arbitrary combinations. Cmake supports in-place and out-of-place builds, and can therefore support multiple builds from a single source tree. Cmake also supports static and dynamic library builds.

PostgreSQL

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It includes most SQL: 2008 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. PostgreSQL runs stored procedures in more than a dozen programming languages, including Java, Perl, Python, Ruby, Tcl, C/C++, and its own PL/pgSQL, which is similar to Oracle's PL/SQL. We have used Python with it. PostgreSQL is suitable for storing spatial data.

PostGIS:

PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.

PyQT4:

PyQt4 is a comprehensive set of Python bindings for Digia's Qt cross platform GUI toolkit. PyQt4 supports Python v2 and v3.

Python2.7:

Python is a widely used high-level programming language for general programming an interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.

User Characteristics:

The user of this system must have this following characteristic:

- User must be a computer literate.
- As the interface of the system is English, the use must be well aware of English.
- User should be well aware of image processing and point cloud functionalities.

MODULES:

VISUALIZATION OF POINT CLOUD:

Point Cloud can be visualized when you click on open button.

TRANSLATION/ROTATION:

Translation is a rigid transformation of the plane that moves every point of the point cloud to a constant distance in a specific direction. Translation and Rotation of point cloud data requires a point cloud to be selected for translation. Translation is applied to align the point cloud with the reference point cloud for the purpose of registration and geo-referencing.

For transformation the project initially checks for the selected point cloud in the active window and extracts the center point of that selected point cloud. Then the translation vector is defined as $(t_x\ t_y\ t_z)$= (0, 0, 0).

The user has to select the transforms (t_x, t_y, t_z) to apply translation. Then a translation of a vector space adds a vector (say x) to every vector in the space, which means it is the transformation g (**v**):**v**→**v**+**x**.

The transformation of vector space has the property that the transformation of vector is the sum of the transformations of its components.

Adding a negative value (subtracting), indicates movement left and/or down, while adding a positive value indicates movement right and/or up.

Rotation is a transformation which turns a point cloud about a fixed point, *C*, called the center of rotation. When working in the coordinate plane, the center of rotation should be stated, and it is not assumed to be at the origin. The point cloud can be rotated with the angle of rotation α, β, γ along x, y and z axis. During rotation, every point is moved the exact same degree arc along the circle defined by the center of the rotation and the angle of rotation.

There are three different sets of rotation in the three dimensional transformation matrixes; one for each axis can be rotated around.

X axis rotation: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$

Y axis rotation: $\begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}$

Z axis rotation: $\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The standard mouse interactions with the 3D view are used to modify the selected entities position (instead of the current camera):

- left click: rotate
- right click: translate

Properties preserved under a **translation** and **rotation** are:

1. **Distance** (lengths of segments remain the same)
2. **Angle measures** (remain the same)
3. **Parallelism** (parallel lines remain parallel)
4. **Co linearity** (points remain on the same lines)
5. **Orientation** (lettering order remains the same)

The transformation can also be paused by clicking on pause button in order to modify the orientation and position and then can restart the transformation by again clicking on the pause button. On pausing transformation the algorithm freezes the entities from moving and on unpausing the transformation releases the entities to move. The shortcut has also been triggered for pausing the transformation. Key space will also pause the transformation.

The applied Transformation can be cancelled by clicking on cancel button. The application brings back all the points in the point cloud by resetting the GL Transformation. The shortcut has also been triggered for cancelling the transformation. Escape key can also be used for cancelling.

After transforming the point cloud the transformation is applied by clicking apply button. Final transformation is applied by subtracting the summation of the center point of the point cloud and the translation matrix and the product of center point and the rotation matrix. The shortcut key used for apply is enter key.

REGISTRATION:

Image registration is the process of aligning two or more images of the same scene. This process involves designating one image as the reference image, also called the fixed image, and applying geometric transformations or local displacements to the other images so that they align with the reference.

Here we are using point cloud data (LiDAR data). The 3D point cloud of the object surface can be obtained by optical equipment such as laser scanners, which can provide the basis for the establishment of the 3D model of the object. However, it is impossible to obtain all the point cloud information of the object at the same viewpoint because the 3D scanning device has a limitation on the field of view or because of the complex geometry of the object itself. In order to obtain the complete point cloud data of the measured object, it is necessary to integrate the part of the surface point cloud data obtained from different angles. The purpose of point cloud registration is to find a 3D rigid body transformation; so that the 3D coordinates of the point cloud at different angles can be correctly matched and overlapped.

- Given the coordinates of a set of points measured in two Cartesian coordinate systems (left, right) find the rigid transformation (Rigid transformation is a combination of translation and rotation), *T* for registration. There will be two ways:

    1. *POINT PAIR REGISTRATION* - Pairing between points is known, closed form (analytic) solutions (Horn Point Cloud Registration Method).

    2. *FINE REGISTRATION* - Pairing between points is unknown, iterative solutions (require initialization and only guarantee convergence to local optimum). Iterative solution that used here is ICP (Iterative Closest Point) algorithm.

Images of the same target area taken at different times can be registered and fused to detect changes over a certain period of time. The information that can be obtained from an image depends on the sensor used. In remote sensing applications different sensors like multispectral, infrared, panchromatic, optical etc, are used to obtain images.

It has been widely used in change detection, image fusion and other related areas. In order to integrate different kinds of sensor data and different temporal data, image registration is an indispensable pre-processing tool in integrating multi-source and multi-temporal images. In change detection process, the image registration accuracy directly influences the accuracy of change detection result.

FINE REGISTRATION:

This method implements the ICP algorithm (Besl et al.) for the registration of two point cloud automatically.

This tool can automatically finely registers two entities.

Main assumptions are:

- both clouds are already roughly registered using translate and rotation that minimizes the error
- both clouds should represent the same object or at least have the same shape (at least on their overlapping parts

The ICP algorithm is an iterative alignment algorithm that works in three phases:

1) Establish correspondence between pairs of features in the two structures that are to be aligned based on proximity,

2) Estimate the rigid transformation that best maps the first member of the pair onto the second and then

3) Apply that transformation to all features in the first structure. These three steps are then reapplied until convergence is concluded. Although simple, the algorithm works quite effectively when given a good initial estimate.

During this process, the registration error (slowly) decreases. We can stop this process either after a maximum number of iterations, or as soon as the error (RMS) difference between two iterations becomes lower than a given threshold. The smaller this threshold is, the longer it will take to converge, but the finer the result should be.

ICP ALGORITHM:

Iterative closest point (ICP) registration is an accurate and reliable method for registration of free form surfaces. ICP algorithm is used to find the rigid transformation T between the target point set S and the reference point set M so that the two matching data satisfy the optimal match under some kind of metric criterion. Assuming that the coordinates of the target point set S are $\{S_i | S_i \in R^3, i = 1, 2, \cdots, N_s\}$, the coordinates of the reference point set M are $\{M_i | M_i \in R^3, i = 1, 2, \cdots N_M\}$, in the k-th iteration, the coordinates of the corresponding point corresponding to the coordinates of

the point set S are $\{M_i^k || M_i^k \in R^3, i = 1, 2, \cdots N_M \}$. The transformation matrix between S and M^k is calculated and the original transform is updated until the distance between the data is less than the given threshold τ. The ICP algorithm steps are as follows:

- Calculate the corresponding point $M_i^k \in M^k$ in the reference set M so that $\|M_i^k - S_i^k\|$=min;
- Calculate the rotation matrix R^k and the translation vector T^k so that

  $\sum_{i=1}^{N} \left|\left| R^k S_i^k + T^k - M_i^k \right|\right|^2$ = min;
- Calculate $S^{k+1}=\{S_i^{k+1} || S_i^{k+1}=R^k S_i^k+T^k, S_i^k \in S\}$;
- Calculate $d^{k+1}= \sum_{i=1}^{N} \left|\left| S_i^{k+1} - M_i^k \right|\right|^2$;
- If d^{k+1} is not less than the given τ value, return (1) until $d^{k+1} < \tau$ or iterations k is greater than the present maximum number of iterations.

POINT PAIR REGISTRATION:

This tool lets the user align two entities by picking at least three equivalent point pairs in both entities. This method is very useful to align clouds quite precisely. It's even sometimes the only way to get a fine result (typically if the two clouds have great differences on large extents, in which case the ICP registration won't work properly).

Using this we can register two point clouds by picking the required points and also register the image with the ground control points that are manually given by the user.

This tool uses Horn Point Cloud Registration Algorithm. It returns the absolute orientation between the two set of point. The output transformation is from left to right co-ordinate system. Determines the best quaternion and optionally a scale to bring the align cloud closer to the reference cloud.

During this process, you can hide either the 'reference' or the 'aligned' cloud with the dedicated checkboxes. You can also freely rotate/translate the view point, enlarge points, zoom in or out, enable or disable OpenGL shaders, etc.

Each time you pick (or manually input) a point, a new entry appears in the corresponding table (either Aligned or Reference), as well as a marker in the 3D view (aligned markers begin with 'A', and reference markers begin with ... 'R').

HORN METHOD:

This method gives a closed-form solution for the least-squares problem of absolute orientation. It provides the best rigid body transformation between two coordinate systems given measurements of the coordinates of a set of points that are not collinear. It uses unit quaternion to represent rotations. This method is used to find relationship between coordinate systems (absolute orientation) that gives the closed form solution.

- Find centroids $\overline{r_l}$ and $\overline{r_r}$ of the two sets of measurements where $\overline{r_l} = \frac{1}{n}\sum_{i=1}^{n} r_{l,i}$ and $\overline{r_r} = \frac{1}{n}\sum_{i=1}^{n} r_{r,i}$. $r_{l,i}$ And $r_{r,i}$ are the measured co-ordinates in left and right hand system.
- Subtract them from all measurements.
- For each pair of coordinates, compute the possible product $x_l' x_r'$, $x_l' y_r'$, … $z_l' z_r'$ of the components of the two vectors.
- These are added up to obtain S_{xx}, S_{xy}, …S_{zz} where $S_{xx} = \sum_{i=1}^{n} x_{l,i}' x_{r,i}'$
  $S_{xy} = \sum_{i=1}^{n} x_{l,i}' y_{r,i}'$
- Compute the 10 independent elements of the 4x4 symmetric matrix N

$$\begin{vmatrix} (S_{xx}+S_{yy}+S_{zz}) & S_{yz}-S_{zy} & S_{zz}-S_{zz} & S_{xy}-S_{yz} \\ (S_{yz}-S_{zy}) & (S_{xy}-S_{yy}-S_{zz}) & S_{xy}+S_{yz} & S_{zz}+S_{xz} \\ (S_{zz}-S_{zz}) & S_{xy}-S_{yz} & (-S_{xx}+S_{yy}-S_{zz}) & S_{yz}+S_{xy} \\ (S_{xy}-S_{yz}) & S_{zz}+S_{zz} & S_{yz}+S_{zy} & (-S_{xx}-S_{yy}+S_{zz}) \end{vmatrix}$$

- From these elements, calculate the coefficients of the quartic that must be solved to get the eigenvalues of N
- Pick the most positive root and use it to solve the four linear homogeneous equations to get the eigenvector. The quaternion representing the rotation is a unit vector in the same direction.
- Compute the scale from the symmetrical form formula, i.e. the ratio of the root-mean-square deviations of the measurements from their centroids.
- Compute the translation as the difference between the centroid of the right measurements and the scaled and rotated centroid of the left measurement.

Finally, after registering point cloud an RMS error report will be displayed which contain the transformation matrix and the overlap percentage.

UNROLL:

This method ‘unrolls’ a point cloud from a cylinder (or conical) shape onto a plane

Cylinder*:* select a single point cloud.

To unroll a cylindrical shape, the parameters are:

- axis of revolution (X, Y or Z)
- cylinder radius
- and optionally a point on the axis

Cone*:* select a single point cloud.

To unroll a conical shape, the parameters are:

- axis of revolution (X, Y or Z)
- half angle (this is the *aperture* angle at the cone apex) in degrees
- the cone apex position in 3D

Straightened Cone*:* select a single point cloud.

This method unrolls a cone-shaped cloud as if it was a cylinder, by first 'straightening' the cone walls.

To unroll a straightened conical shape, the parameters are:

- axis of revolution (X, Y or Z)
- half angle (this is the *aperture* angle at the cone apex - in degrees)
- the cone apex position
- the base radius of the 'straightened' cone (= cylinder)

RASTERIZATION:

The main purpose of this tool is to rasterize a point cloud (i.e. convert it to a 2.5D grid) and then export it as a new image to perform further image processing. Rasterization is the task of taking an unrolled point cloud and converting it into a raster image (pixel or dots), to store in any format. In normal usage, the term refers to the rasterization of 3D models onto a 2D plane for display on a computer screen ("screen space") is often carried out by fixed function hardware within the graphics pipeline.

We can say that rasterization is used to solve the visibility problem. Visibility consists of being able to tell which parts of 3D objects are visible to the camera. Some parts of these objects can be hidden because they are either outside the camera's visible area or hidden by others objects.

The terms rasterization comes from the fact that polygons (triangles in this case) are decomposed in a way, into pixels and as we know an image made of pixels is called a raster image.

Rasterization usually reduces the image to one flat layer, and thus limits edit ability to a minimum. You will want to keep a non-rasterized version of your file archived at all times, just to make adjustments later, if necessary

The user must define the main (raster) grid generation parameters:

- the grid step size (update the resulting grid size below so that the user can check that the grid is neither too big nor too small before actually generating the grid)
- the projection direction (X, Y or Z - default: Z)
- how the 'height' of each cell grid will be computed:
    - minimum height of all points falling in this cell
    - average height of all points falling in this cell
    - maximum height of all points falling in this cell

Export raster:

The (raster) grid can be exported to several destinations.

Cloud

The grid can be exported as a new cloud (see the "Cloud" button in the 'Export' tab in the bottom-left corner). The grid is always exported as a 3D cloud (with the chosen 'height' as the 'Z' dimension). A 'height' scalar field is also generated by default.

Several additional scalar fields can be generated:

- population: number of input points falling in each cell
- min height: minimum height of the points falling in each cell
- average height: average height of the points falling in each cell (may be redundant with the default 'height' scalar field)
- max height: maximum height of the points falling in each cell (may be redundant with the default 'height' scalar field)
- average height: average height of the points falling in each cell (may be redundant with the default 'height' scalar field)
- height std. dev.: standard deviation of the height values of the points falling in each cell
- height range: range of the height values of the points falling in each cell

Image

The grid can be exported as a simple image file.

ASCII matrix

The grid can be exported as an array/matrix of height values saved as an ASCII file.

This file should be easily imported in Excel for Matlab for instance. There's no file header. The number of rows is simply the number of lines in the file, and the number of columns corresponds to the number of values found on each line (should always be the same).

DISPLAY METHODS:

Full screen:

This method simply makes the main application window full screen.

Refresh:

This method simply forces the active 3D view to refresh its content (all OpenGL primitives are redrawn)

Toggle Centred Perspective

This method toggles the current projection of the active 3D view between the 'orthographic' and 'object-centred perspective' modes

Perspective view:

The perspective projection consists in projecting a 3D scene on a 2D plane as if it was seen by the human eye (or a single camera).

Object-centred Perspective:

In the 'object-centred perspective' mode, the object rotates when the user moves the mouse while pressing the left button (the camera orientation is fixed). Using the mouse wheel will make the object farther or nearer from the current point of view.

Viewer-based Perspective:

In the 'viewer-based perspective' mode, the camera rotates when the user moves the mouse while pressing the left button. And using the mouse wheel make the camera move forward or backward in the current viewing direction.

Lock rotation about vert. axis:

This method simply locks the camera rotation around the vertical (Z) axis (in the active 3D view).

Bubble-view (polar):

This viewing mode can only be properly enabled via a 'GBL sensor' entity. They are generally associated to point clouds coming from proprietary files (PTX, FARO, DP, etc.). To enable it, simply browse the sensor entity properties and click on the 'Apply' button ('Apply viewport' item). While in this mode, the camera position is fixed (normally at the sensor centre) and the user can only rotate it (with the left mouse button) or change the zoom (with the mouse wheel).

Render to file:

This tool can 'render' the current 3D view as an image file (most of the standard file formats are supported). It can also apply a zoom so as to render the screen to a much higher resolution than the actual screen resolution.

You must first set the output image filename (you can use the '...' button to browse a particular file or folder on your computer). Most common image file formats are supported (bmp, jpg, png, etc.). By default the output image will have the same resolution as the 3D view (i.e. the same size in pixels). With the 'zoom' factor, you can increase the rendered image resolution (the resulting size is displayed on the right). Application will render the 3D view content off-screen in a potentially much larger buffer than you actual screen

Adjust zoom

This tool let the user define the current zoom, potentially in a very accurate way (e.g. in order to get a very specific dimension per pixel for instance)

Test frame Rate

This tool makes the active 3D view spin for a short time (~10 seconds) in order to estimate the average 'fps' (frame per second). The result is displayed in the Console.

Console

The console is a dockable widget where all messages (standard information, warnings and errors) are traced. Some algorithms also output results in it.

Reset all GUI elements

By default Application saves the current GUI configuration (position and visibility of the toolbars, etc.) when quitting. This tool can be used to restore the original configuration.

EDIT MENU:

CROPPING:

- This tool is used to crop one or several clouds inside a 3D box.
- It is required to select one or several clouds before starting the crop.

3D box editing dialog by default, the box is initialized to the bounding-box of all selected clouds (this default box can be restored anytime by clicking on the 'Default' button).

The user can define the cropping box in multiple ways:

- By defining the center and dimensions of the box [default]
- By defining the min corner and the dimensions of the box
- By defining the max corner and the dimensions of the box

Notes:

- For cubical boxes (the same dimension in all directions), you can check the 'keep square' checkbox.
- You can recall the previous box (if you call the tool several times) by clicking on the 'Last' button. The 'Last' button only appears if the tool has been called at least twice.

Eventually on clicking the 'OK' button the input clouds is cropped and the corresponding subsets is created (or on clicking the 'Cancel' button the process can be canceled).

SEGMENTATION:

This tool allows the user to interactively segment the selected entities by defining a 2D polygon (or a rectangle) on the screen. This process can be repeated multiple times, changing the orientation of the entities each time, so as to properly segment the entities in 3D. Each time the user can decide to keep the points (or triangles) falling inside or outside the polygon border.

Polygon edition mode

By default the tool starts in 'polygonal' editing mode. This means that you can start drawing the polygon right away:

- left click: create a new polygon vertex
- Right after the first vertex is created, you'll see that the first polygon edge will start to "follow" the mouse cursor. You have to define the position of the second vertex (left click) in order to 'fix' it. This process will start over with the next edge and so on.
- Right click: stop the polygon edition (warning: the currently 'floating' vertex won't be added to the polygon)

Rectangle edition mode

You can switch the 'Polygon edition' mode to 'Rectangle edition' mode by clicking on the down arrow next to the 'polygon' icon:

In 'Rectangle edition' mode you have to left click a first time to define the first corner of the rectangle then click a second time to define the opposite corner (alternatively you can keep the left mouse button pressed and the opposite corner will be created once you release the button).

Paused mode

In paused mode, the mouse can be used to modify the entities orientation and position in the standard way.

The user has multiple choices:

- modify the current entities orientation and segment more points (click on the button to leave the 'paused' mode and define a new polygon/rectangle)
- reset the current selection
- validate the current segmentation and create two clouds: one with the selected points and one with the others
- validate the current segmentation and create only one cloud with the visible points - the other points will be deleted
- cancel the segmentation process(no modification will occur)

MERGING:

Merging is the process of taking two or more groups of point cloud data and combining the data into a single unified set. Both Fine Registration and Align Point Pair Registration overlays both the point cloud data and then the overlaid data is merged using merging tool for further processing.

IMAGE PROCESSING:

EDGE DETECTION:

It is an Image Processing Technique for finding the boundaries of object within images. It works by detecting the discontinuities in brightness. Edges are the abrupt change in the intensity of pixels.

The Process of identifying the edges in an image is used as the fundamental assets in image analysis. To identify the shape information of an image we need to identify the edges because most of the shape information of an image is enclosed in edges. Detecting the edges of the image reduces the unnecessary information in the image while preserving the structure of the image and extracts the important feature of the image.

Procedure:

1. The obtained raster image from rasterization is load for detecting the edges in that image.
2. Then the image is converted to gray image from RGB image.
3. The gray scale image is smoothened by using Gaussian filter which suppress as much noise as possible without destroying true edges.
4. The noise free image if taken as input for Enhancement which applies differentiation. Often the points that lie on an edge are detected by:
    - Detecting the local maxima or minima of the first derivative.
    - Detecting the Zero-crossings of the second derivative.

Five different edge detection techniques have been implemented for edge enhancement. For each pixel the gradient is calculated based on a 3x3 neighborhood around that particular pixel.

$(i-1, j-1)$	$(i-1, j)$	$(i-1, j+1)$
$(i, j-1)$	(i, j)	$(i, j+1)$
$(i+1, j-1)$	$(i+1, j)$	$(i+1, j+1)$

I. LAPLACIAN EDGE DETECTION:

Laplacian Operator is also a derivative operator which is used to find edges in an image. Laplacian is a second order derivative mask. In this mask we have two further classifications one is Positive Laplacian Operator and other is Negative Laplacian Operator. Another difference between Laplacian and other operators is that unlike other operators Laplacian didn't take out edges in any particular direction but it take out edges in following classification.

- Inward Edges
- Outward Edges

Laplacian edge detection can be applied only on smooth image. Gaussian smoothing operator is used to smoothen the image that is used to blur images and remove details and noise.

The formula for the Laplacian of a function *f (x, y)* is

$$\nabla 2\, f = \partial 2\, f/\partial x2 + \partial 2\, f/\partial y2$$

Approximating $\nabla 2\, f$:

$$\partial 2\, f/\partial x2 = f\,(i, j+1) - 2\, f\,(i, j) + f\,(i, j-1)$$
$$\partial 2\, f/\partial y2 = f\,(i+1, j) - 2\, f\,(i, j) + f\,(i-1, j)$$

$$\nabla 2\, f = -4\, f\,(i, j) + f\,(i, j+1) + f\,(i, j-1) + f\,(i+1, j) + f\,(i-1, j)$$

We can easily build the filter on the basis of the kernel:

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

All the pixels adjacent to the pixel at (i, j) are added with -1 and the pixel at (i, j) is add with 4 and all the corner pixels are with 0. The negative value in the kernel will return near zero value when traversing region is of constant intensity and positive or negative value

when traversing region is of change intensity. If the kernel value crosses zero then it indicates that the sudden change in the intensity of the pixels and that indicates edge.

II. SOBEL EDGE DETECTION:

The operator calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point and therefore how likely it is that part of the image represents an edge, as well as how that edge is likely to be oriented.

The Sobel operator is the magnitude of the gradient computed by **Magnitude** $\overline{\mathbf{G}=}$ $\sqrt{S_x^2 + S_y^2}$

To reduce the complexity of above square root calculation, gradient magnitude is sometimes approximated by absolute summation i.e. S = abs (S_x) + abs (S_y).

Using this information, we can also calculate the gradient's direction: $\boldsymbol{\theta = a \tan^2 S_x + S_y}$

Where the partial derivatives are computed by

s_x= *(a2 + ca3 + a4) - (ao + ca7 + a6)*

s_y= *(ao* + cal + *a2) - (a6 + ca5 + a4)*

With the constant c = 2.

Like the other gradient operators, $S_x$and S_y can be implemented using the kernels:

S_x =

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

S_y =

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

Here the kernel S_x is sensitive to changes in the *x* direction, i.e., edges that run vertically, or have a vertical component. Similarly, the kernel S_y is sensitive to changes in *y* direction, i.e., edges that run horizontally, or have a horizontal component. But with these two kernels to give a single measure of the presence of an edge we have to combine the results of convolution.

The two gradients computed at each pixel ($S_x$and S_y) by convolving with above two kernels can be regarded as the *x* and *y* components of gradient vector. This vector is oriented along the direction of change, normal to the direction in which the edge runs.

III. PERWITT EDGE DETECTION:

The **Prewitt operator** is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Prewitt operator is either the corresponding gradient vector or the norm of this vector. The Prewitt operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical directions and is therefore relatively inexpensive in terms of computations like Sobel operators. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

The Prewitt operator uses the same equations as the Sobel operator, except that the constant c = 1.

Therefore:

S_x =

-1	0	1
-1	0	1
-1	0	1

S_y =

-1	-1	-1
0	0	0
1	1	1

Note that, unlike the Sobel operator, this operator does not place any emphasis on pixels that are closer to the center of the masks.

IV. SCHARR EDGE DETECTION:

This filter is somehow same as sobel operator. OpenCV addresses this inaccuracy for kernels of size 3 by using the CV Scharr function. This is as fast but more accurate than the standard Sobel function. It implements the following kernels:

-3	0	3

S_x =

-10	0	10
-3	0	3

S_y =

-3	-10	-3
0	0	0
3	10	3

V. CANNY EDGE DETECTION:

Canny edge detection is a multi-step algorithm that can detect edges with noise suppressed at the same time.

1. Smooth the image with a Gaussian filter to reduce noise and unwanted details and textures.

$$g(m,n) = G_\sigma(m,n) * f(m,n)$$

Where:

$$G_{\sigma} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{m^2+n^2}{2\sigma^2}\right)$$

2. Compute gradient of g(m,n) using any of the gradient operators (Roberts, Sobel, Prewitt, etc.) to get:

$$M(n,n) = \sqrt{g_m^2(m,n) + g_n^2(m,n)}$$

And

$$\theta(m,n) = \tan^{-1}[g_n(m,n)/g_m(m,n)]$$

3. Threshold M:

$$M_T(m,n) = \begin{cases} M(m,n) & if\ M(m,n) > T \\ 0 & otherwise \end{cases}$$

Where T is so chosen that keeps all edge elements while most of the noise is suppressed.

4. Suppress non-maxima pixels in the edges in M_T obtained above to thin the edge ridges (as the edges might have been broadened in step 1). To do so, check to see whether each non-zero $M_T(m,n)$ is greater than its two neighbors along the gradient direction $\boldsymbol{\theta}$ (m,n). If so, keep $M_T(m,n)$ unchanged, otherwise, set it to 0.
5. Threshold the previous result by two different thresholds T_1 and T_2 (where$T_1 < T_2$) to obtain two binary images T_1 andT_2. Note that T_2 with greater T_2 has less noise and fewer false edges but greater gaps between edge segments, when compared to T_1 with smallerT_1.
6. Link edge segments in T_2 to form continuous edges. To do so, trace each segment in T_2 to its end and then search its neighbors in T_1 to find any edge segment in T_1 to bridge the gap untill reaching another edge segment inT_2.

CORNER DETECTION:

Corner detection is an approach used within computer vision systems to extract certain kinds of features and infer the contents of an image. We are using corner detection to prepare a digital documentation which describes the monument or cultural heritage under consideration in a better way. The question arise that what makes corner so special since it is the intersection of two edges, it represents a point in which the directions of these two edges change. Hence, the gradient of the image (in both directions) have a high variation, which can be used to detect it.

Consider a grayscale image I. We are going to sweep a window w (x, y) (with displacements u in the x direction and v in the right direction) I and will calculate the variation of intensity.

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

Where:

W (x, y) is the window at position (x, y)

I(x, y) is the intensity at (x, y)

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

Since we are looking for windows with corners, we are looking for windows with a large variation in intensity. Hence, we have to maximize the equation above, specifically the term:

$$\sum_{x,y} [I(x+u, y+v) - I(x,y)]^2$$

$$E(u,v) \approx \sum_{x,y} \left[I(x,y) + uI_x + vI_y - I(x,y)\right]^2$$

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

$$E(u,v) \approx [u\ v] \left(\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

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

A score is calculated for each window, to determine if it can possibly contain a corner:

$$R = \det(M) - k(trace(M))^2$$

Where:

det (M) = λ1λ2, Trace (M) =λ1+λ2

A window with a score R greater than a certain value is considered a “corner”.

CHANGE DETECTION:

This feature is used to perform damage detection in which user has to input two images and using the feature of change detection we can detect the change between two images taken at different time, doing so we can find the change that has occurred over that period of time. The predefined function of OpenCV library calculates the difference between two input test images by using subtract function which calculates the per-element difference between two arrays or array and a scalar.

RGB AND IHS TRANSFORMATION:

RGB to IHS conversion is an image enhancement technique. Image Enhancement can be defined as conversion of the image quality to a better and more understandable level for feature extraction or image interpretation, while radiometric correction is to reconstruct the physically calibrated value from the observed data.

The RGB color model is an additive color model in which red, green and blue are added together in various ways to reproduce a broad array of colors. The main purpose of the RGB color model is for the sensing, representing and display of images in electronic systems. The IHS color space is very important and attractive color model for image processing applications because it represents colors similarly how the human eye senses colors.

The IHS color model represents every color with three components: hue (h), saturation(s), intensity (I).

- The Hue component describe the color itself in the form of an angle between [0, 360] degrees. 0 degree means red, 120 means green and 240 means blue. 60 degrees is yellow, 300 degrees is magenta.
- The saturation component signals how much the color is polluted with white color. The range of the saturation component is [0, 1].
- The intensity range is between [0, 1] and 0 means black, 1 means whites.

Hue is more meaningful when saturation approaches 1 and less meaningful when saturation approaches 0 or when intensities approaches 0 or 1. Intensity also limits the saturation value.

RGB to IHS:

Transform any three bands of data from the RGB system into the IHS system in which the three component images represent intensity, hue, and saturation. The equations for making this transformation are described below.

- Read the RGB image.
- Represent the RGB image in the range [0,1].
- Find the IHS component by calculating

$$\Theta = \cos^{-1}\frac{\frac{1}{2}[(R-G)+(R-B)]}{(R-G)^2+(R-G)(G-B)^{\frac{1}{2}}}$$

$$H = \begin{cases} \Theta, & if\ B \le G \\ 360 - \Theta, & if\ B > G \end{cases}$$

$$S(Saturation) = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

$$I(Intensity) = \frac{1}{3}(R+G+B)$$

- Display the IHS image.

IHS to RGB:

After enhancing the saturation image, the IHS value is converted back into RGB images by inverse equations.

- Read IHS image.
- Represent the H component in (0, 360).
- If H is between 0 to 120

$$B = I * (1 - S)$$

$$R = I * (1 + (S * \cos(H))/\cos(60 - H)$$

$$G = 3 * I - (R + B)$$

- If H is between 120 to 240 (120<H<240)

$$R = I * (1 - S)$$

$$G = I * (1 + (S * \cos(H))/\cos(60 - H)$$

$$B = 3 * I - (R + B)$$

- If H is between 240 and 360 $(240 \leq H \leq 360)$

$$G = I * (1 - S)$$

$$B = I * (1 + (S * \cos(H))/\cos(60 - H)$$

$$R = 3 * I - (R + B)$$

- Convert the RGB component to (0, 255) range.
- Display the RGB image.

GLCM TEXTURE ANALYSIS:

Texture consists of texture primitives or texture elements, sometimes called texels. Texture can be described as fine, coarse, grained, and smooth, etc.

Such features are found in the tone and structure of a texture. Tone is based on pixel intensity properties in the Texel, while structure represents the spatial relationship between texels. If texels are small and tonal differences between texels are large a fine texture results. If texels are large and consist of several pixels, a coarse texture results.

GLCM texture analysis comes under Second order measures which measures the likelihood of observing a pair of grey value that occur a specified distance and direction or orientation in a part of image. Statistics considers relationship between groups of pixels in original image and is computed on matrices derived from original image.

A gray level co-occurrence matrix (GLCM) contains information about the positions of pixels having similar gray level values.

A co-occurrence matrix is a two-dimensional array, P, in which both the rows and the columns represent a set of possible image values.

A GLCM P_d [i, j] is defined by first specifying a displacement vector d= (x, y) and counting all pairs of pixels separated by having gray levels i and j. The GLCM is defined by:

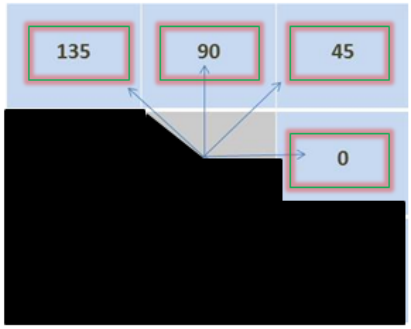
– Where nij is the number of occurrences of the pixel values (i, j) lying at distance d in the image.

– The co-occurrence matrix P_d has dimension n× n, where n is the number of gray levels in the image.

To find GLCM TEXTURE we must follow the algorithm whose steps are:

- The image at first is split into 3 three color band, Red(R Band), Green(G Band), Blue(B Band).

- The initial inputs required are the
  - The **Color Band** to be used : ***R, G, B***
  - The Relative **distance** between the pixel pair: ***d***
  - The Relative **orientation** / rotational angle : ***θ (0º, 45º, 90º, 135º)***



we consider θ as horizontal (0°), front diagonal (45°), vertical (90°) and back diagonal (135°)

- Count all pairs of pixels in which the first pixel has a value ***i***, and its matching pair displaced from the first pixel by **d** has a value of ***j***, in direction of angle ***θ,*** for the selected color band .

- This count is entered in the **ith** row and **jth** column of the matrix P_d **[i, j]**

- Note that P_d **[i, j]** is not symmetric, since the number of pairs of pixels having gray levels **[i, j]** does not necessarily equal the number of pixel pairs having gray levels **[i, j].**

To understand the framework of GLCM we need to understand the concept that how GLCM works-

GLCM texture considers the relation between two pixels at a time, called the **reference** and the **neighbor** pixel. Each pixel within the window becomes the reference pixel in turn, starting in the upper left corner and proceeding to the lower right. Pixels along the right edge have no right hand neighbor, so they are not used for this count.

• Gray level co-occurrence matrices capture properties of a texture but they are not directly useful for further analysis, such as the comparison of two textures.

• Numeric features are computed from the co-occurrence matrix that can be used to represent the texture more compactly.

From this matrix, generate a list of features:

–Contrast,

–Dissimilarity

–Homogeneity

–Angular second moment,

–Energy

–Entropy

Contrast:

Contrast is a measure of the local variations present in an image. Also called sum of squares variance.

$$\sum_{i,j=0}^{N-1} P_{i,j}\,(i-j)^2$$

If there is a large amount of variation in an image the **P [i, j]** ‟s will be concentrated away from the main diagonal and contrast will be high.

Dissimilarity:

In the Contrast measure, weights increase exponentially (0, 1, 4, 9, etc.) as one moves away from the diagonal. However, in the dissimilarity measure weights increase linearly (0, 1, 2,3 etc.)

$$\sum_{i,j=0}^{N-1} P_{i,j} \quad |i-j|$$

Homogeneity (also called the **"Inverse Difference Moment"**):

Dissimilarity and Contrast result in larger numbers for more contrast windows. If weights decrease away from the diagonal, the result will be larger for windows with little contrast.

Homogeneity weights values by the inverse of the Contrast weight, with weights decreasing exponentially away from the diagonal:

$$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{(i-j)^2}$$

Angular Second Moment (ASM):

ASM uses $P_{i,j}$ as a weight for itself. High values of ASM occur when the window is very orderly.

$$\mu = \sum_{i,j=0}^{N-1} P_{i,j}^2$$

Energy (Uniformity):

This feature returns the sum of squared elements in the GLCM. Range = [0 1] : Energy is 1 for a constant image. The property Energy is also known as uniformity, uniformity of energy, and square root angular second moment.

$$\text{Energy} = \sqrt{ASM}$$

Correlation:

Returns a measure of how correlated a pixel is to its neighbour over the whole image. Range = [-1 1] : Correlation is 1 or -1 for a perfectly positively or negatively correlated image. Correlation is NaN for a constant image. This feature measures the joint probability occurrence of the specified pixel pairs.

$$\sum_{i,j=0}^{N-1} P_{i,j} \frac{(i-\mu)(j-\mu)}{\sigma^2}$$

DATABASE PART:

PostgreSQL 9.3 has been used as the backbone for the backend of our project. PostgreSQL is the most powerful and suitable software for the combination of Python and PostGIS, which is a spatial database extender for PostgreSQL, object-relational database.

Authentication-

The authentication part comes at high necessity where the question of confidentiality arrives. This section is required for both normal users and Administrators. The registered users can authenticate themselves and login to the application only when he can proceed with his tasks.

The user authenticated if is a normal user, gets access to only manage his projects, while if he is an administrator, he gets an allowance to manage the users as well as all the projects. Only the administrator has the authority to delete projects.

In case, a user forgets his password, he has to request the administrator for a new password and ones the administrator changes the password, the user gets notified about it. This is done to maintain a two way security to prevent unauthorized modifications.

Project Management-

All users and administrators get the authority to create new projects and manage existing ones. The Administrator can manage all the projects while the others can only manage projects created by him or permitted access by the administrators.

Both users and administrators can grant another user the privilege to insert and select data to and from the database as well as revoke earlier privileges.

User Management-

The Administration is the only one who gets the authority to manage users. He can add new users and manage existing users as in,

- Provide the user validity, change his password,
- Add his to a group, encrypt his password,
- Allow his permissions to manage database and users, for both new and existing users. He can also delete an existing user or revoke his rights.

Database Creation-

For the database to get accessed, we first need to create the database, store data (image pixels) into it and access and visualize it accordingly. All the extracted properties and values of the point cloud will be stored in the database and will be retrieved from the same for the visualization of the data and values. Values which are stored in the database are:

1. X,Y,Z Co-ordinate -

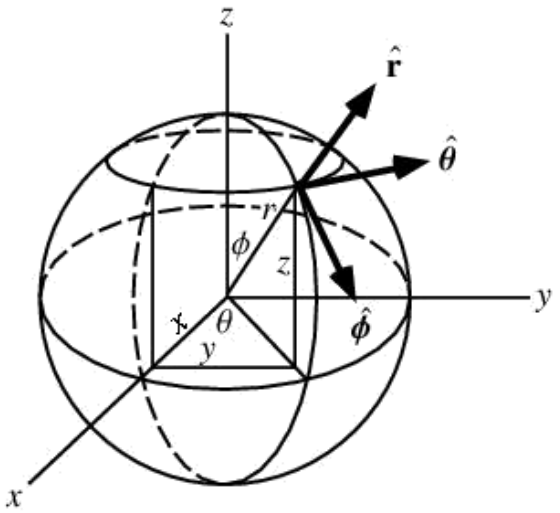
The X, Y, Z co-ordinate of the points of the point cloud will be extracted and stored directly from the ASCII text file of point cloud.

2. Band Pixels-

A raster dataset contains one or more layers called bands. For example, a color image has three bands (red, green, and blue) while a digital elevation model (DEM) has one band (holding elevation values), and a multispectral image may have many bands.

Each raster band contains the actual cell values, as well as some key properties, such as:

- Statistics: minimum, maximum, and mean
- cell values Histogram of cell values
- A value attribute table (optional additional attribute information about various cell values)
- Default color map for raster display (optional)



3. Theta-

Spherical coordinates, also called spherical polar coordinates (Walton 1967, Arfken 1985), are a system of curvilinear coordinates that are natural for describing positions on a sphere or spheroid. Define to be the azimuthal angle in the -plane from the x-axis with (denoted when referred to as the longitude).

$$\theta = \tan^{-1}\frac{y}{x}$$

4. R (Radius):

R to be distance (radius) from a point to the origin.

$$r = \sqrt{x^2 + y^2 + z^2}$$

5. Depth:

Depth be the polar angle (also known as the zenith angle and colatitudes, with $\phi = 90 - \delta$ where δ is the latitude from the positive z-axis with

$0 \leq \phi \leq \pi$, $\phi = \cos^{-1}\left(\frac{z}{r}\right)$

Where $r \in (0, \infty)$, $\theta \epsilon (0, 2\pi)$ and $\phi \in (0, \pi)$ and the inverse tangent must be suitably defined to take the correct quadrant of (x, y) into account.

In terms of Cartesian co-ordinates,

X=$r \cos\theta \sin\ \phi$

Y= $r \sin\theta \sin\ \phi$

Z= $r \cos\ \phi$

So the unit vectors are,

$$\hat{r} = \frac{\frac{dr}{d\gamma}}{\left|\frac{dr}{d\gamma}\right|}$$

$$= \begin{bmatrix} cos\theta sin\emptyset \\ sin\theta sin\emptyset \\ cos\emptyset \end{bmatrix}$$

$$\hat{\theta} = \frac{\frac{dr}{d\theta}}{\left|\frac{dr}{d\theta}\right|}$$

$$= \begin{bmatrix} -sin\theta \\ cos\theta \\ 0 \end{bmatrix}$$

$$\widehat{\emptyset} = \frac{\frac{dr}{d\emptyset}}{\left|\frac{dr}{d\emptyset}\right|}$$

$$= \begin{bmatrix} cos\theta sin\emptyset \\ sin\theta sin\emptyset \\ -sin\emptyset \end{bmatrix}$$

6. Statistics:

ST_SummaryStats — Returns summary stats consisting of count, sum, mean, stddev, min, max for a given raster band of a raster or raster coverage. Band 1 is assumed if no band is specified where

- count gives the number of pixels(n).
- Min is minimum pixel value.
- Max is maximum pixel value.
- mean distance = sum (distances) / n
- standard deviation = sqrt ($\sum (X_i - X_m)^2/n$)

Where

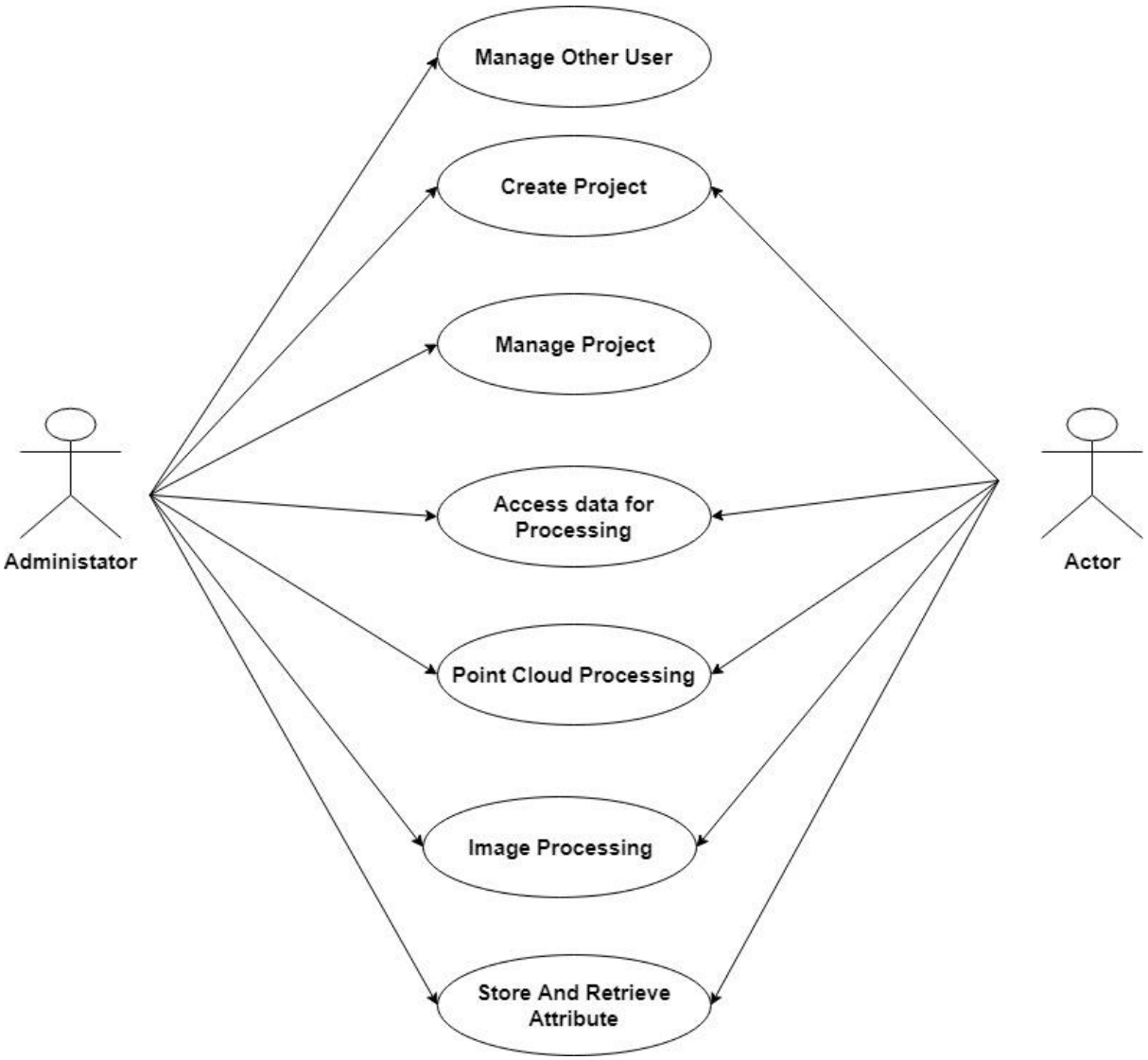
n: number of all distances

X_i: distance from point I to its next neighbour

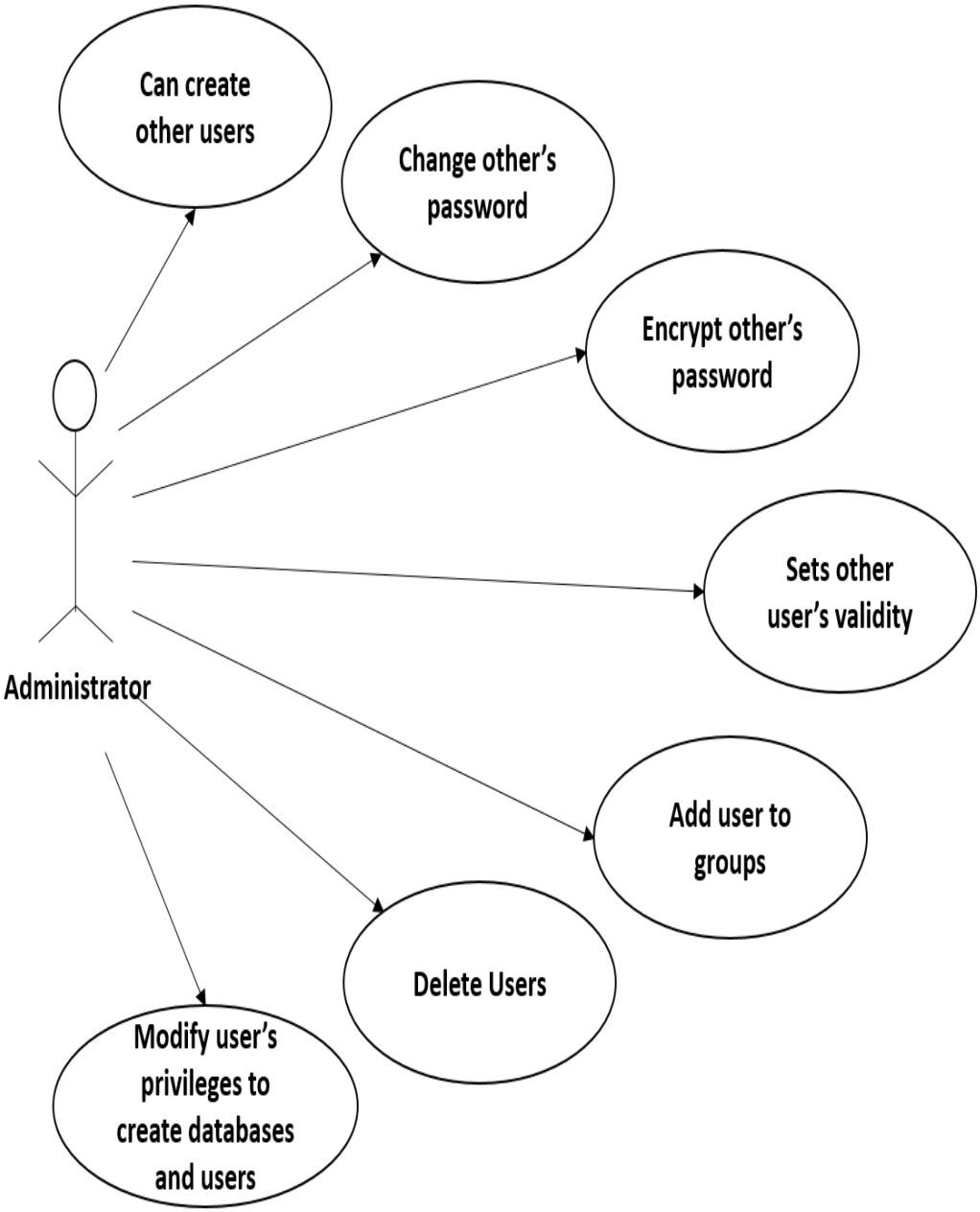
X_m: mean distance.

PROJECT DESIGN DETAILS:

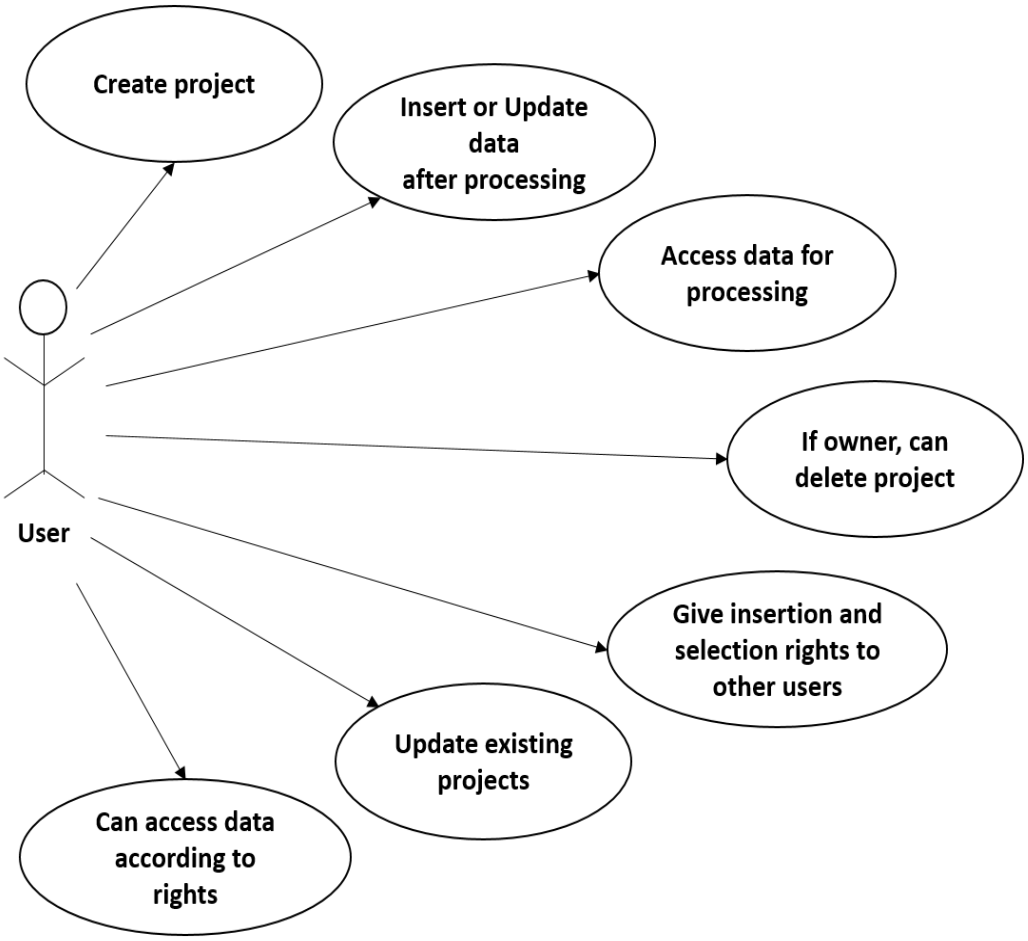
Overall Design:



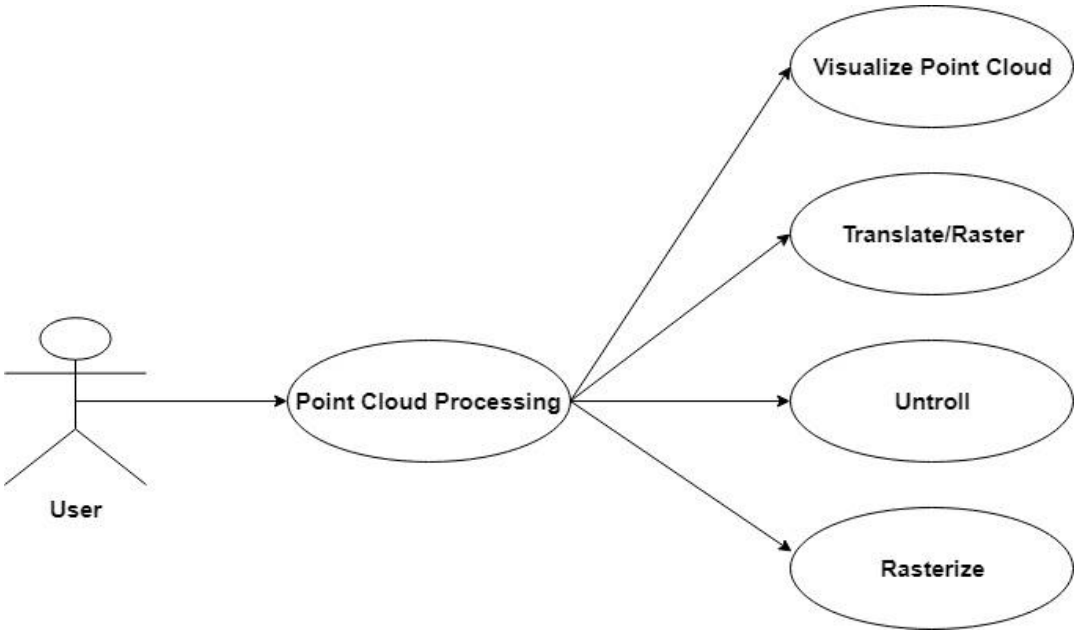
User Management Use Case Diagram



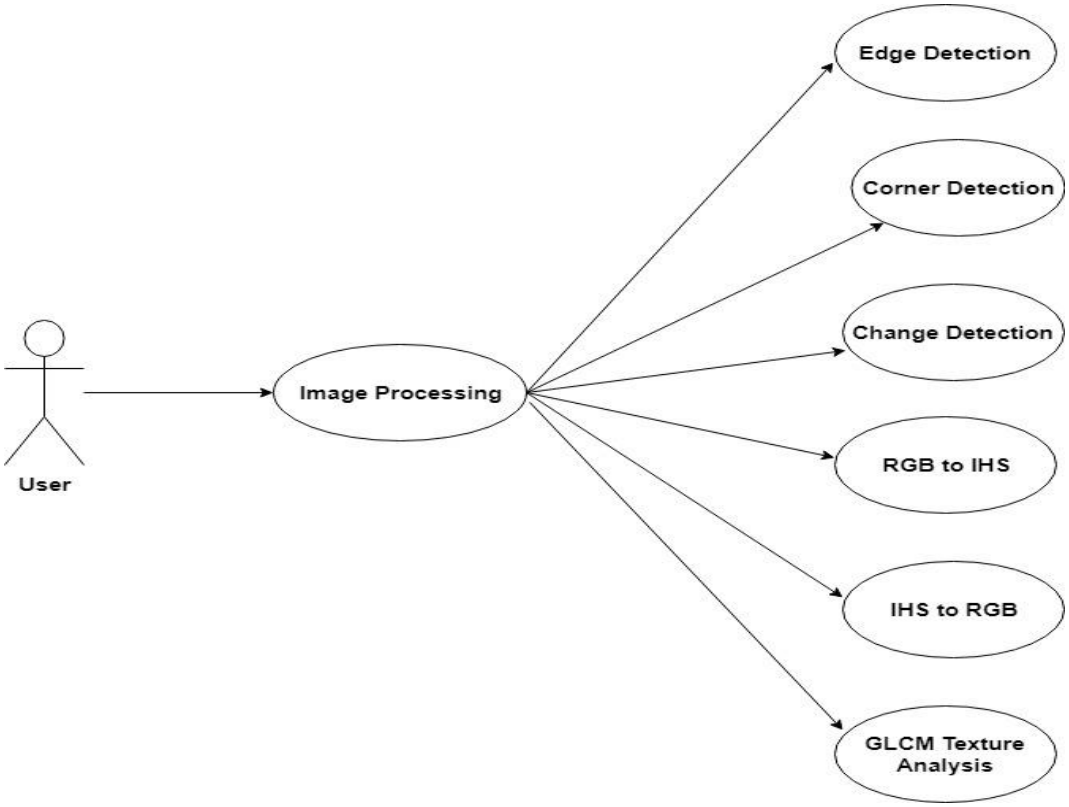
Project Management Use Case Diagram



Use Case Diagram for Point Cloud Processing:

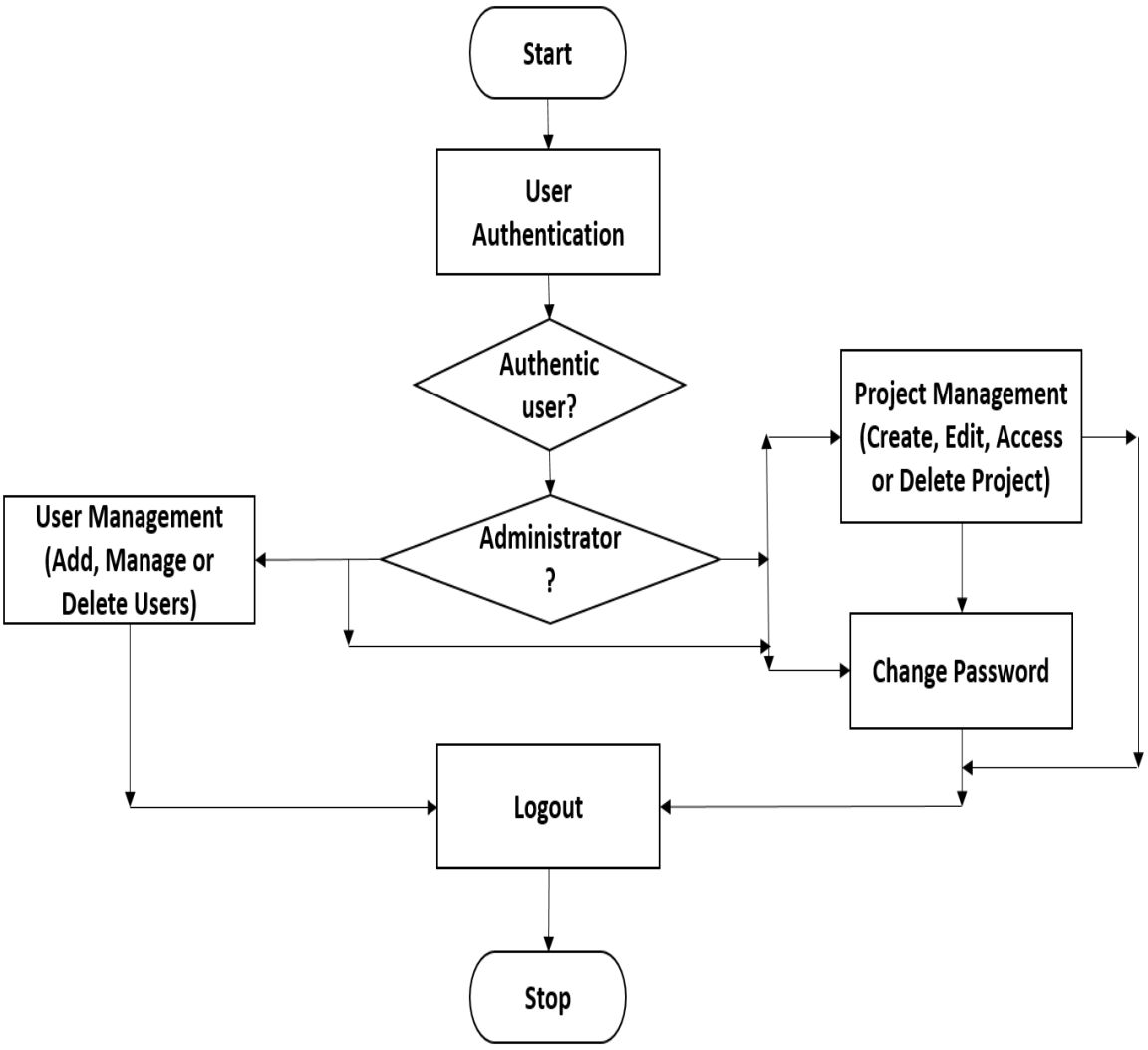


Use Case Diagram for Image Processing:

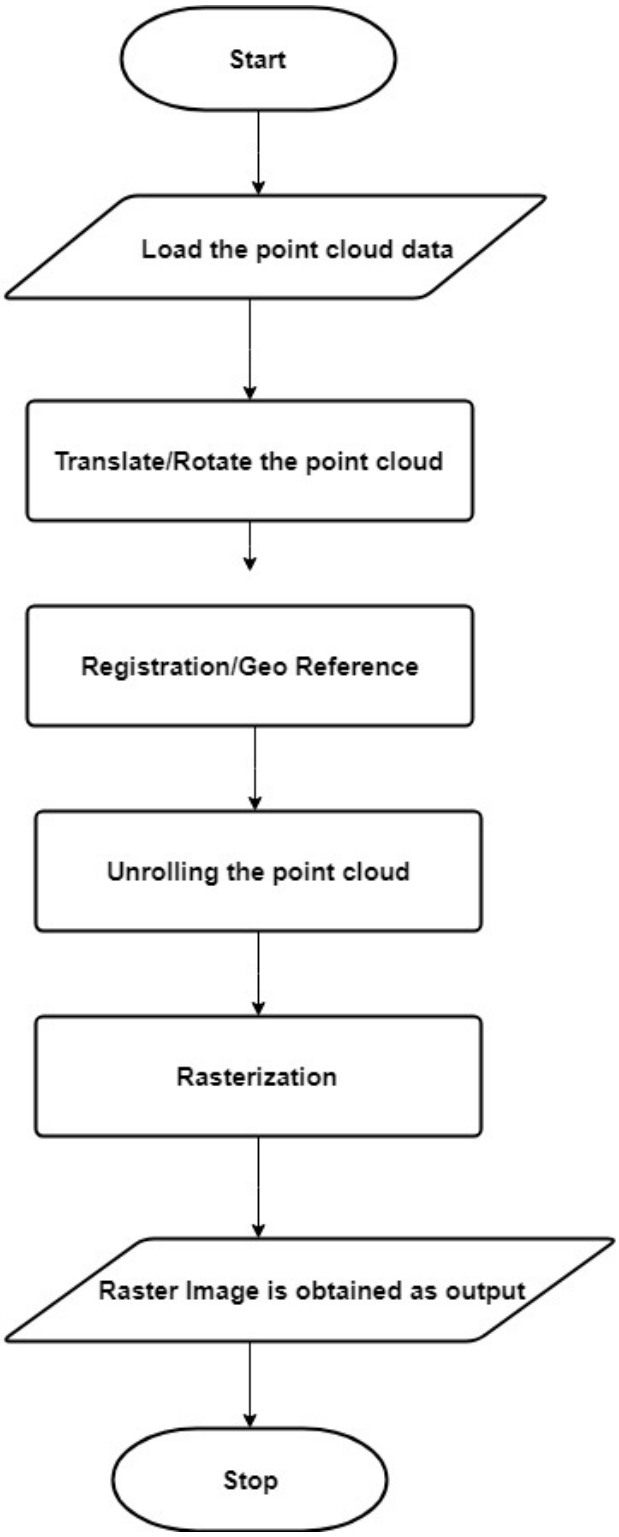


Flow Chart

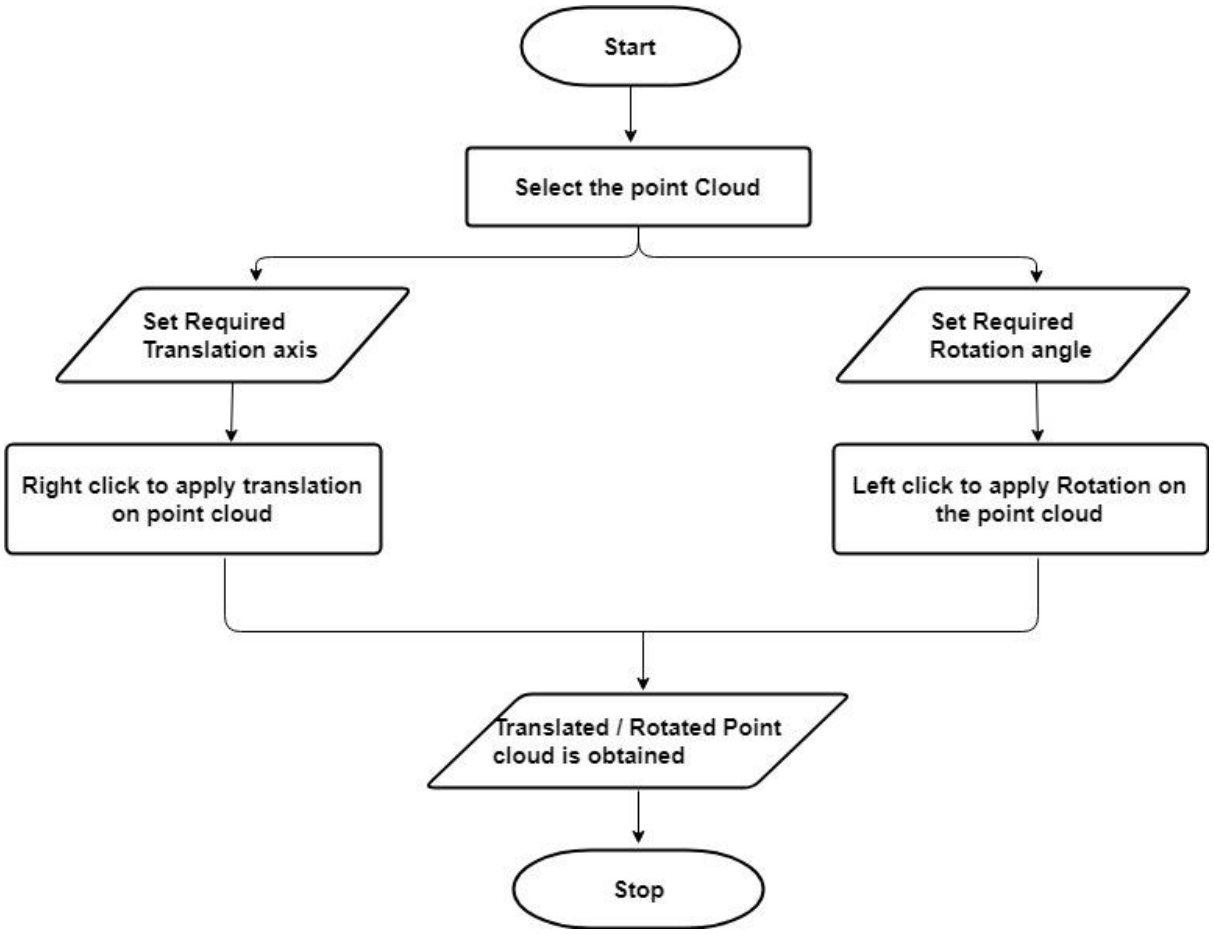
Data Access Layer Flowchart



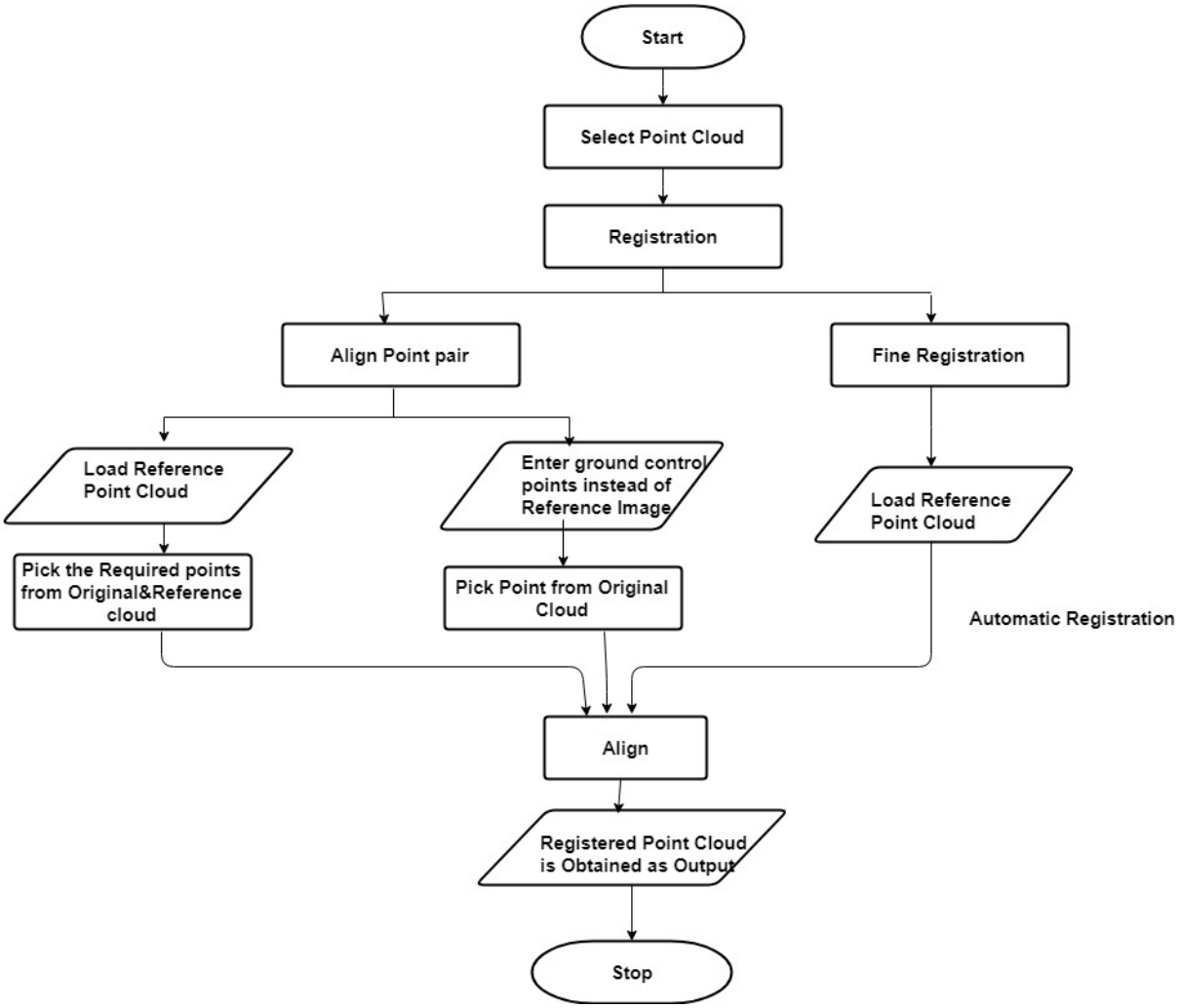
Flow Chart for Point Cloud Processing:



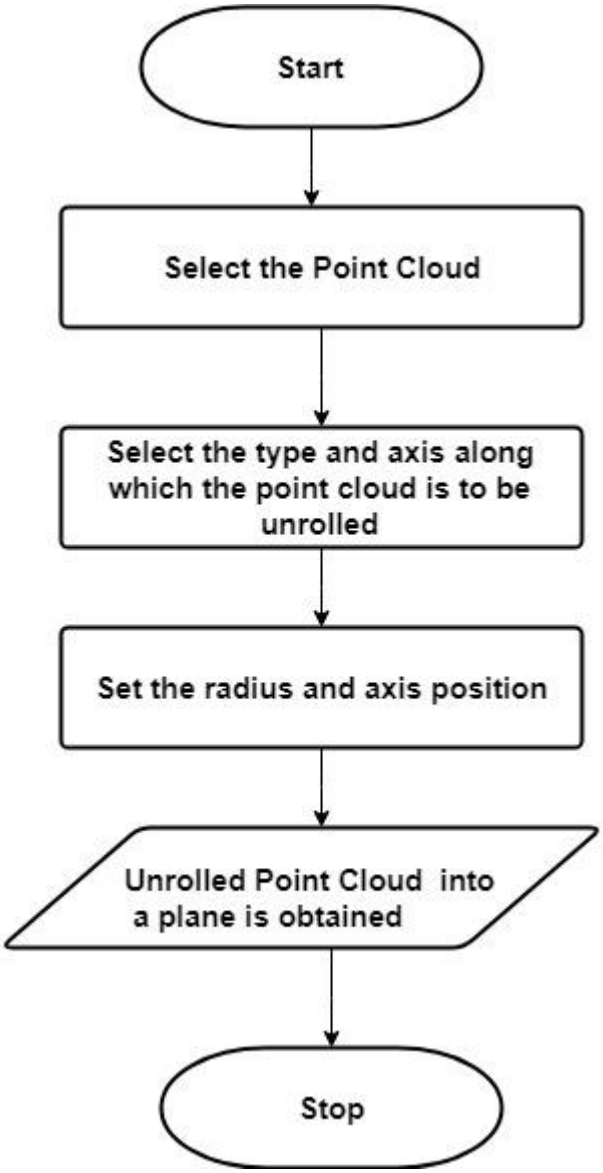
Flow Chart For Translate/Rotate:



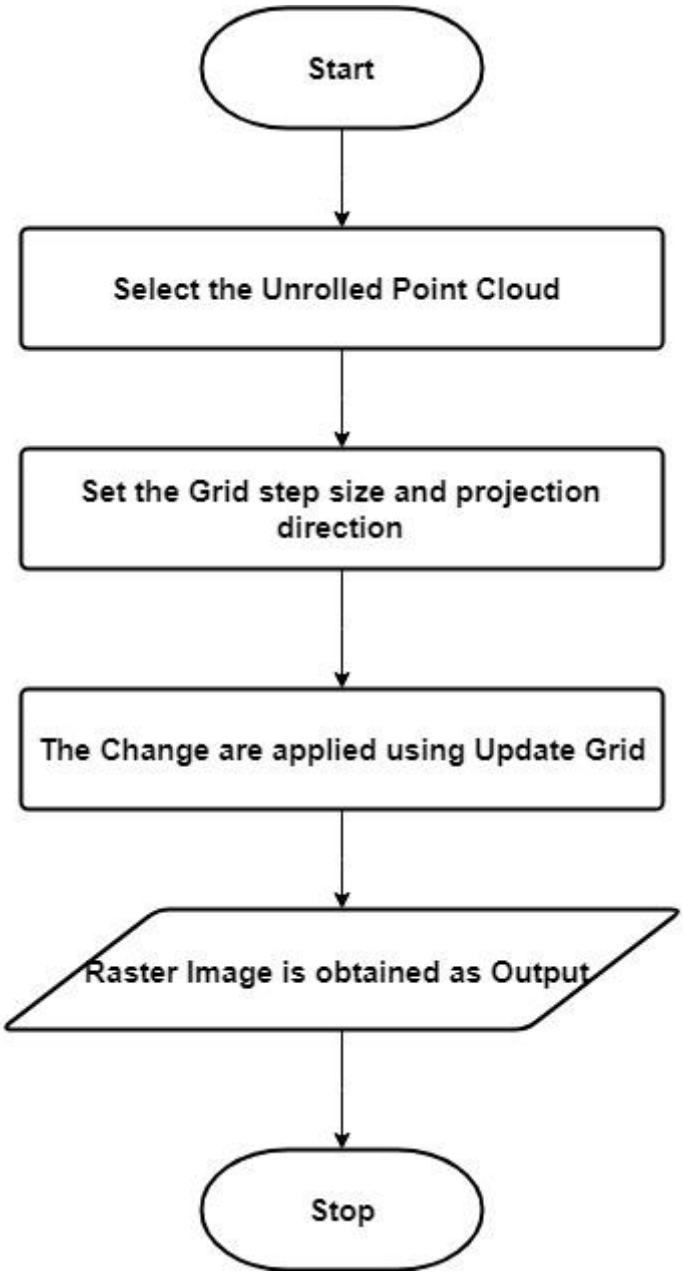
Flow Chart for Registration:



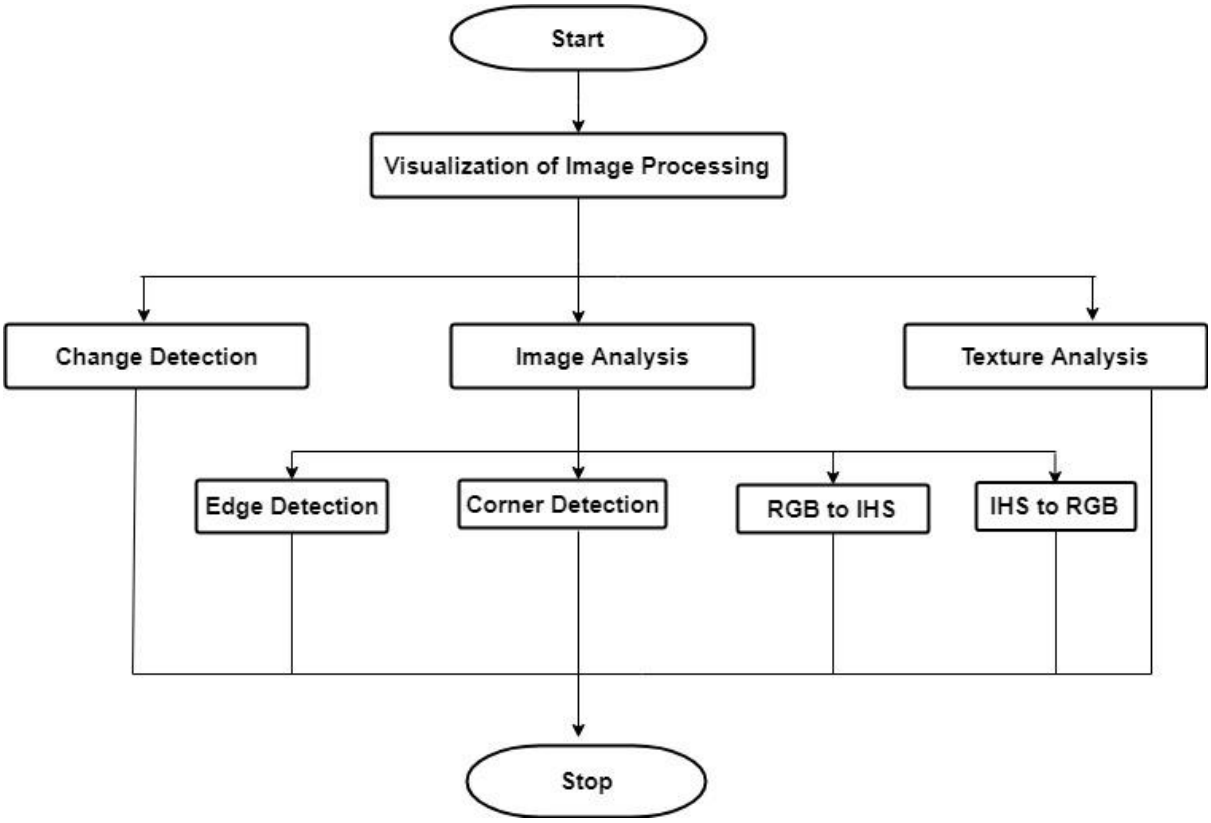
Flow Chart for Unroll:



Flow Chart for Rasterization:

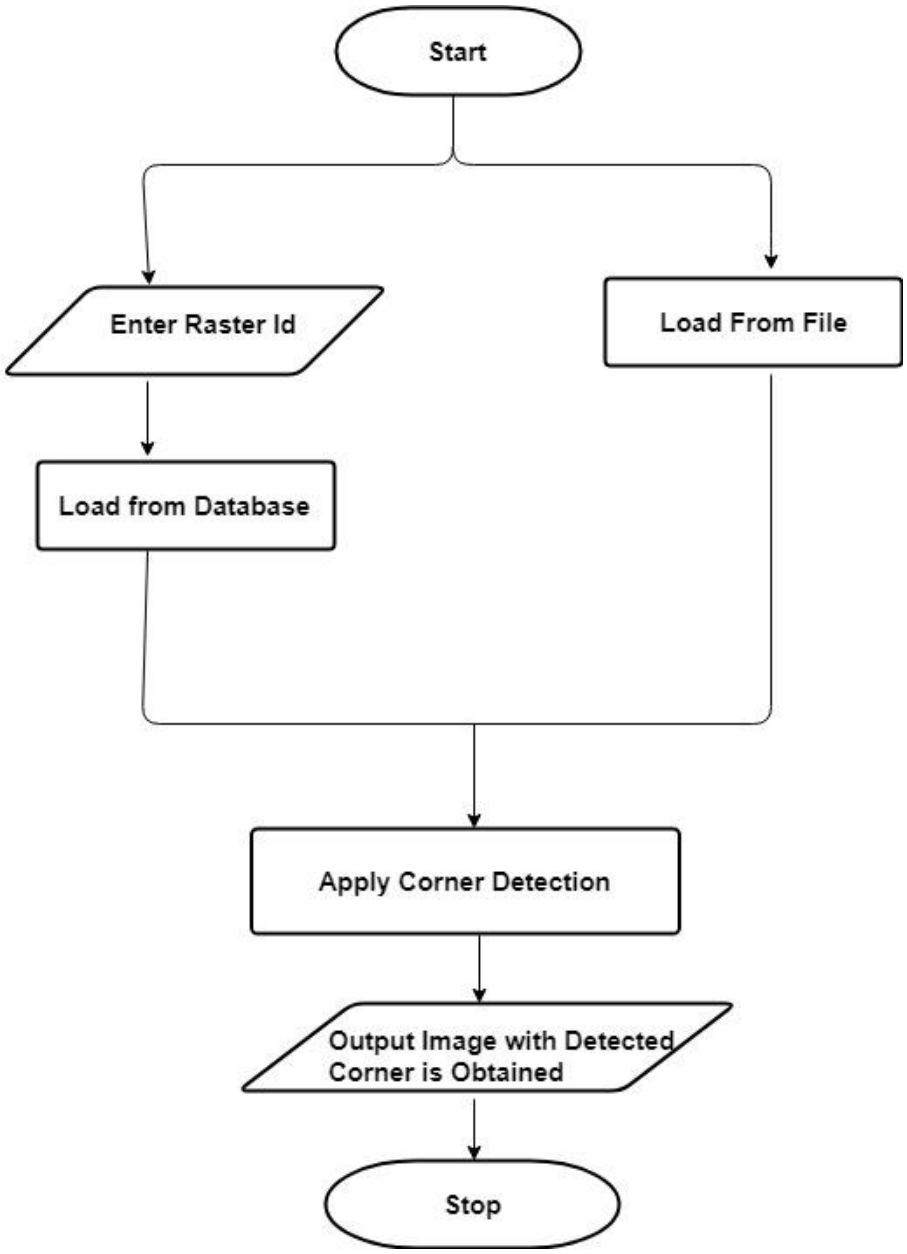


Flow Chart for Image Processing:

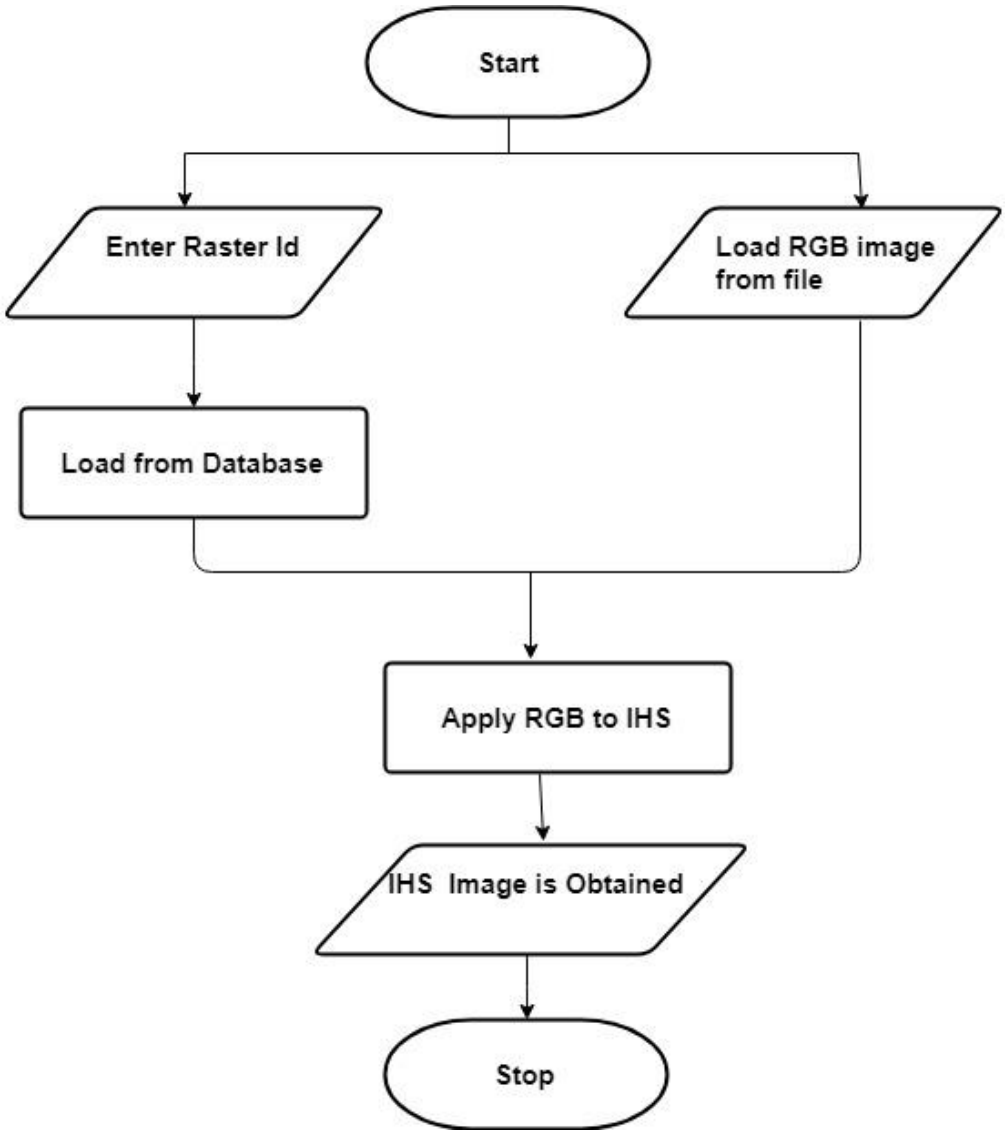


Flow Chart for Edge Detection:

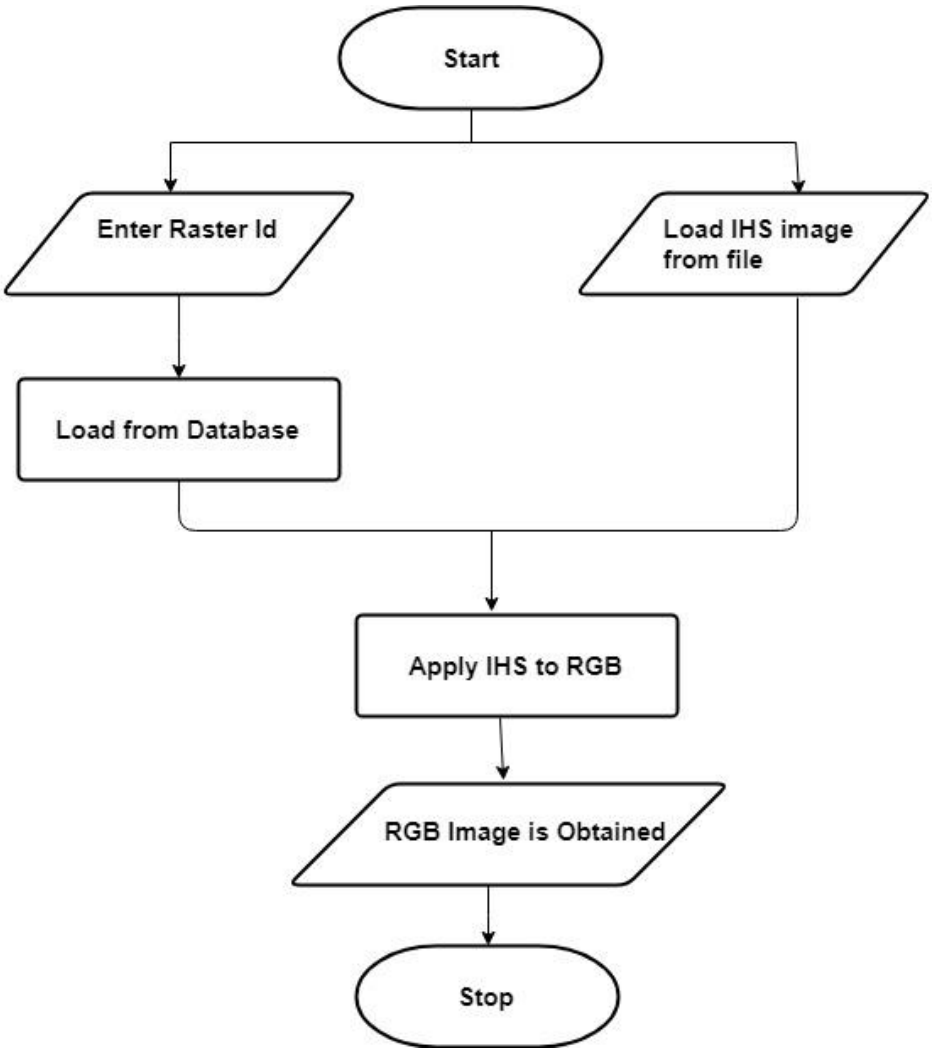
Flow Chart for Corner Extraction:



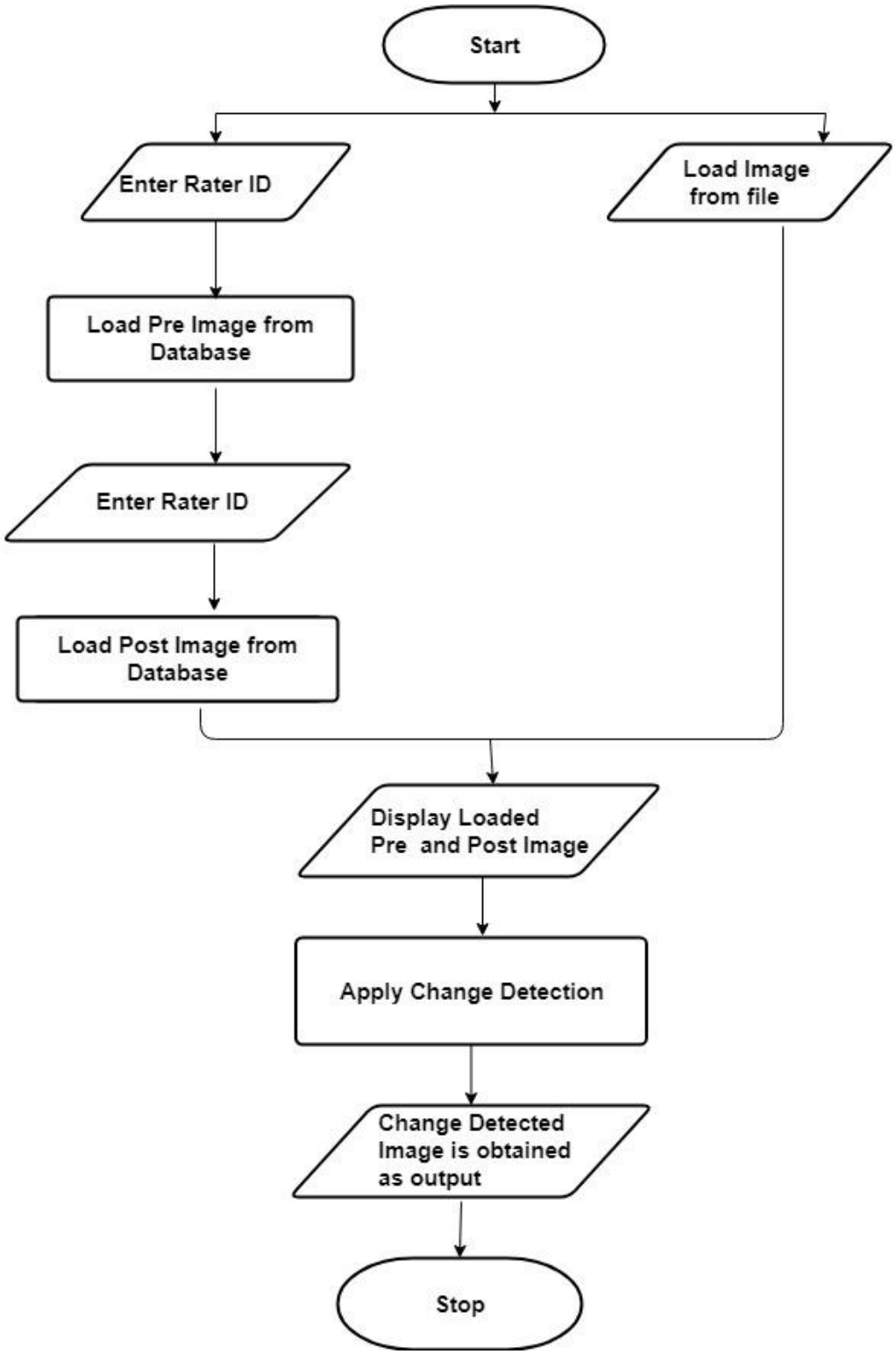
Flow Chart for RGB to IHS:



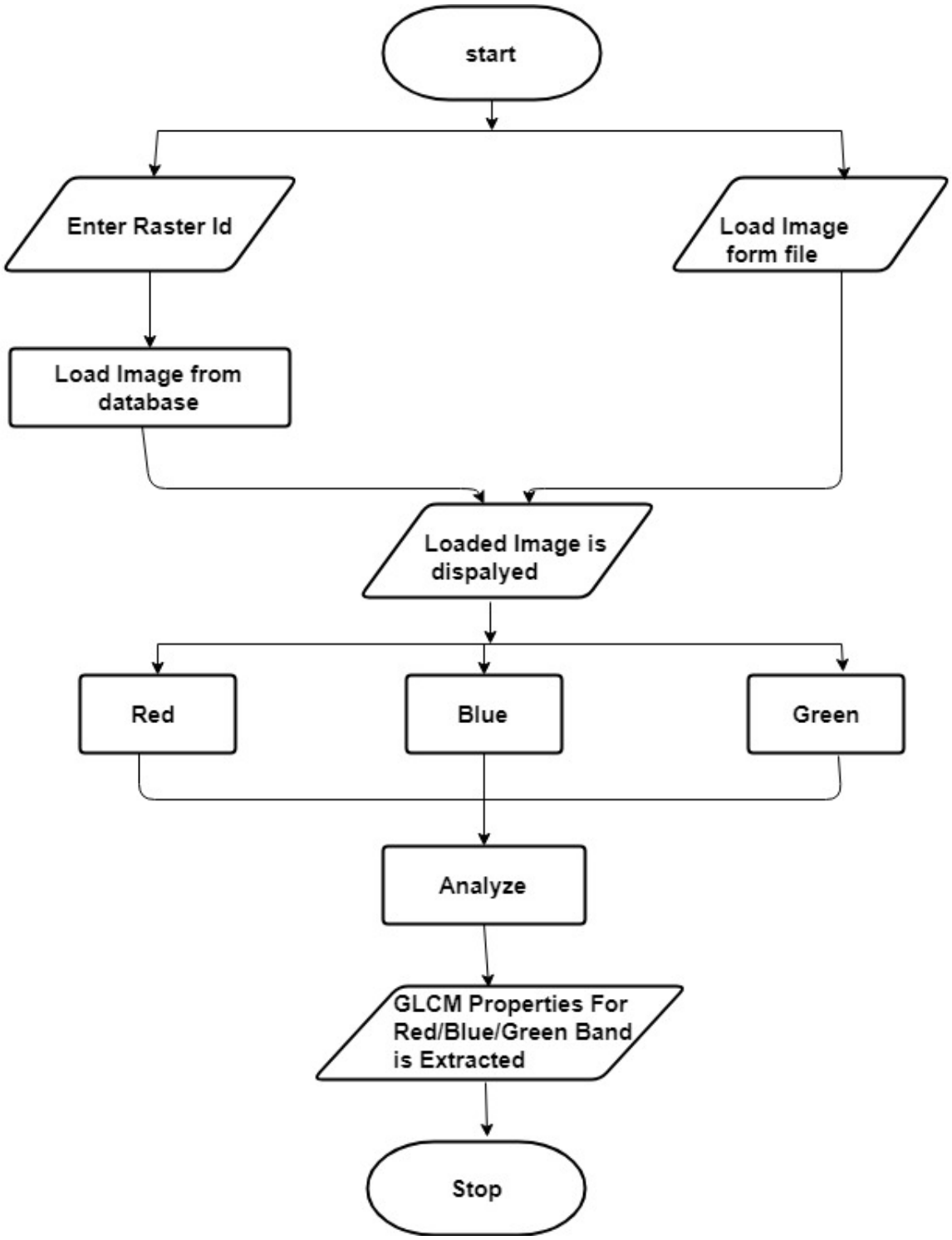
Flow Chart for IHS to RGB:



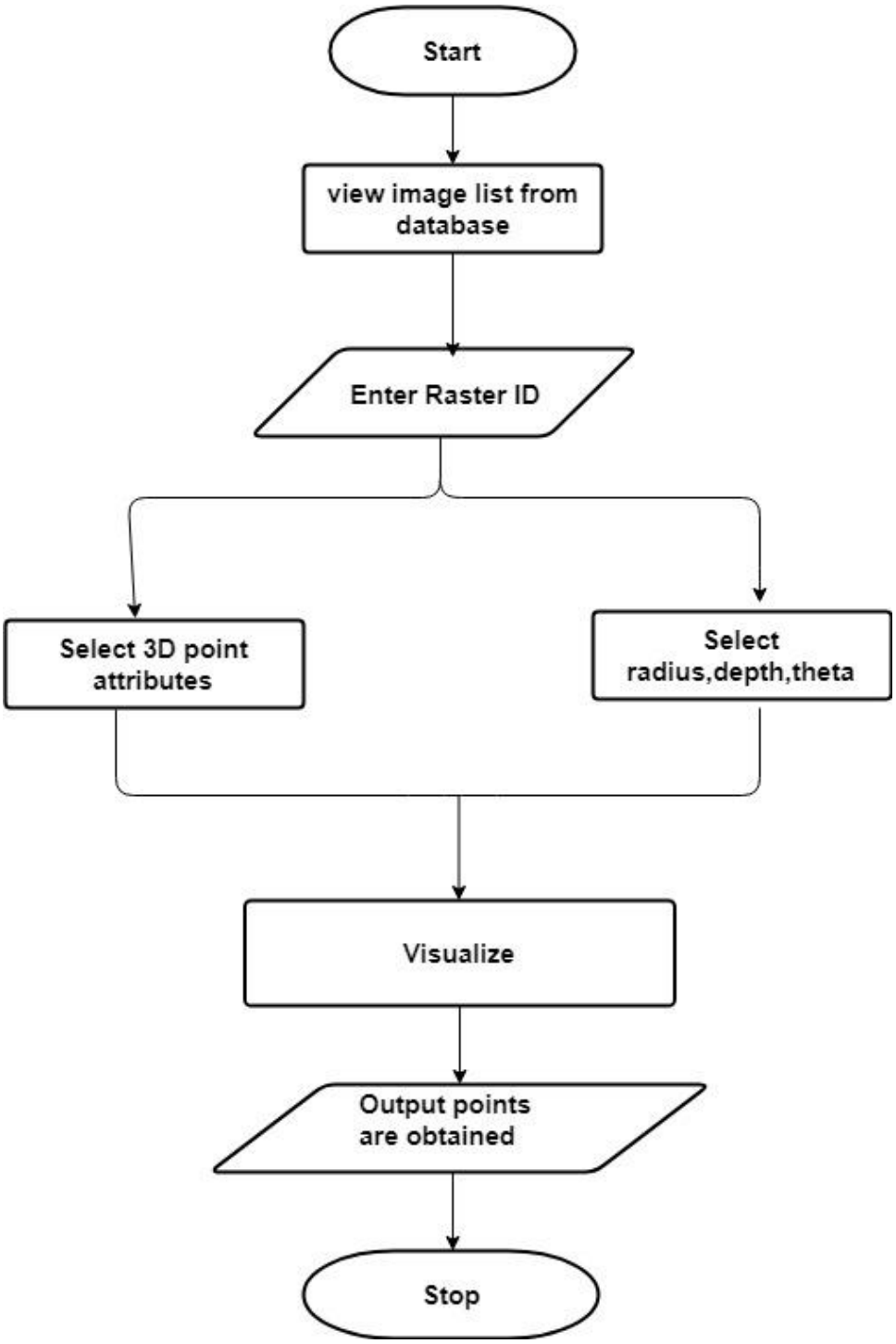
Flow Chart for Change Detection:



Flow Chart for Texture Analysis:



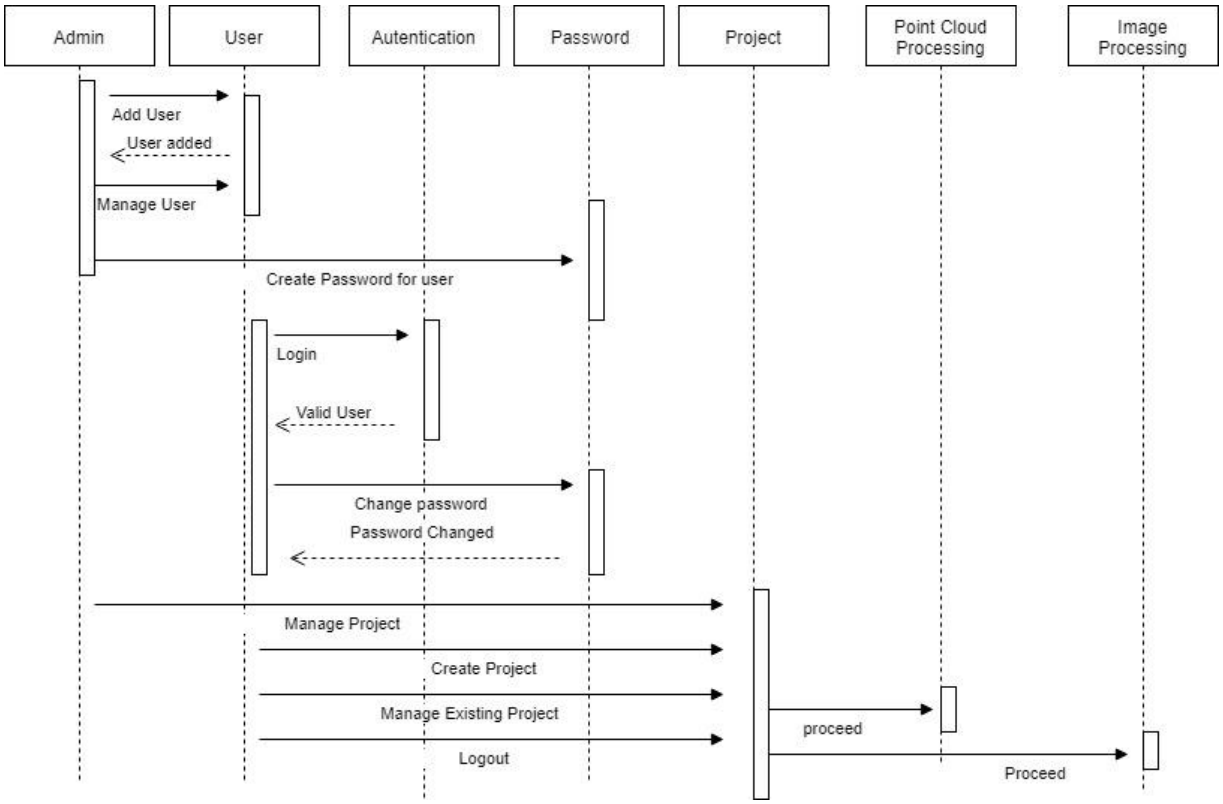
Flow Chart for Point Attributes:



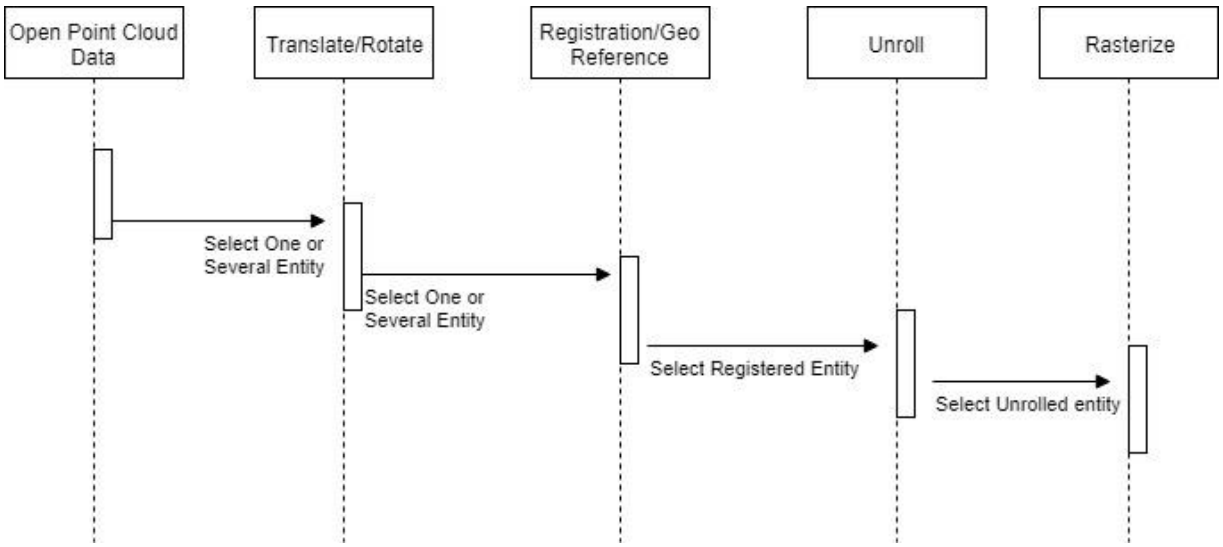
SEQUENCE DIAGRAM:

Sequence diagram shows object interaction arranged in time sequence. In debicts the objects and classes involved in the scenario and sequence of the messages exchanged between the objects needed to carry out the functionalities of the scenario.

Overall Sequence Diagram:



Sequence Diagram for Point Cloud Processing:



TESTING:

Testing is the process of evaluating a system or its components with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Unit Testing:

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

All the modules have been individually tested and all the test cases have passed successfully. And no defaults occur.

Integration Testing:

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

All the individual modules are integrated and tested together. All the test cases have passed successfully. And no defaults occur.

Functional Testing:

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

Functions are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. Each functionalities of this application are tested by providing give set of inputs in order to know the actual behavior of the applications and thereafter comparing with the expected results as per the given specifications.

Functional testing mainly involves:

- Verifying user interfaces
- Verifying end to end work flows

Verifying correct data storing in the database. All the requirements are tested successfully.

System Testing:

System testing is a level of software testing where complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. Usually software is only one element of a larger computer based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer based system.

The created standalone application is tested among various systems with different windows Operating system and for both 32 nit and 64 bit systems.

Acceptance Testing:

Acceptance testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

CONCLUSION:

In recent days, due to high increase in pollution, high rate of damages in our heritages are noticed which is even rising at an alarming rate. These heritages are the most precious assets a country has and needs severe maintenance to keep them sustained for coming eras. The maintenance is not possible for humans to check and do on a regular basis, manually. This gave rise to the concept for our application, “DHAROHAR”.

There were many applications already available in the market to do so but in order to perform all the required processing completely, the need was to utilize multiple applications. The transfer of data from one application to another, causes loss of data and a lot of inconveniences. Our application ran over all those issues and looked forward towards one standalone application, to perform all the processing in one platform.

“DHAROHAR” is capable of performing multiple functions as:

- Image Analysis
- Change Detection
- Texture Analysis
- Point Attributes
- Registration
- Unroll
- Rasterization
- Database Storage and Retrieval

The application is thus a one stop solution to all the requirements for the regular survey and maintenance.

FORESEEABLE ENHANCEMENTS

1. Still more languages can be added to the application for localization.
2. Detecting the cracks using laser can be implemented in the application.
3. All the output of image Processing can also be stored in database for future use.

SCREEN SHORTS:

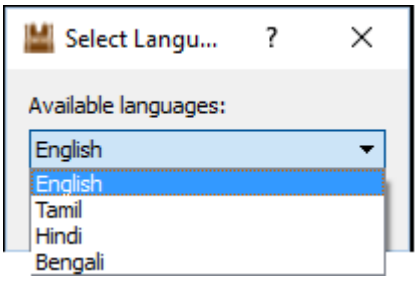


Fig 1: Language selection

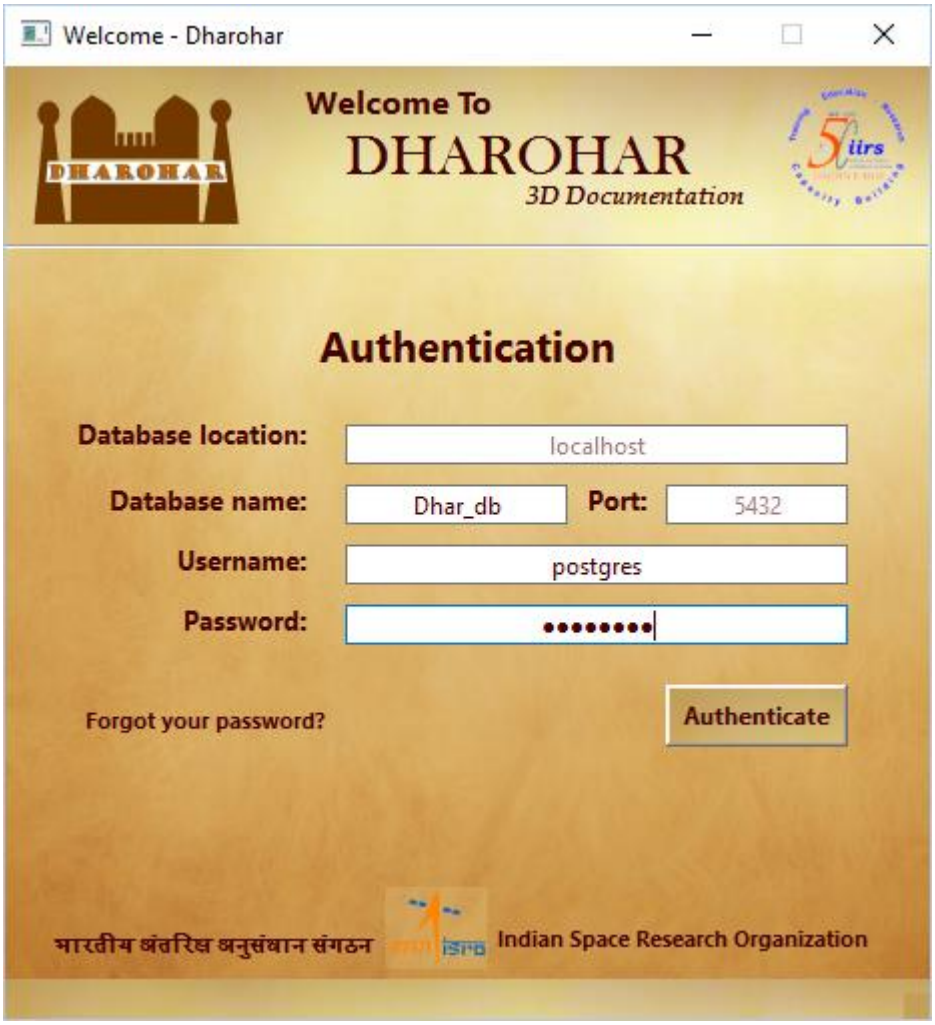


Fig 2: Authenticating For logging in to the application

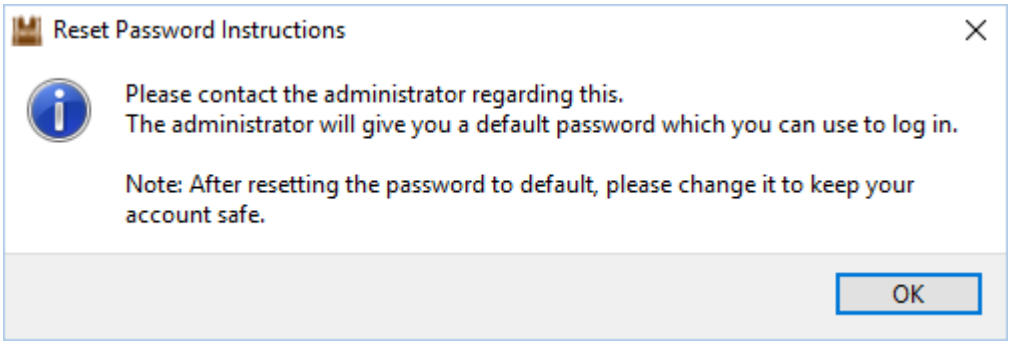


Fig 3: Reset password message appears while user logging for the first time

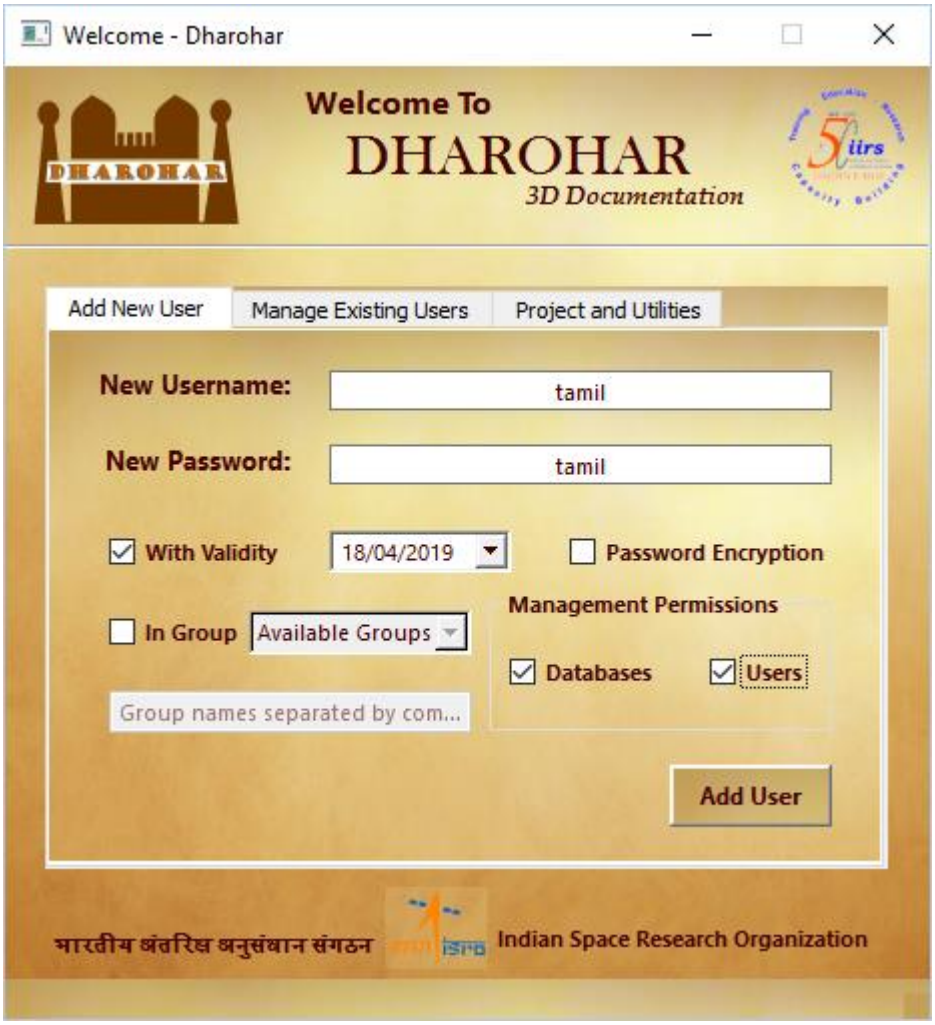


Fig 4: Administrator adding new user

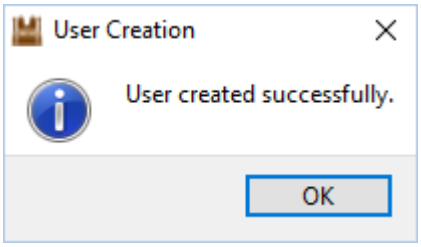


Fig 5: After successfully adding an user

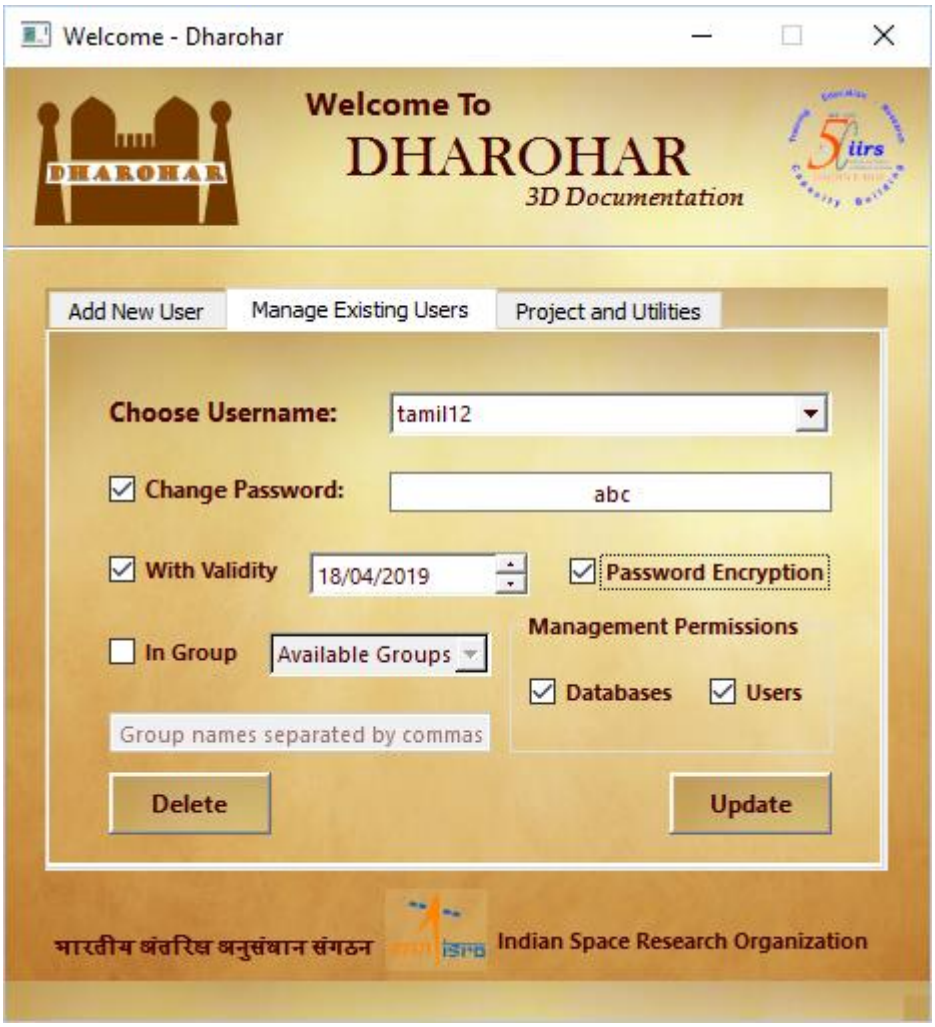


Fig 6: Administrator managing existing users

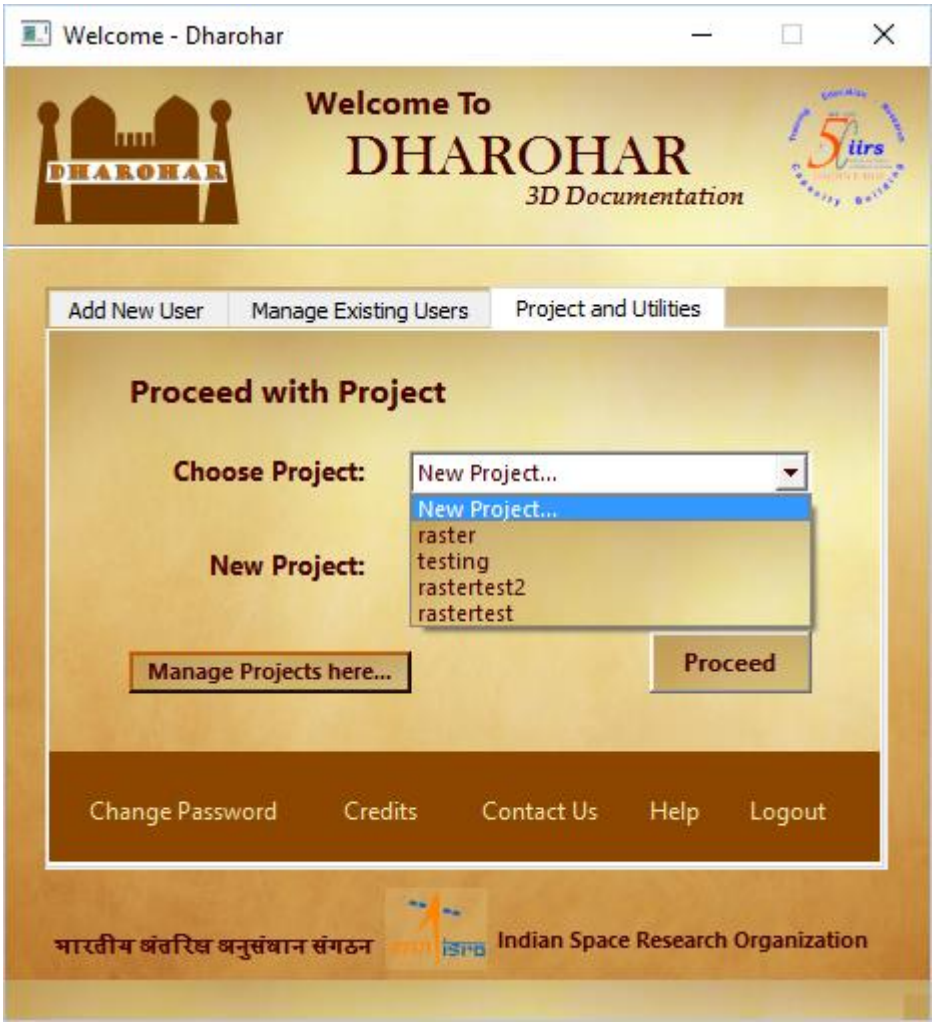


Fig 7: Choosing an existing project


Welcome - Dharohar
Welcome To
DHAROHAR
3D Documentation
Back
Manage Project 'raster'
Select User:
Usernames
Usernames
tamil
tamil1
jerald
abc
tamil12
ezhil
Privileges
Delete
Grant
Revoke
भारतीय अंतरिक्ष अनुसंधान संगठन
Indian Space Research Organization

Fig 8: Managing existing project

Fig 9: Change password

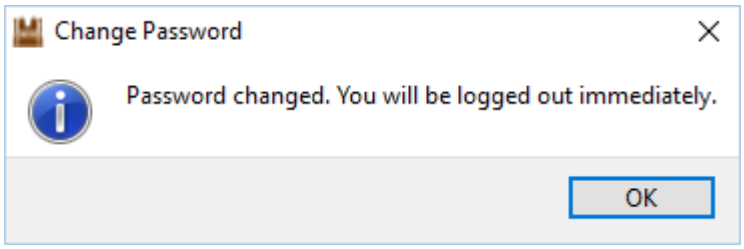


Fig 10: Password changed successfully

Fig 11: Contact us

Fig 12: Credits

Fig 13: Creating new project

Fig 14: Project Visualization – Refresh and Load Image list disabled on opening new project

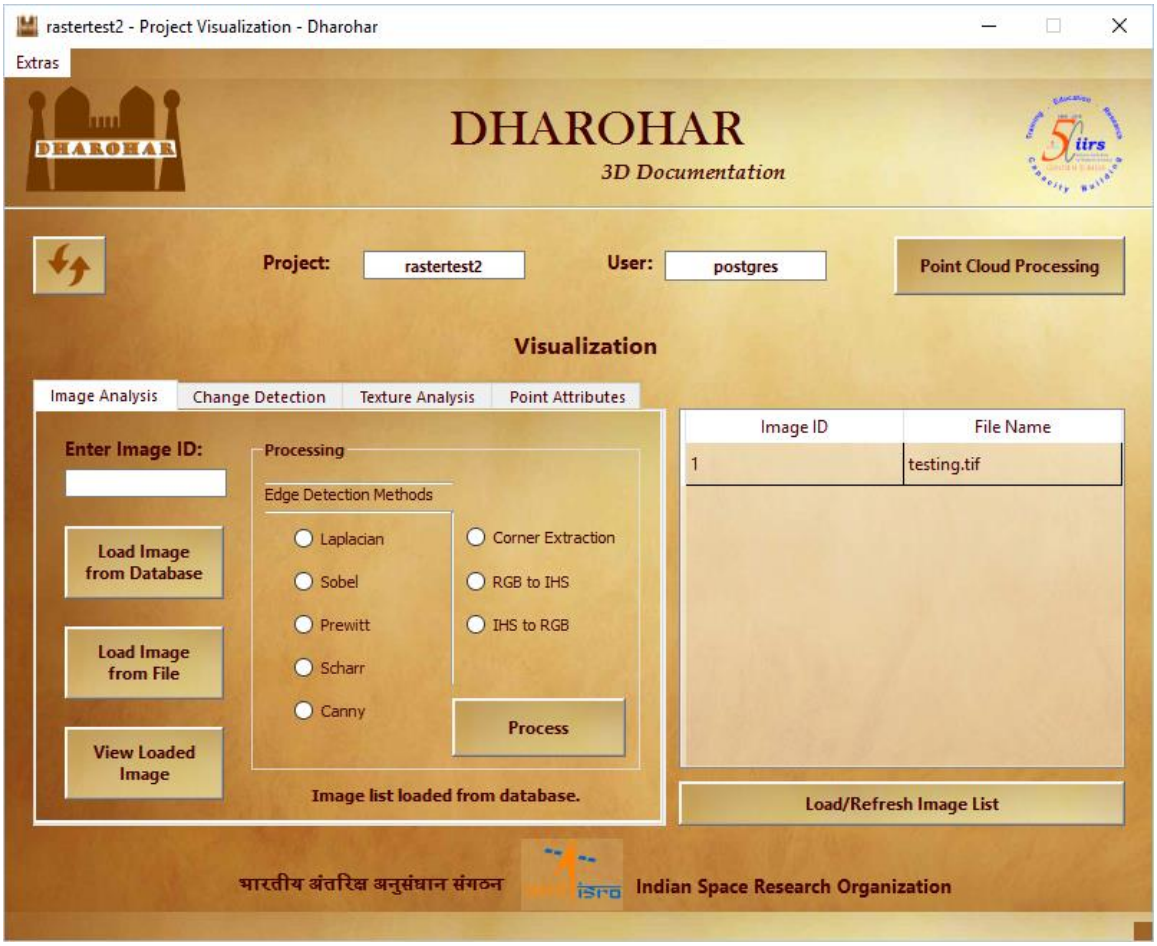


Fig 15: Project Visualization – Refresh and Load Image list Enabled on opening existing

Fig 16: Point cloud Processing

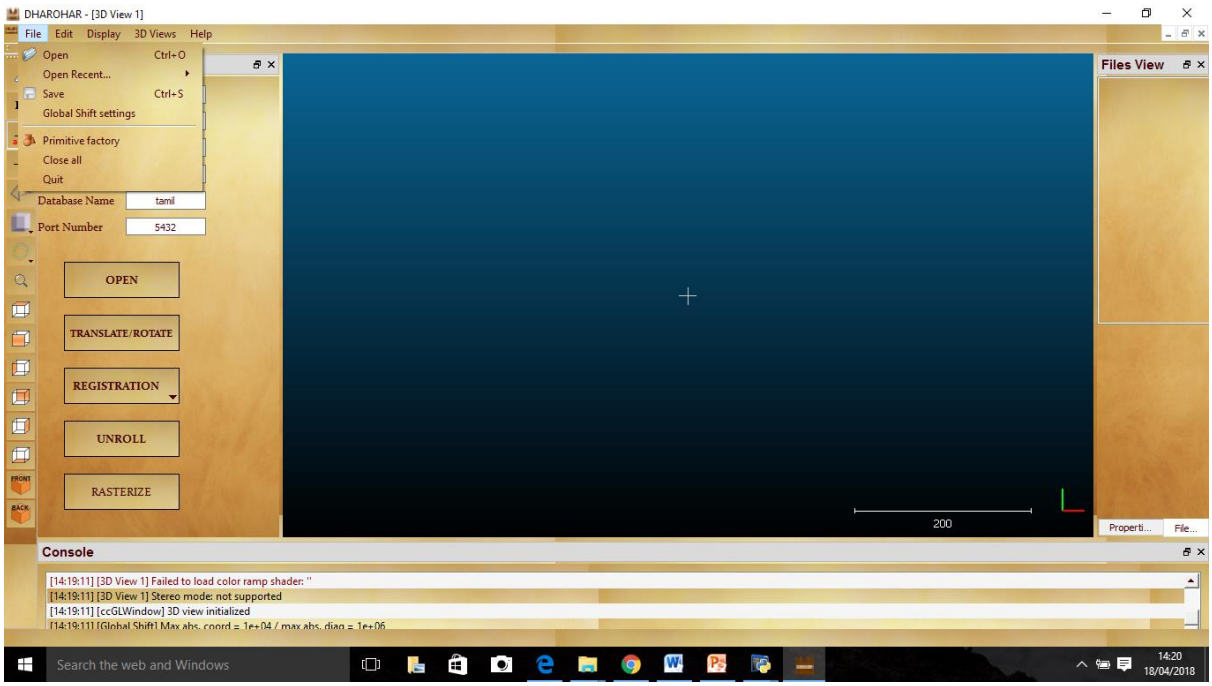


Fig 17: File Menu

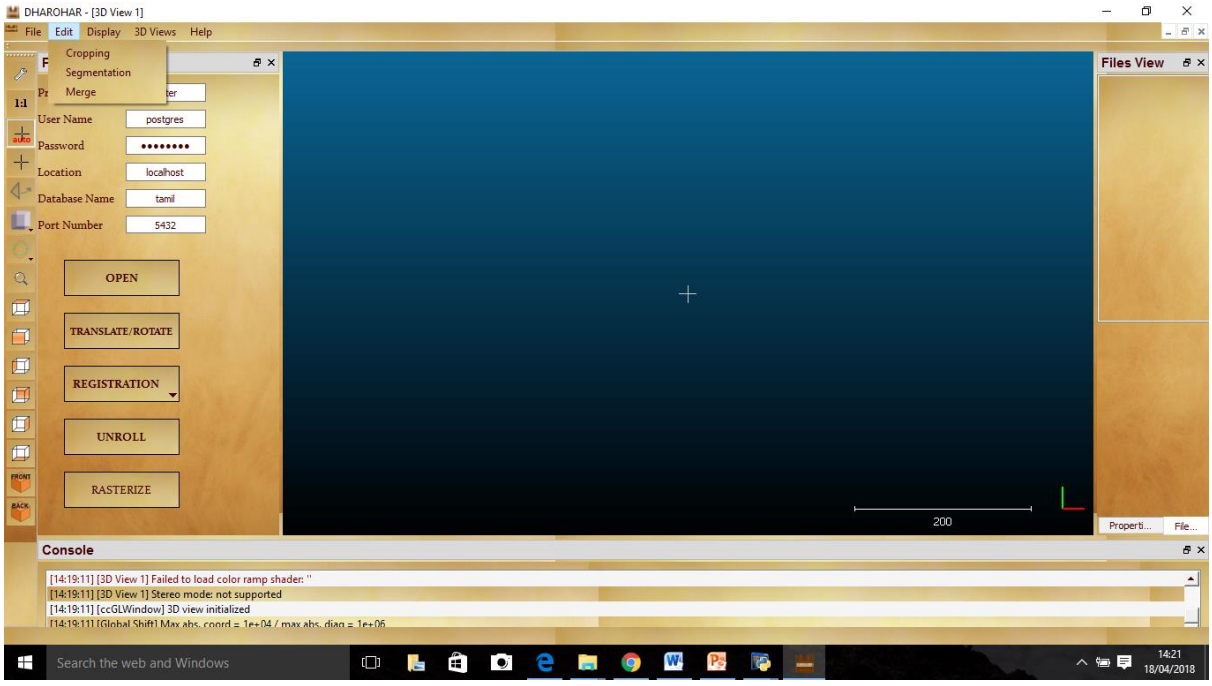


Fig 18: Edit Menu

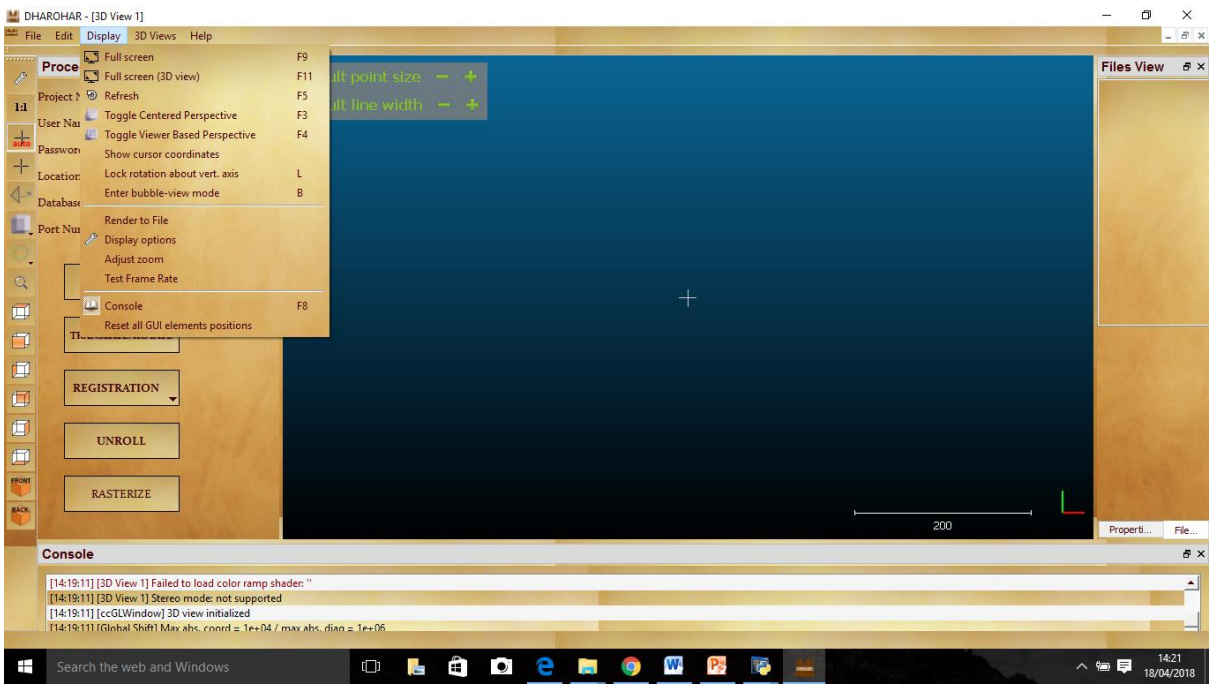


Fig 19: Edit menu

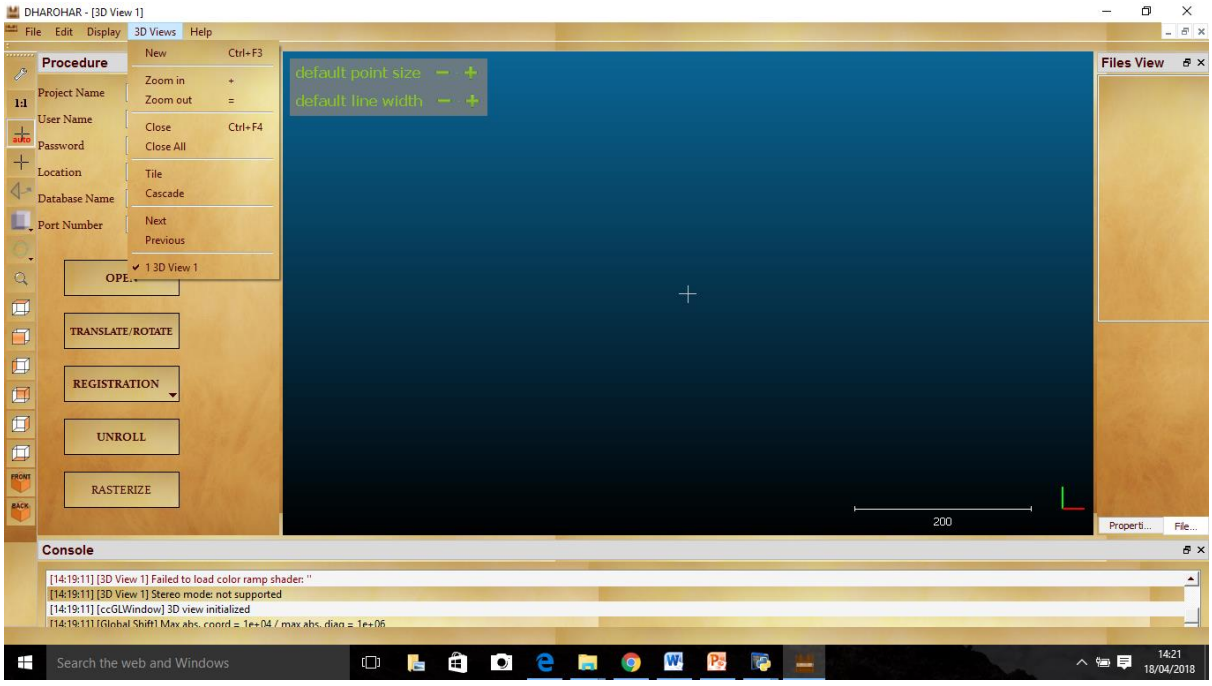


Fig 20: 3D view menu

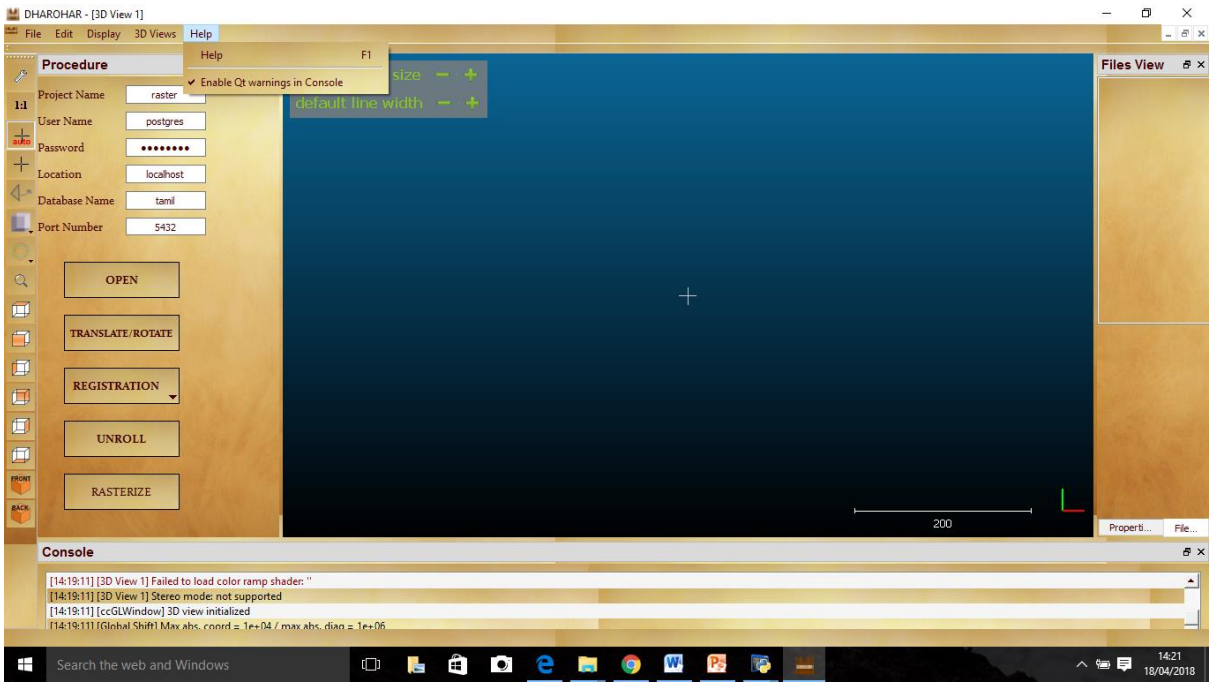


Fig 21: Help menu

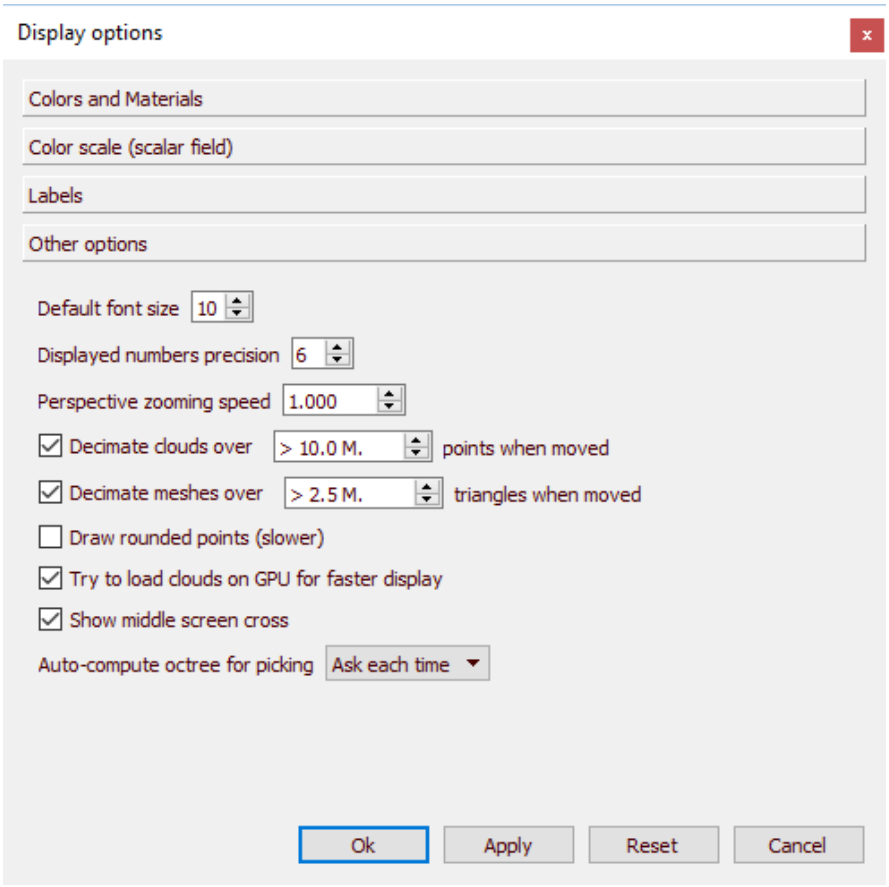


Fig 22: Display options

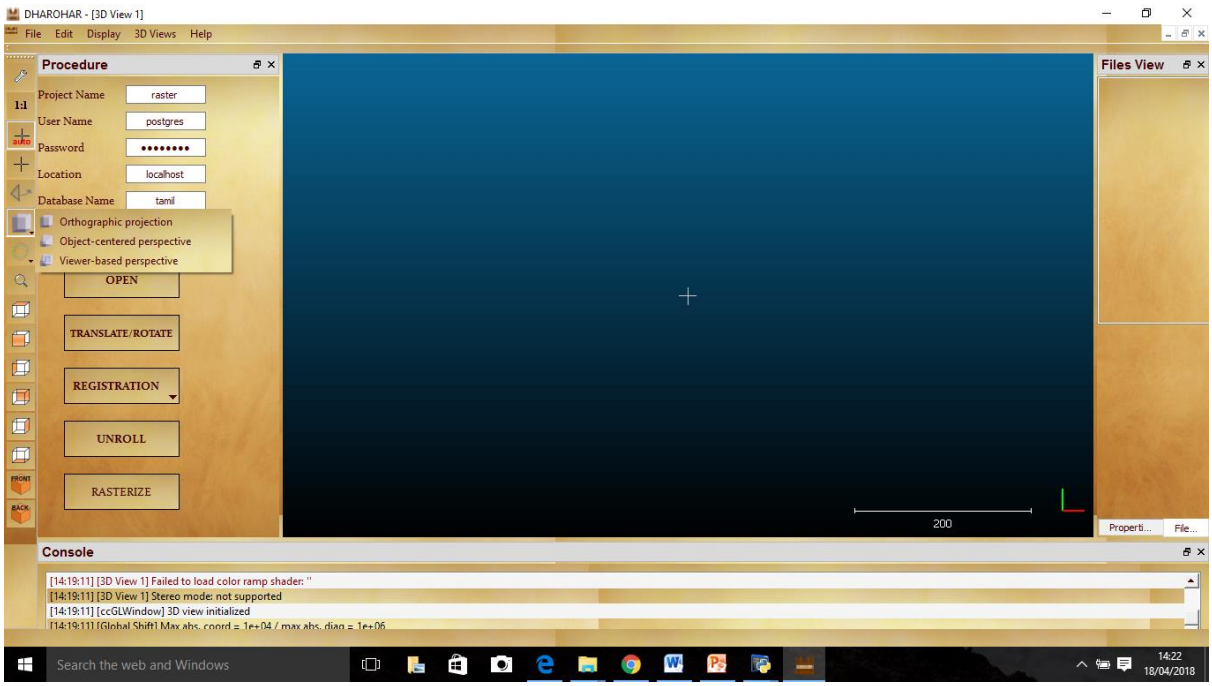


Fig 23: Projections

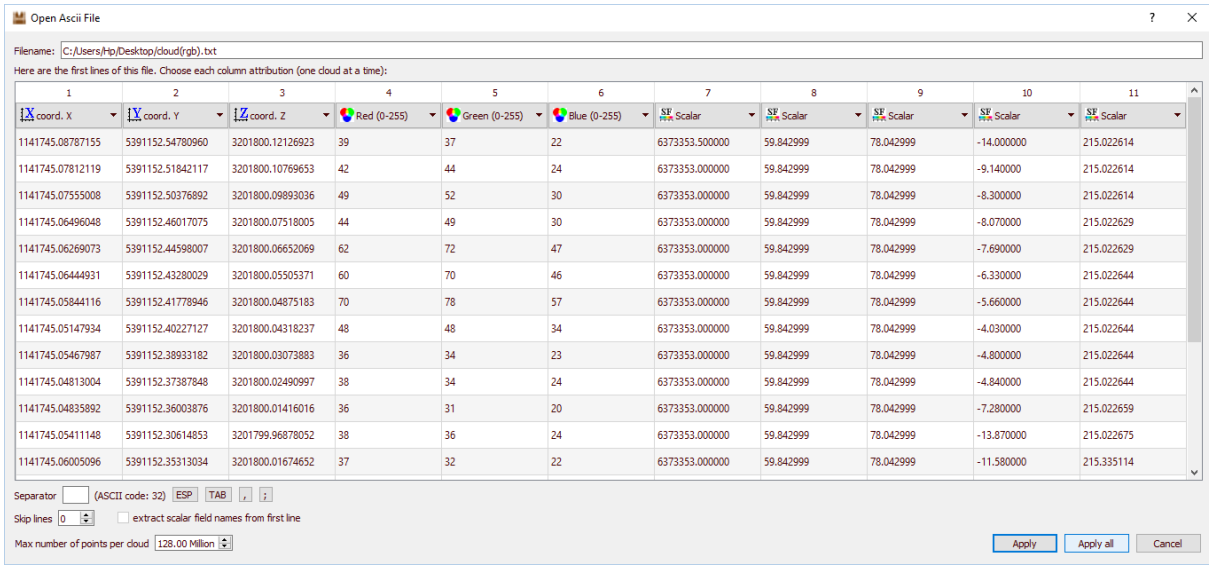


Fig 24: Open Point cloud

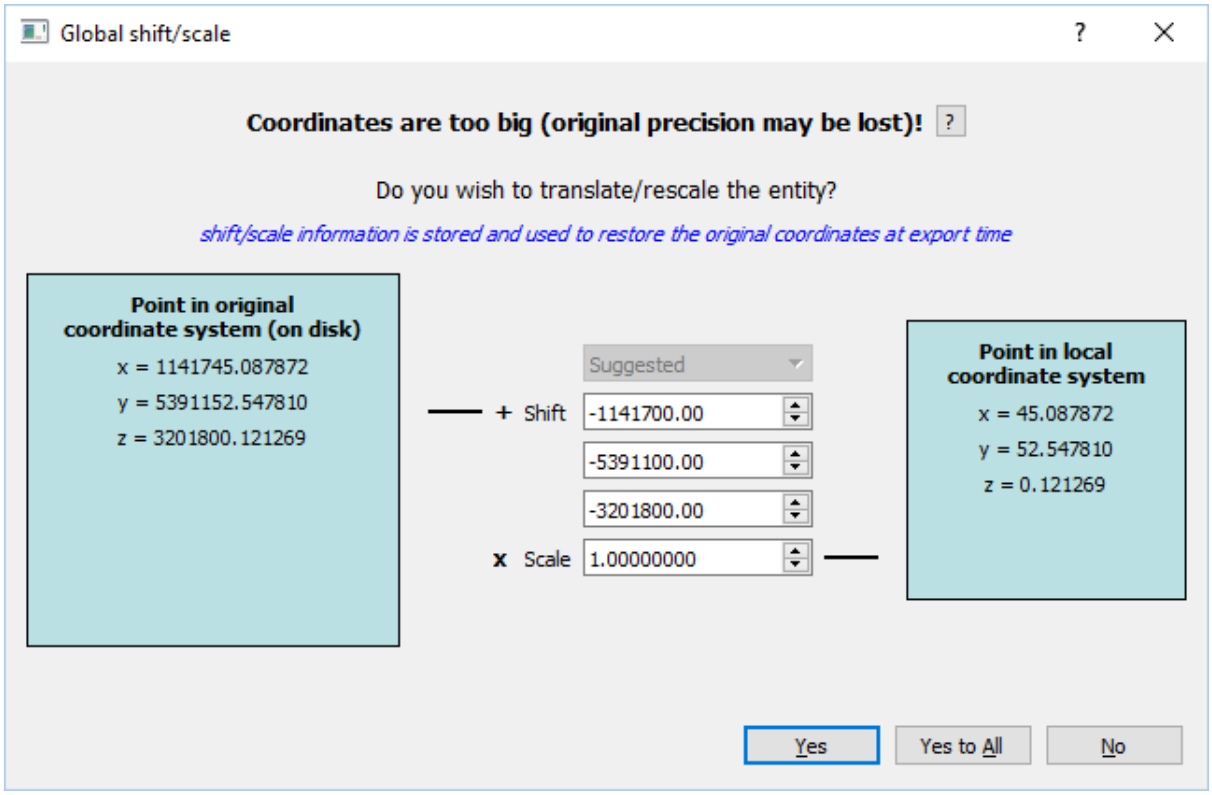


Fig 25: Global Shift Settings

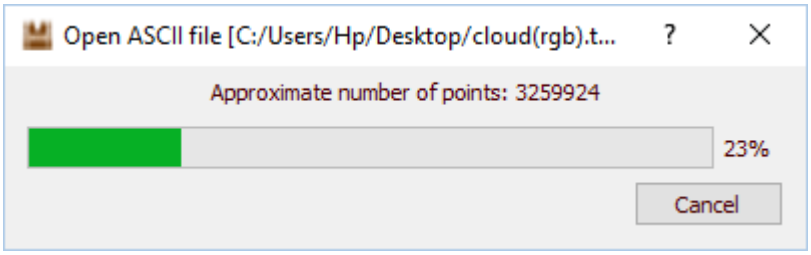


Fig 26: On Opening Point Cloud

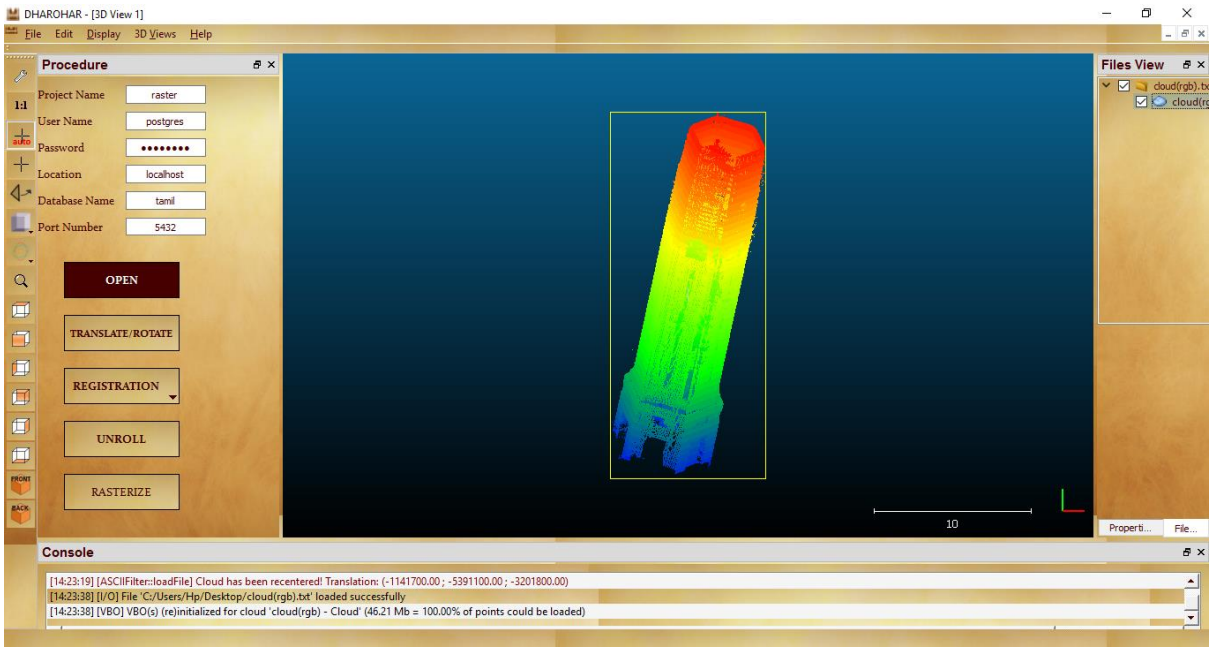


Fig 27: View of Opened point cloud

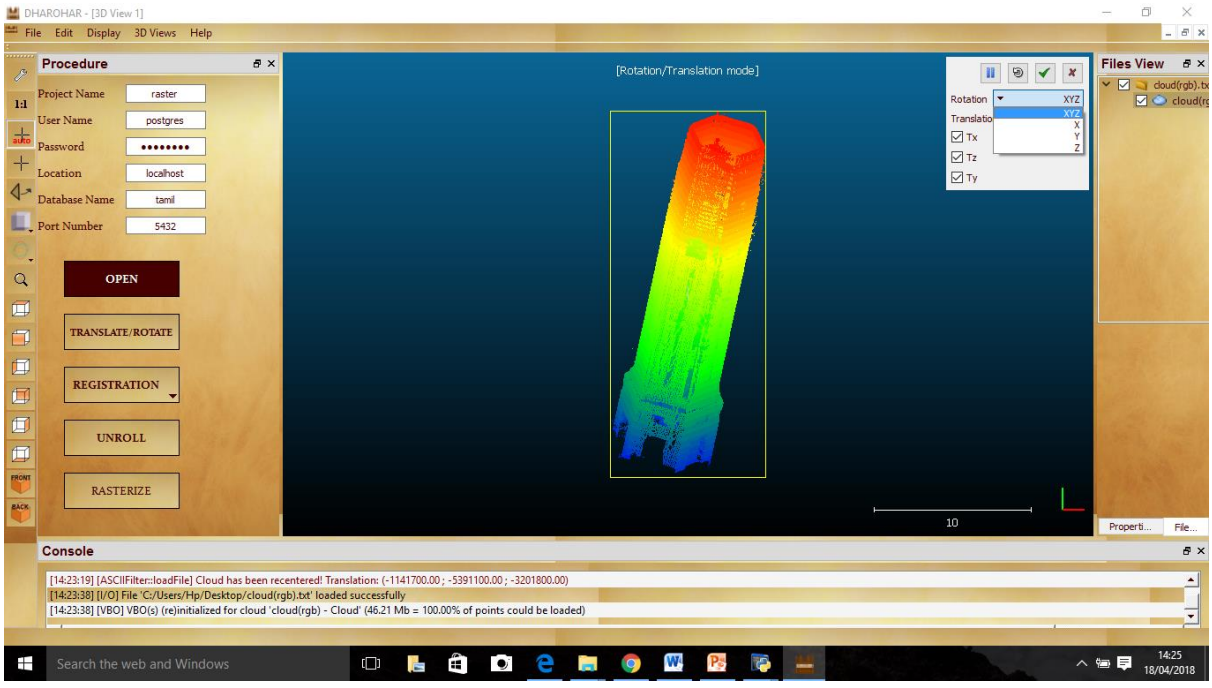


Fig 28: Translate / Rotate

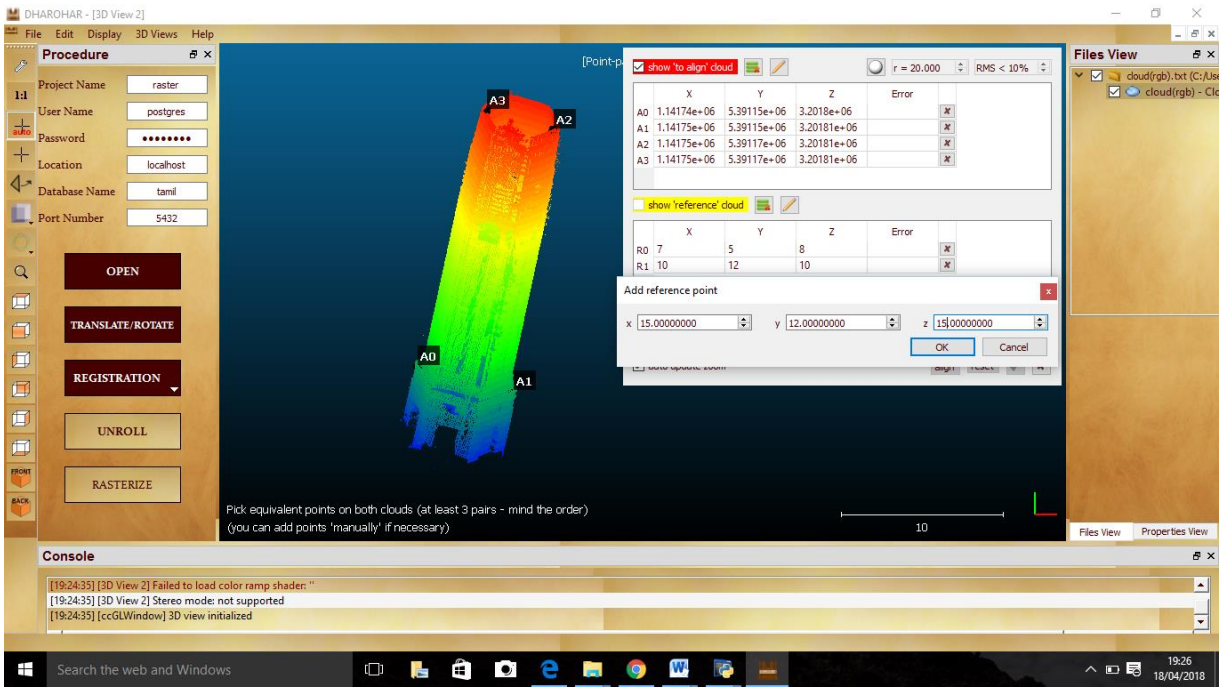


Fig 29: Align Point pair registration

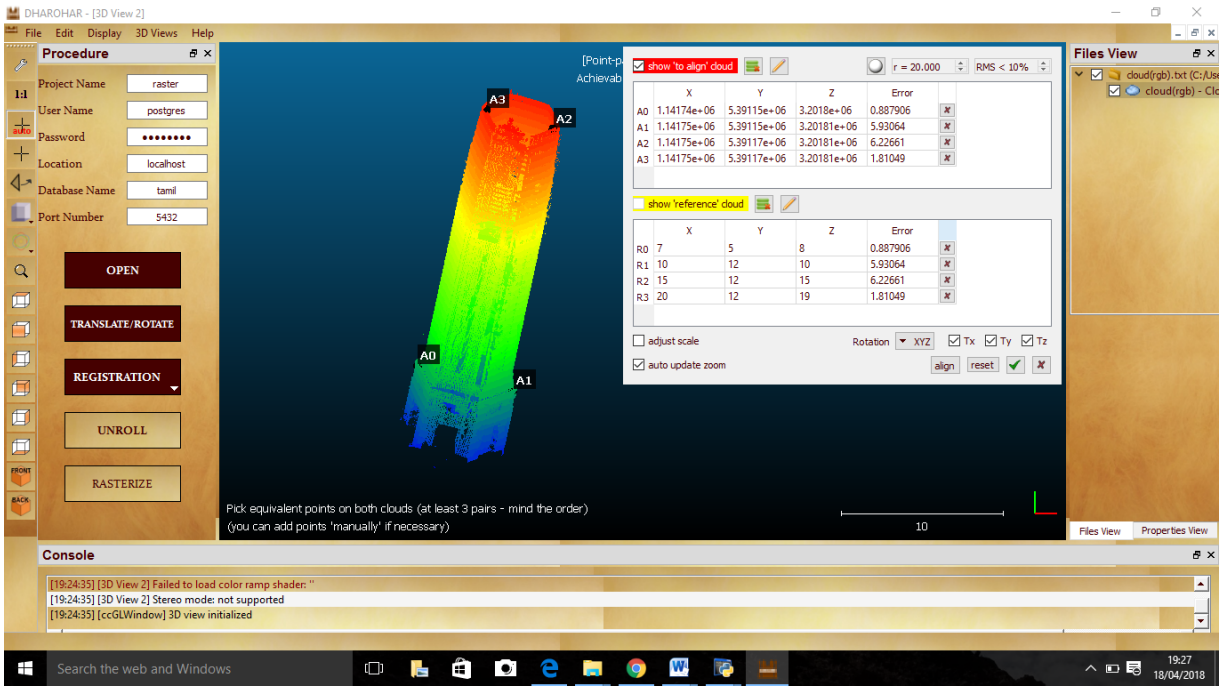


Fig 30: On Applying Registration

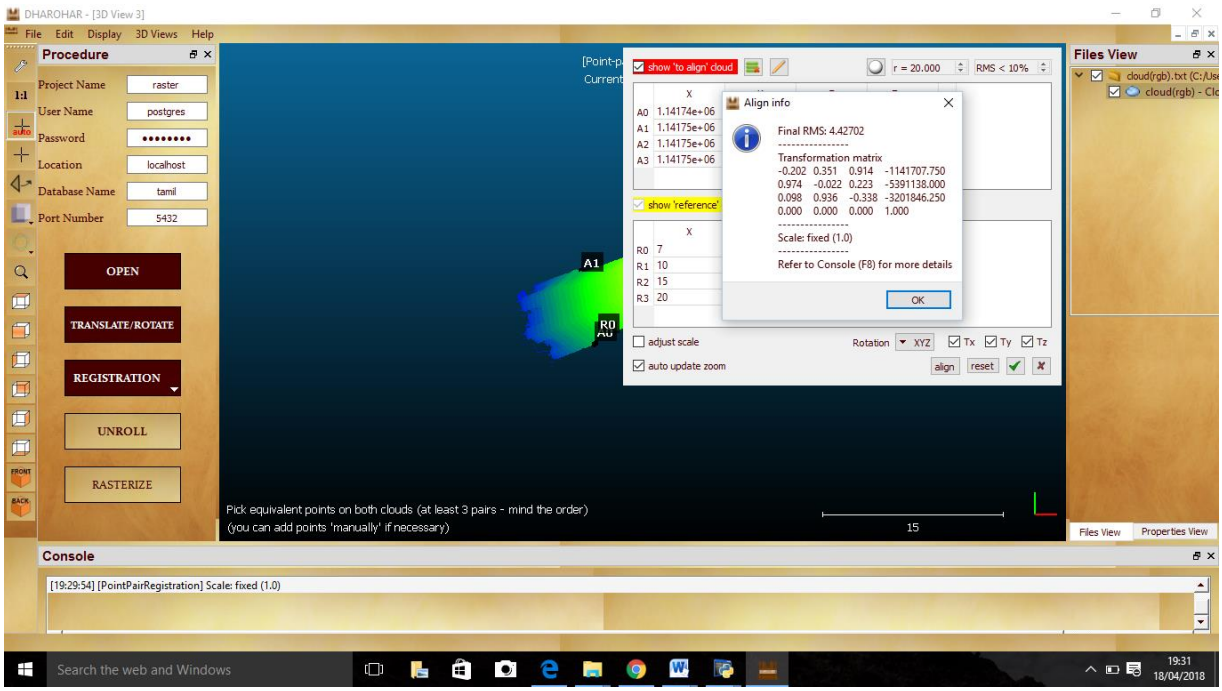


Fig 31: RMS output after Registration

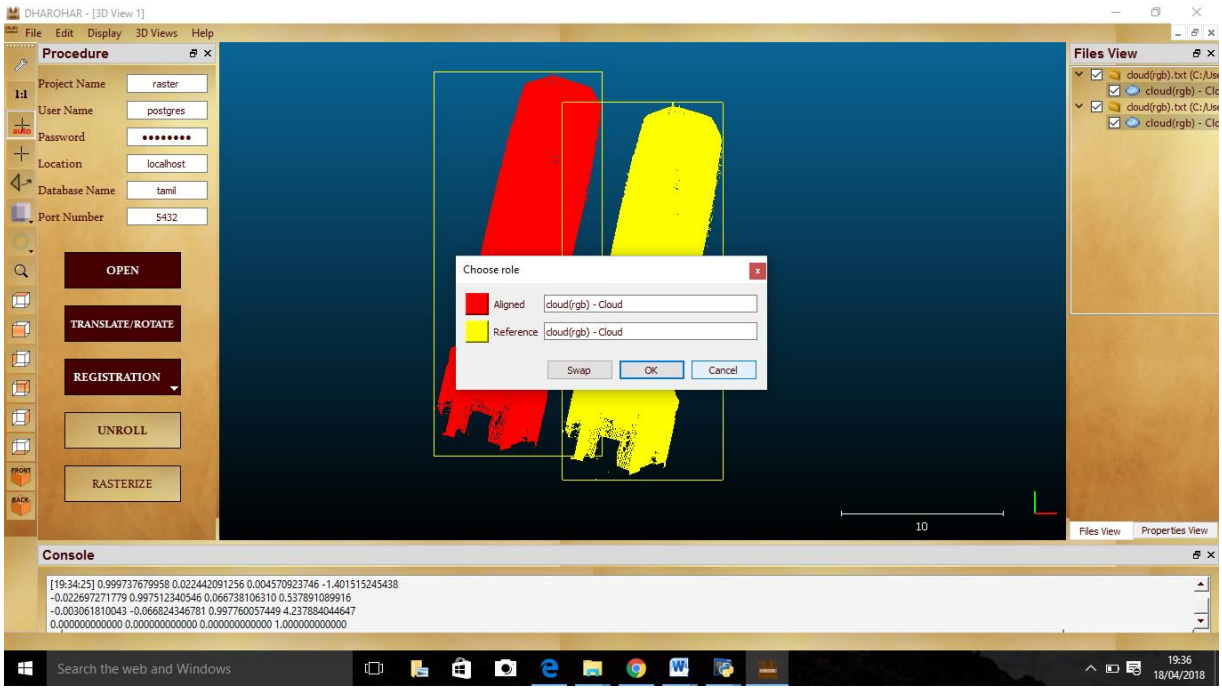


Fig 32: Fine Registration

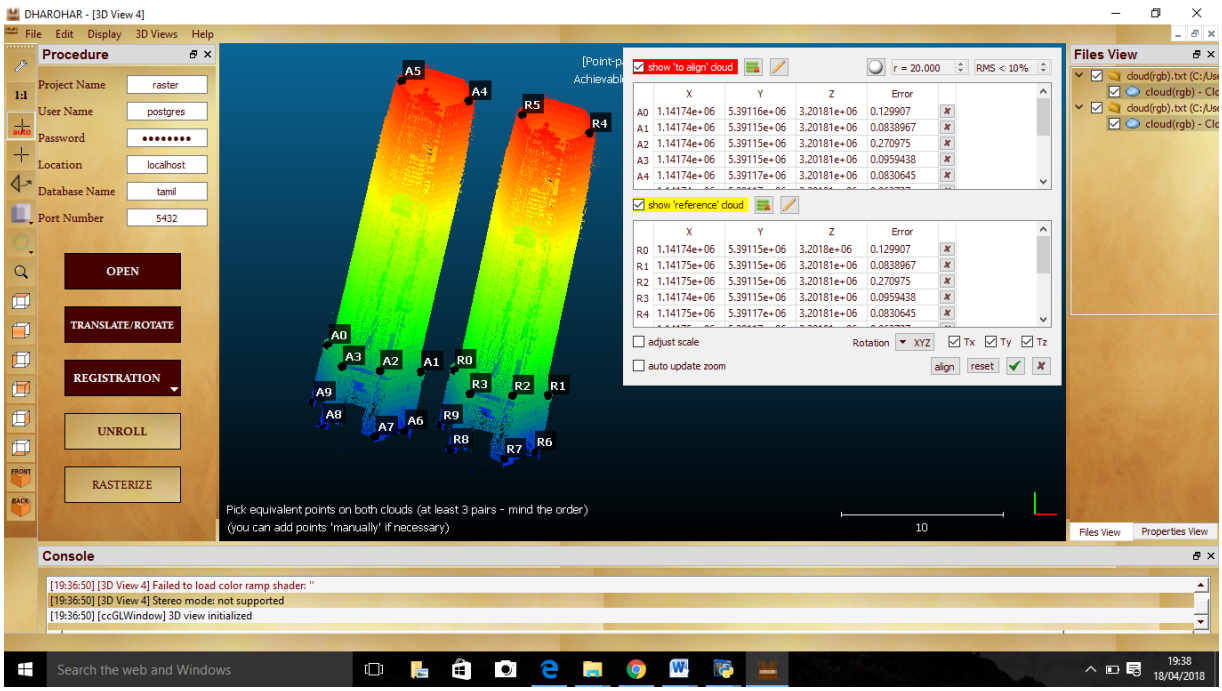


Fig 33: Picking Points on Align and Reference point cloud

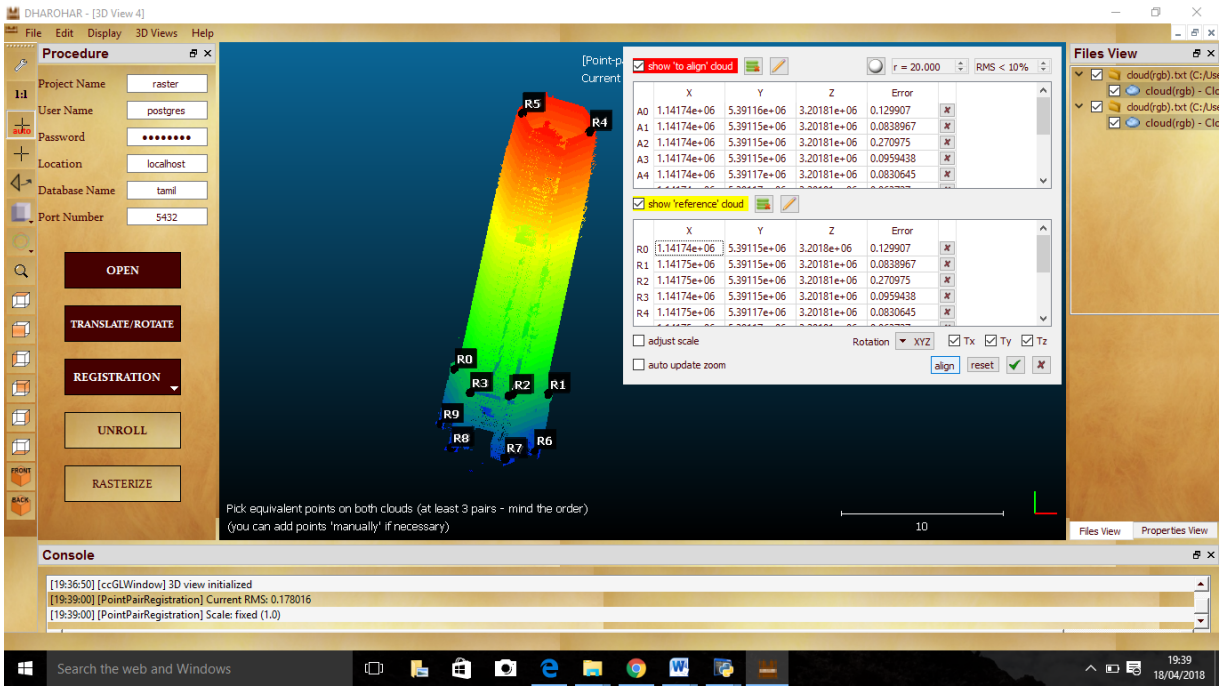


Fig 34: On Applying Registration

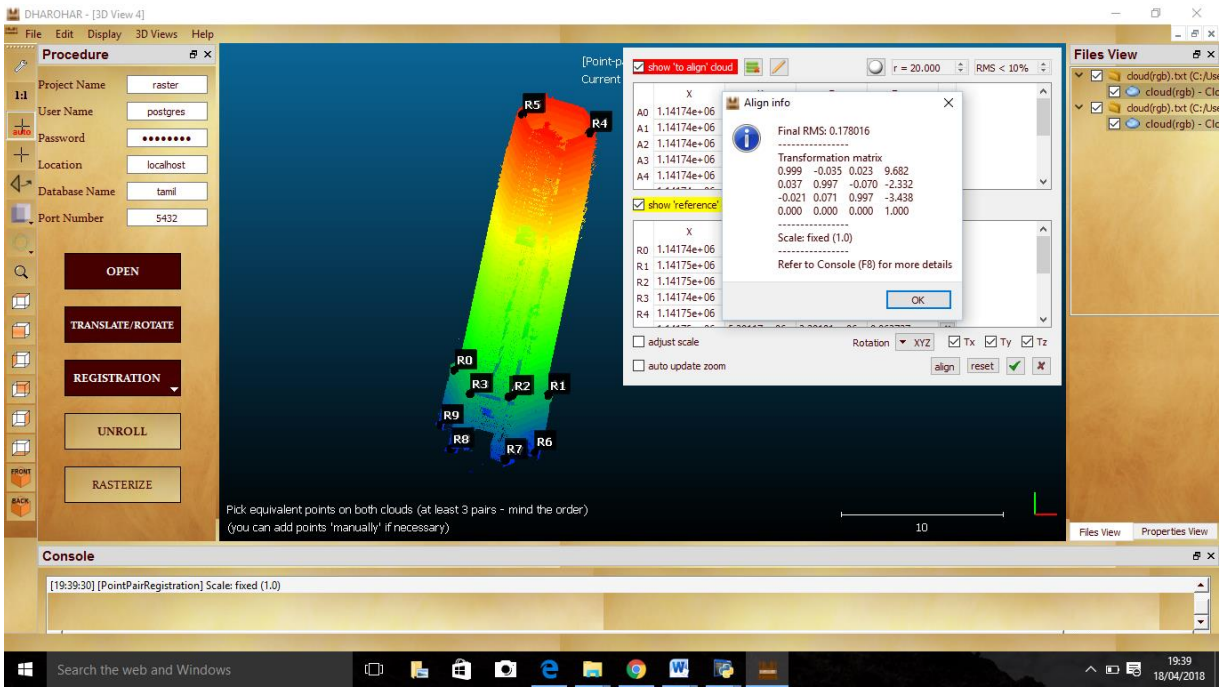


Fig 35: RMS output after Registration

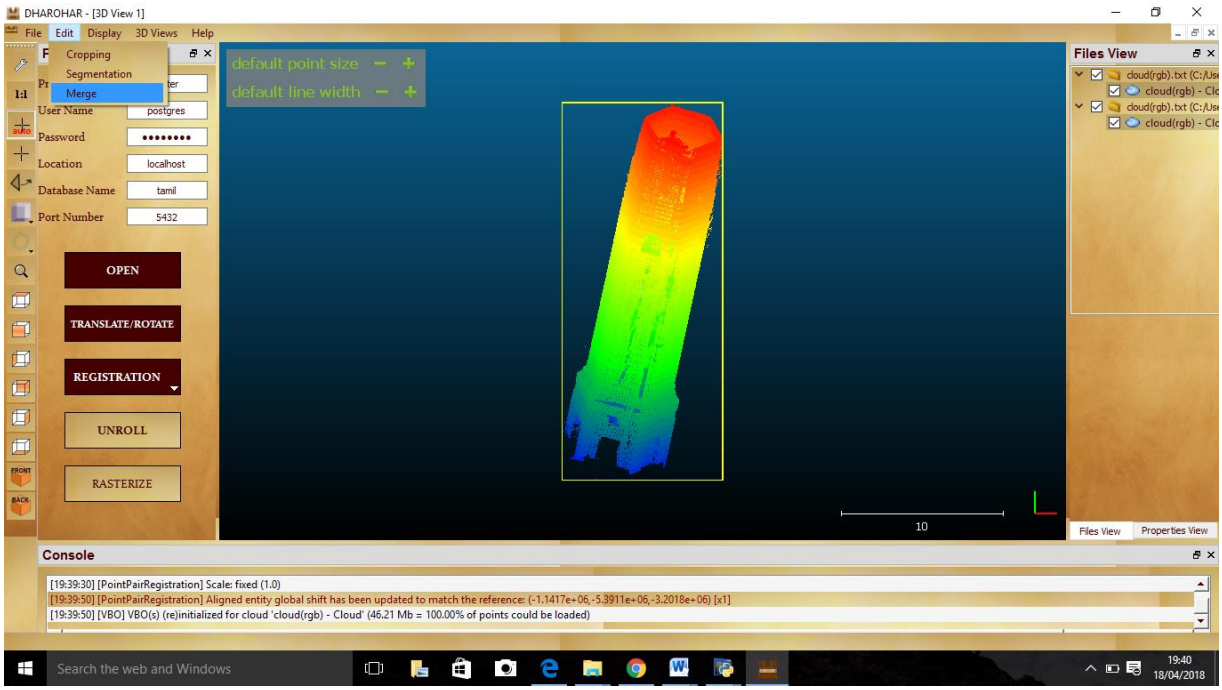


Fig 36: Merging Registered Point cloud

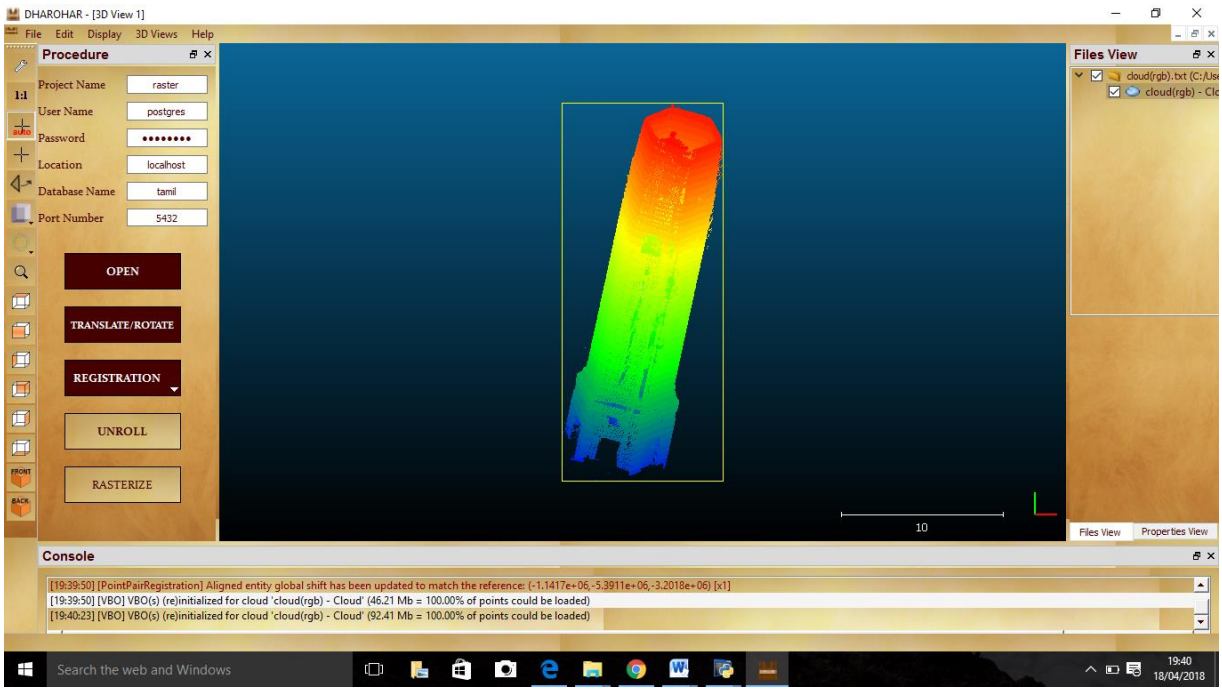


Fig 37: Merged point cloud

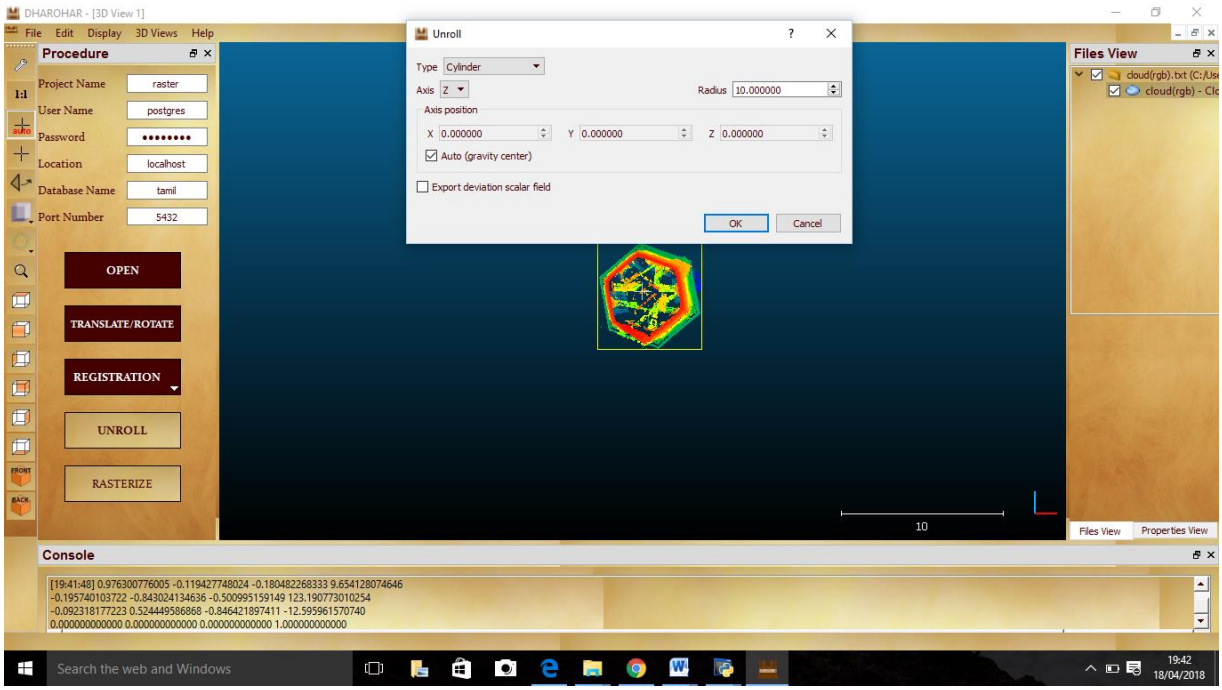


Fig 38: Unroll

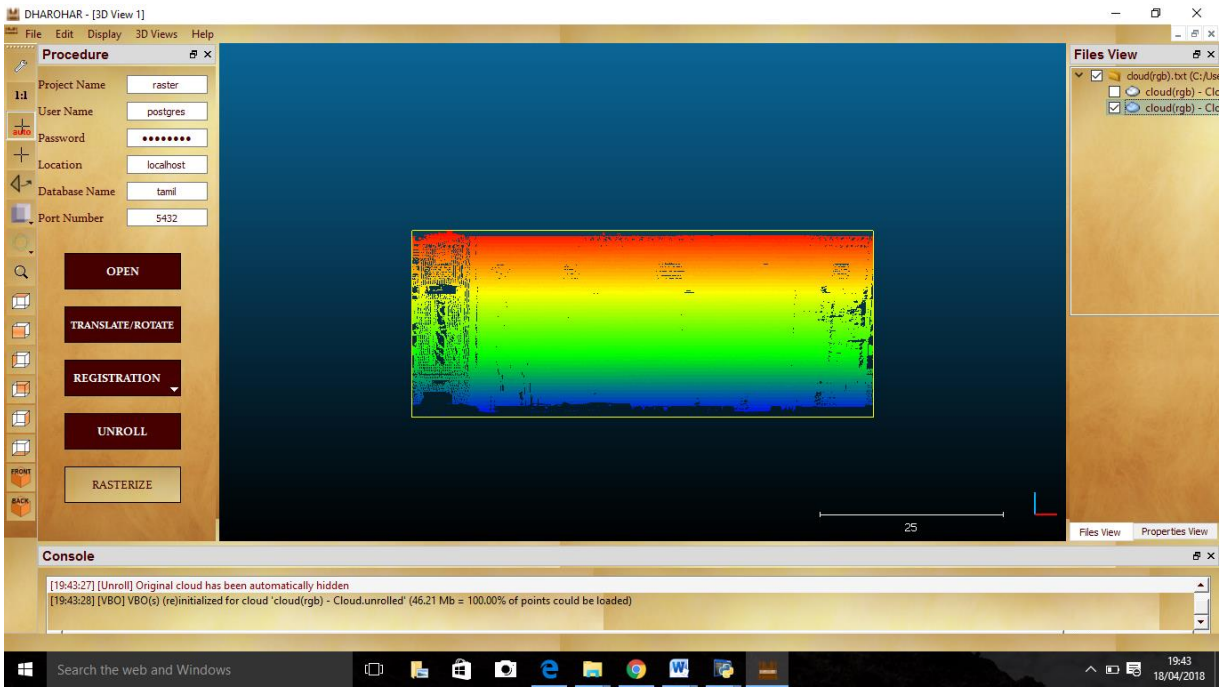


Fig 39: Unrolled Point cloud

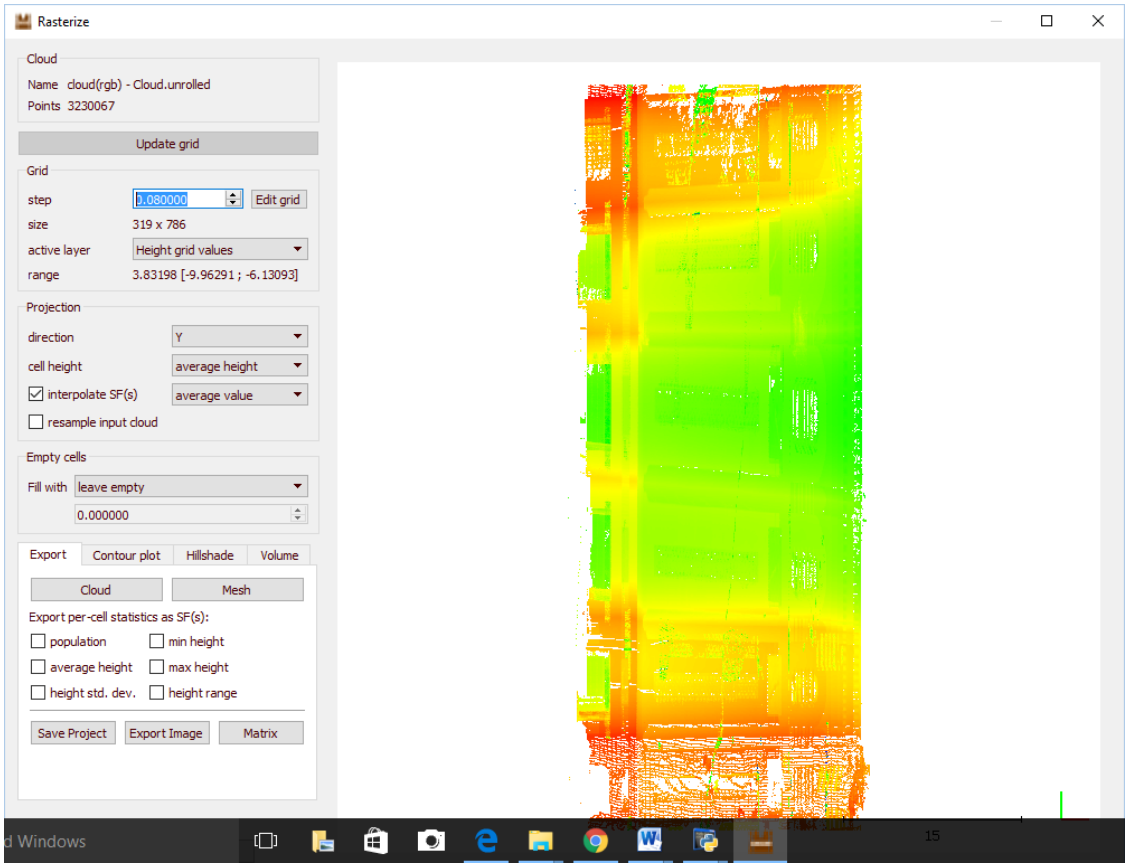


Fig 40: Rasterization

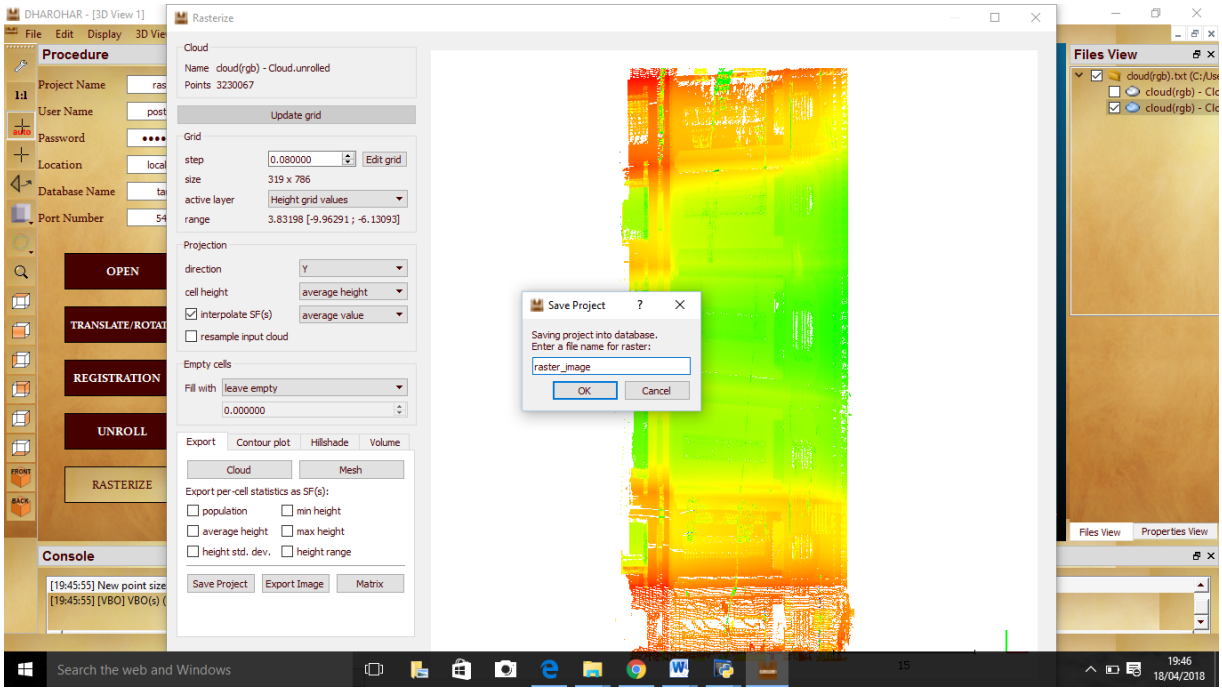


Fig 41: Saving Raster Image to Database

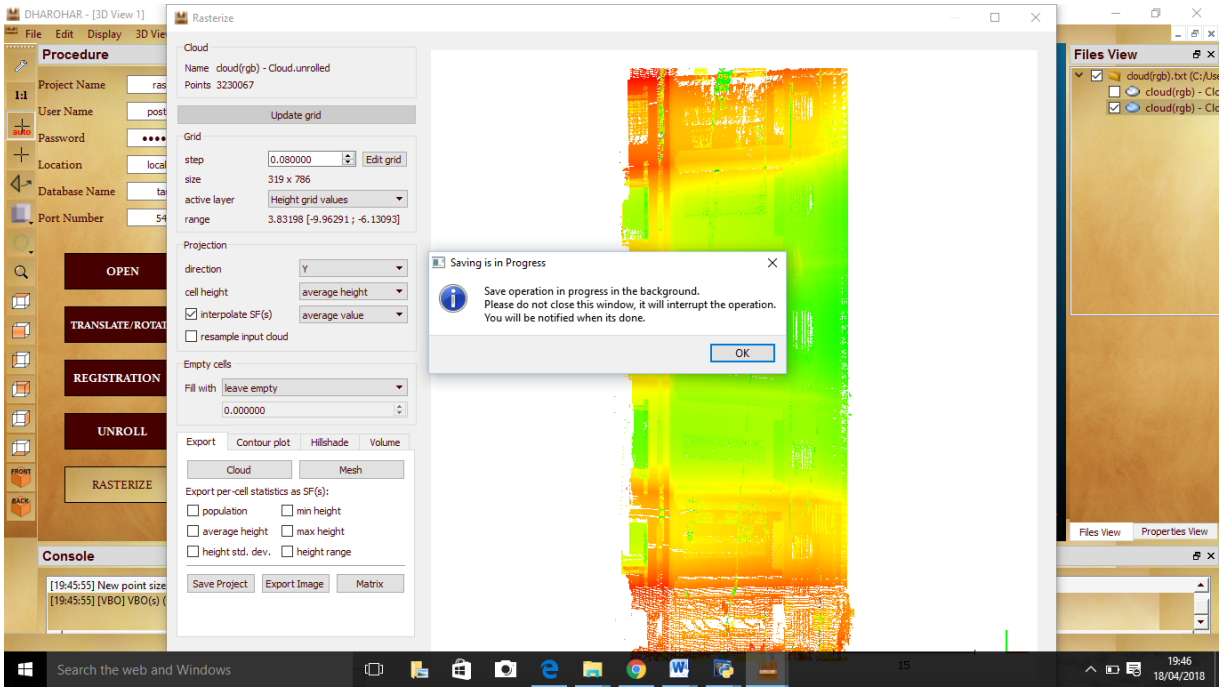


Fig 42: Background processing for saving raster image

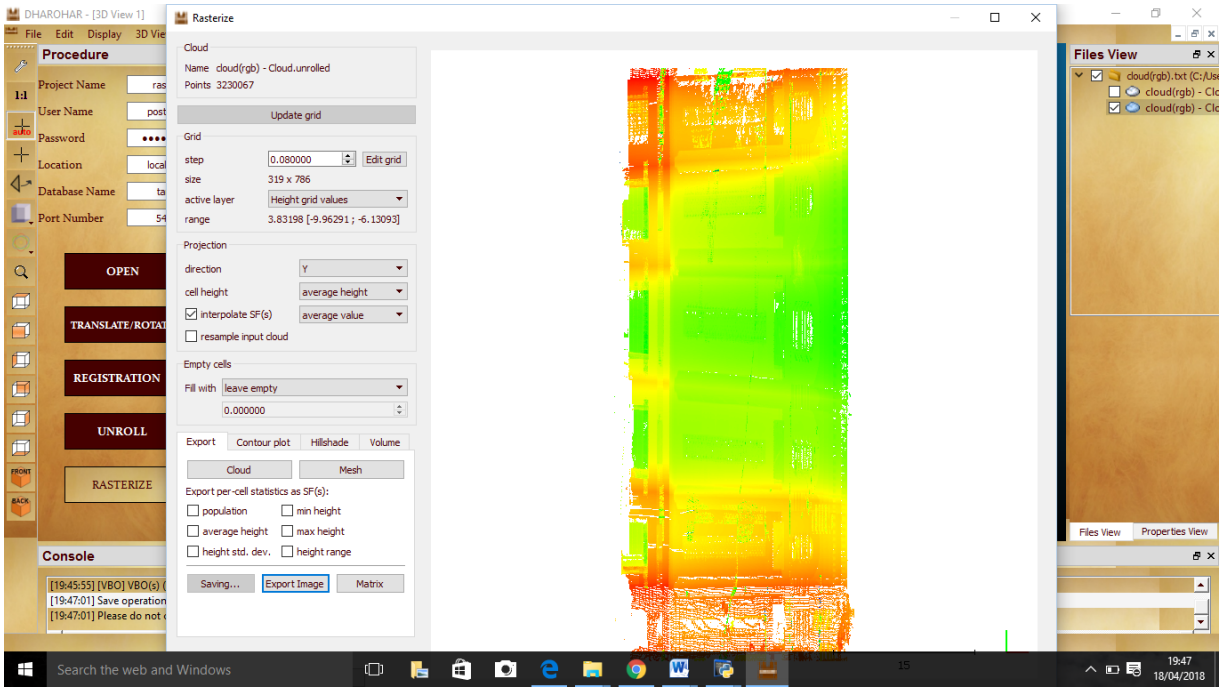


Fig 43: Saving on Process

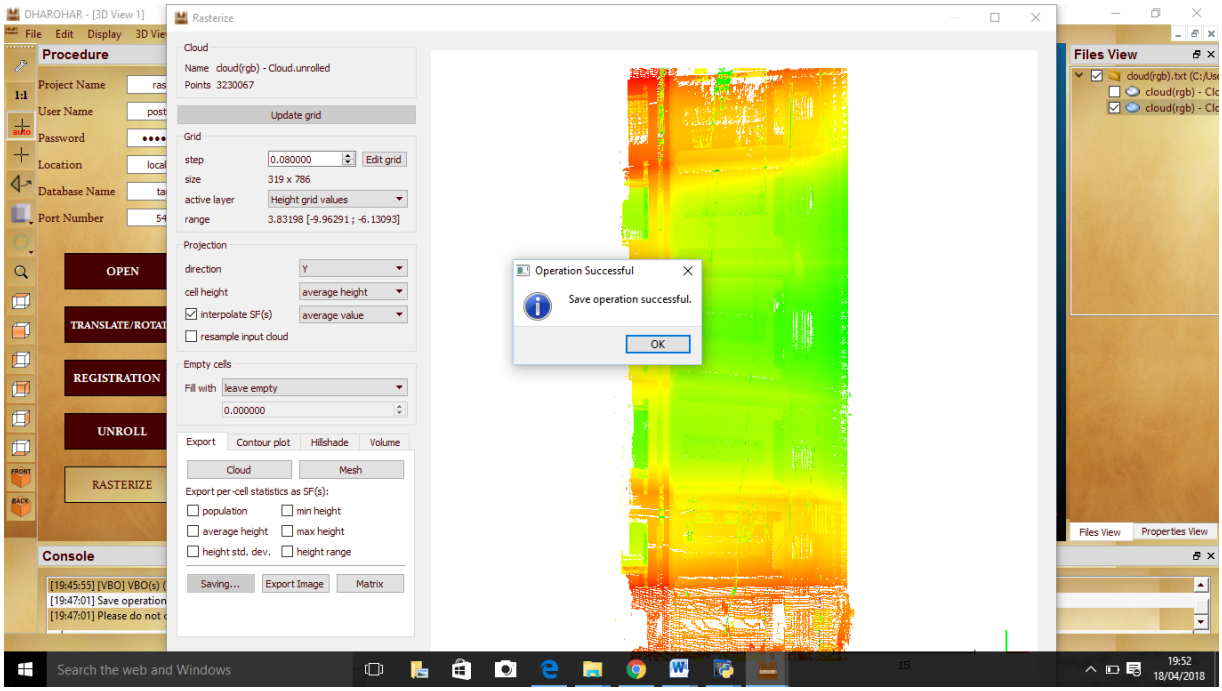


Fig 44: Raster Image saved to database

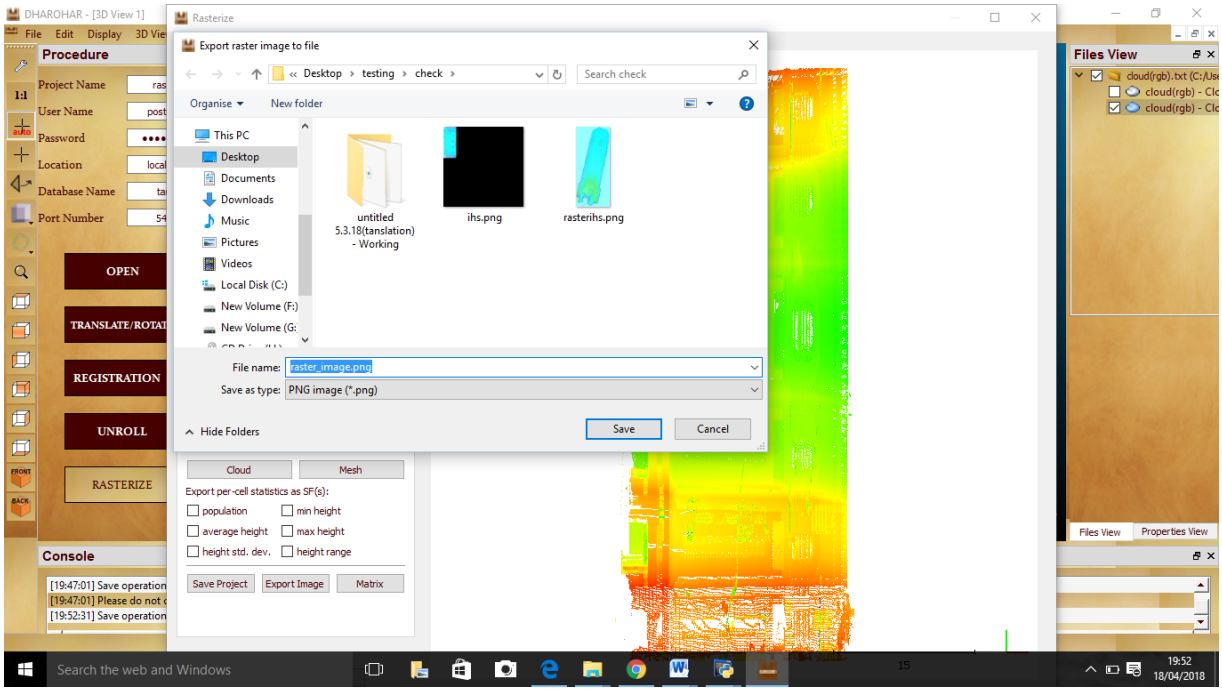


Fig 45: Exporting raster image to file

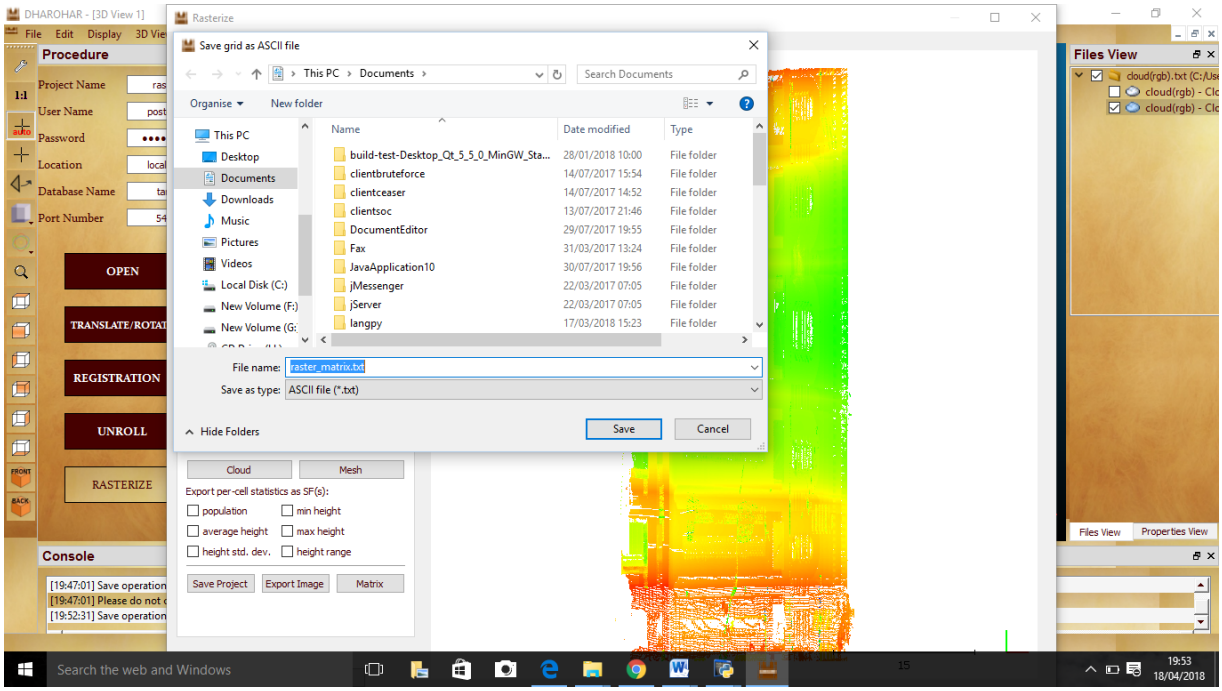


Fig 46: Exporting Matrix in the form of ASCII file to file

Fig 47: Loading raster image saved in the database

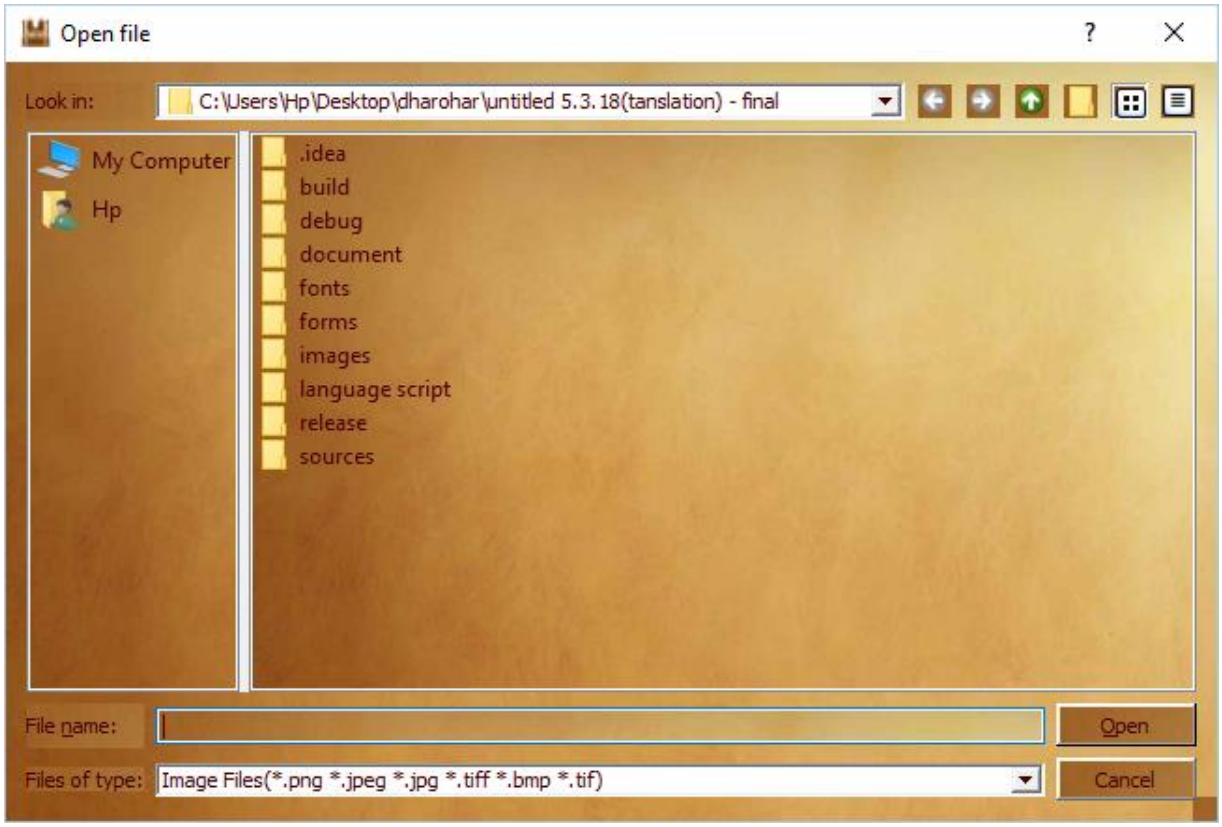


Fig 48: Loading raster image from file

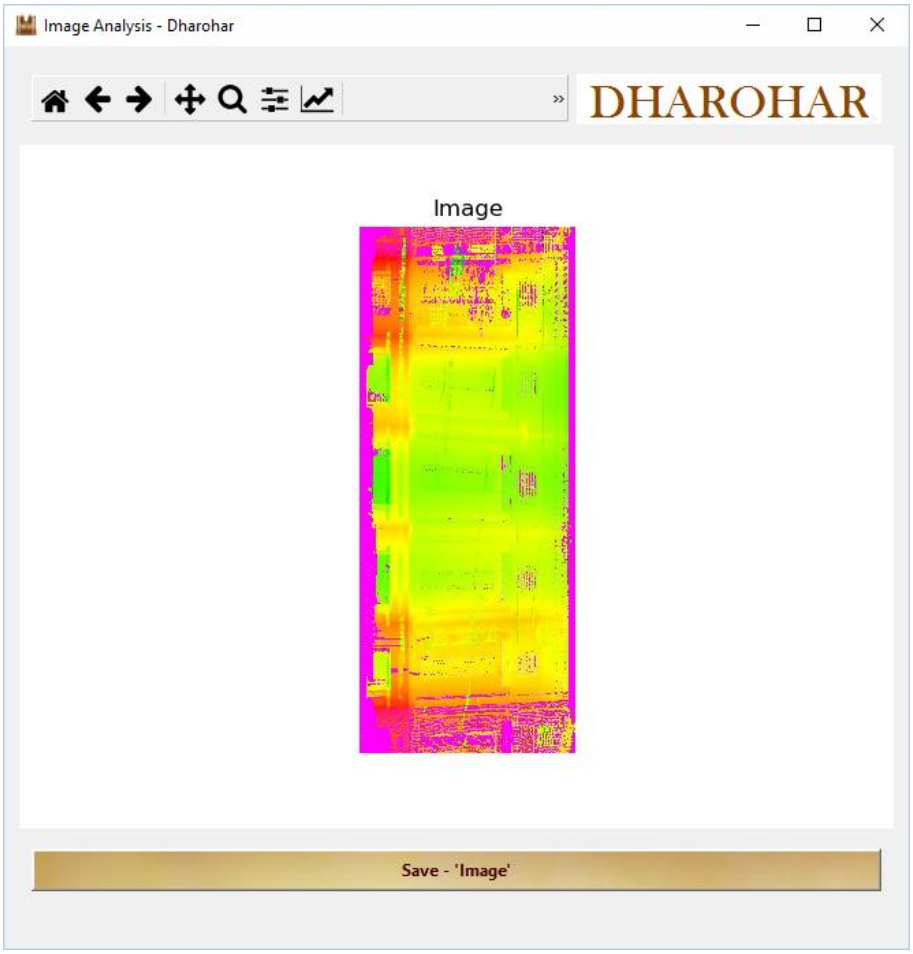


Fig 49: Viewing loaded raster image

Fig 50: Selecting filters for Edge Detection

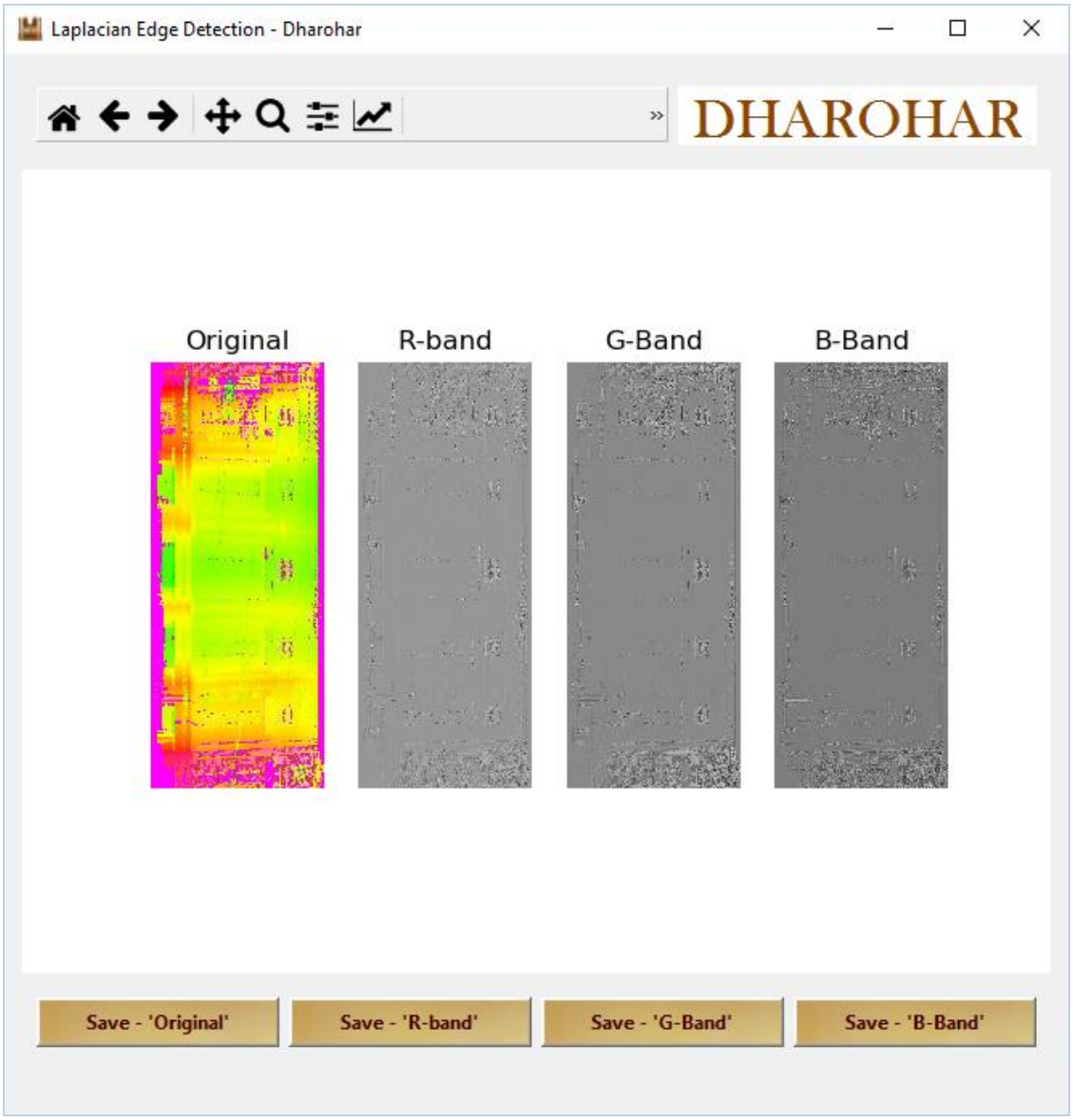


Fig 51: Laplacian output

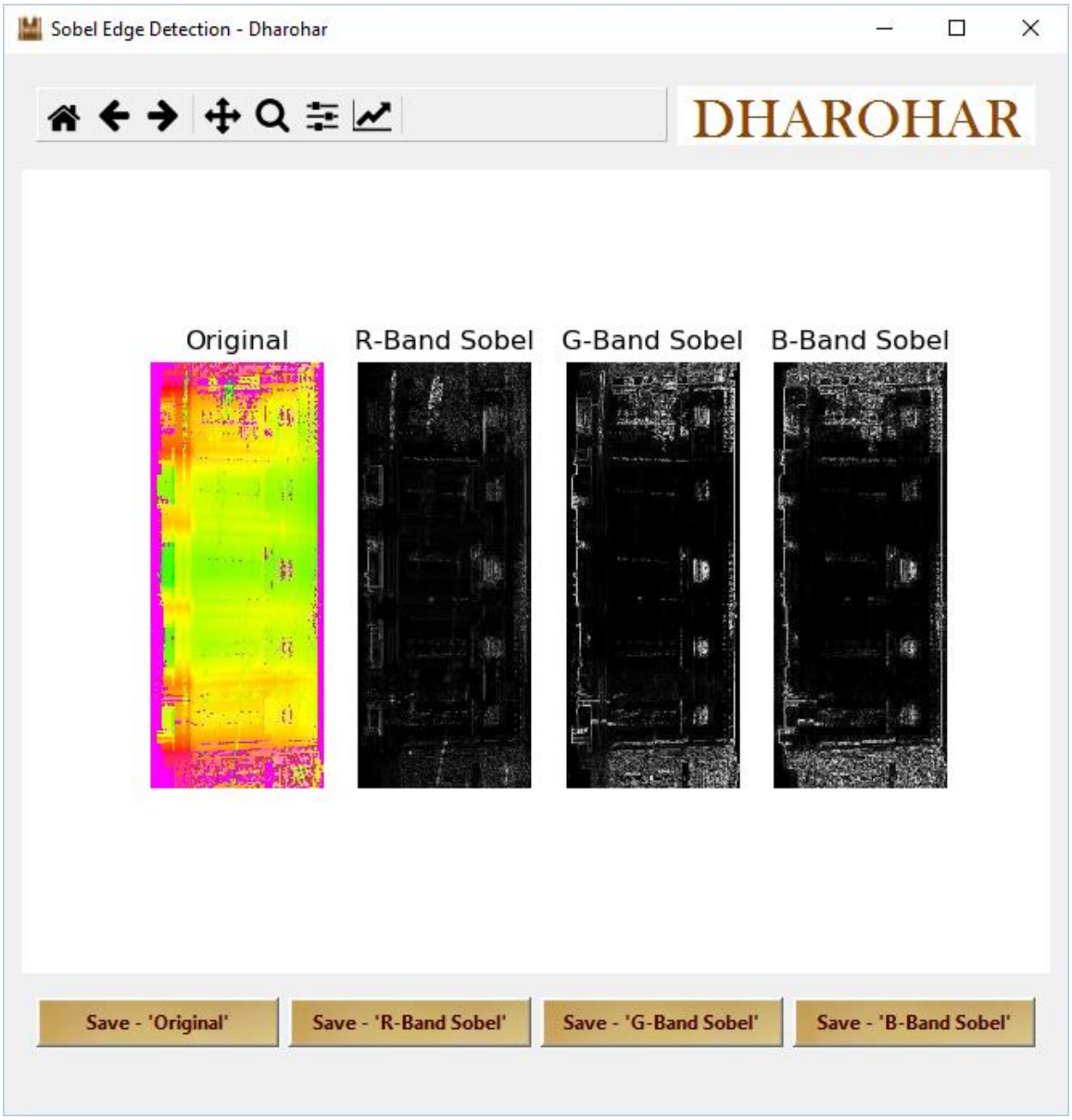


Fig 52: Sobel output

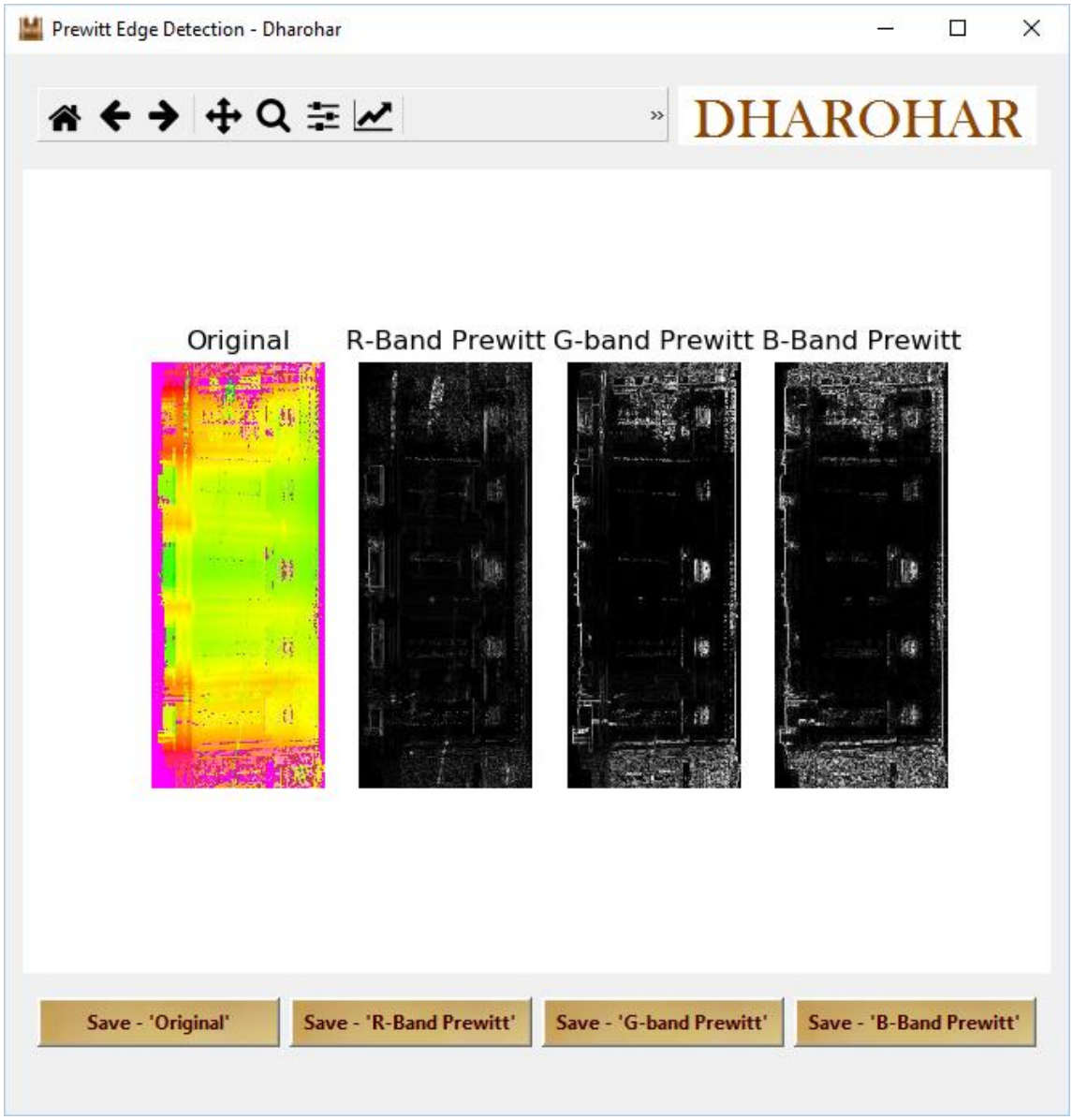


Fig 53: Prewitt output

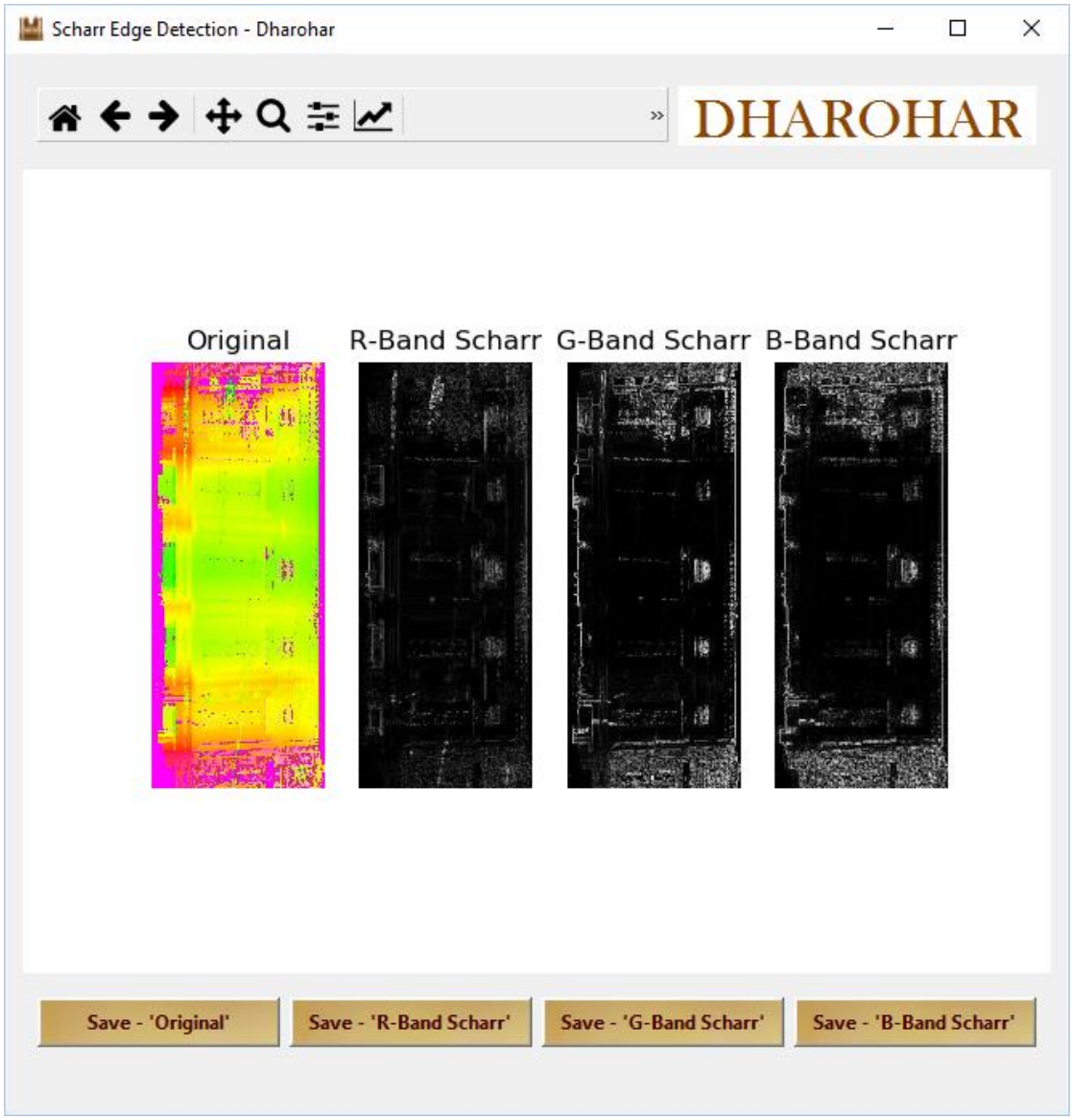


Fig 54: Scharr output

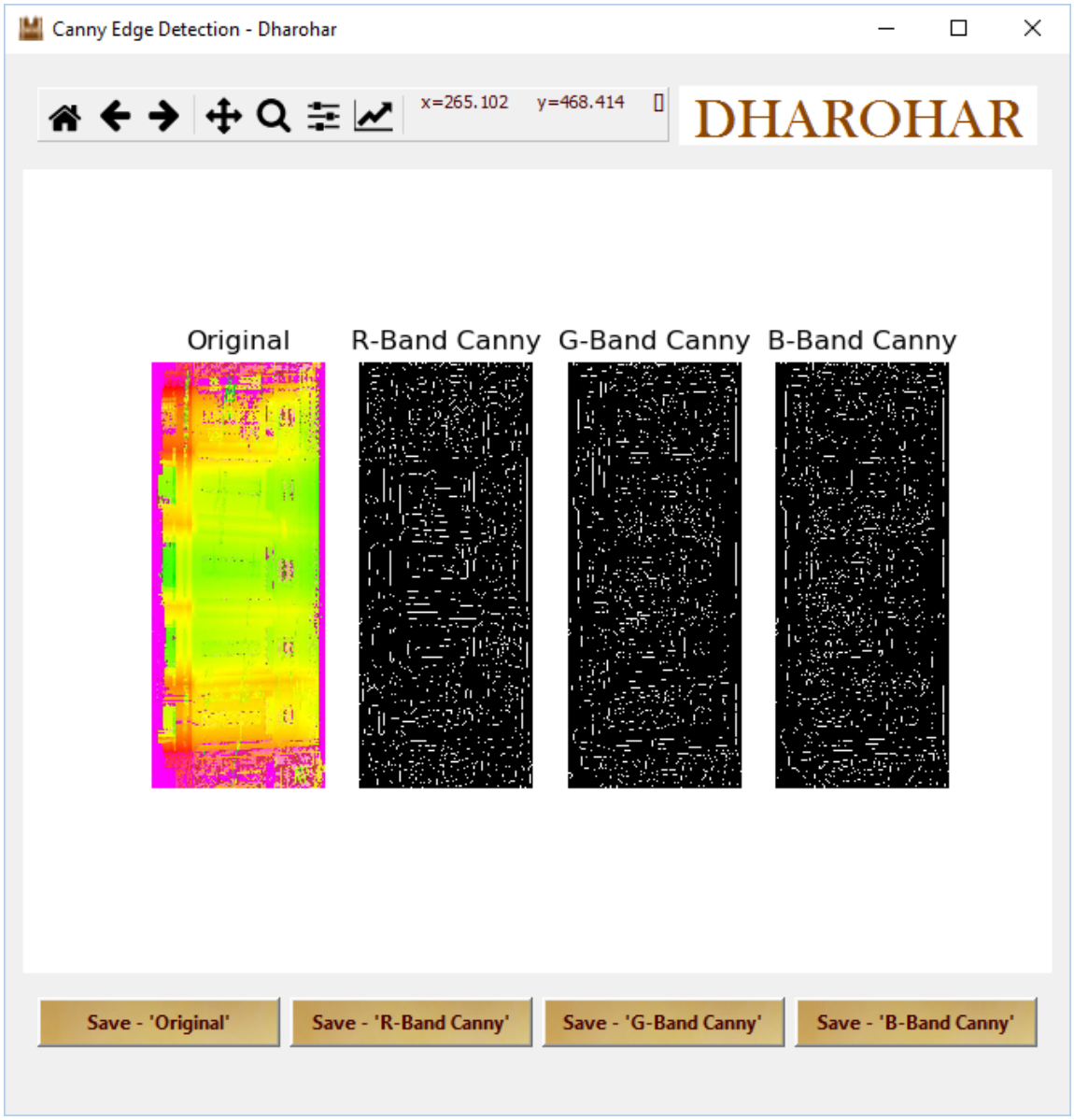


Fig 55: Canny output

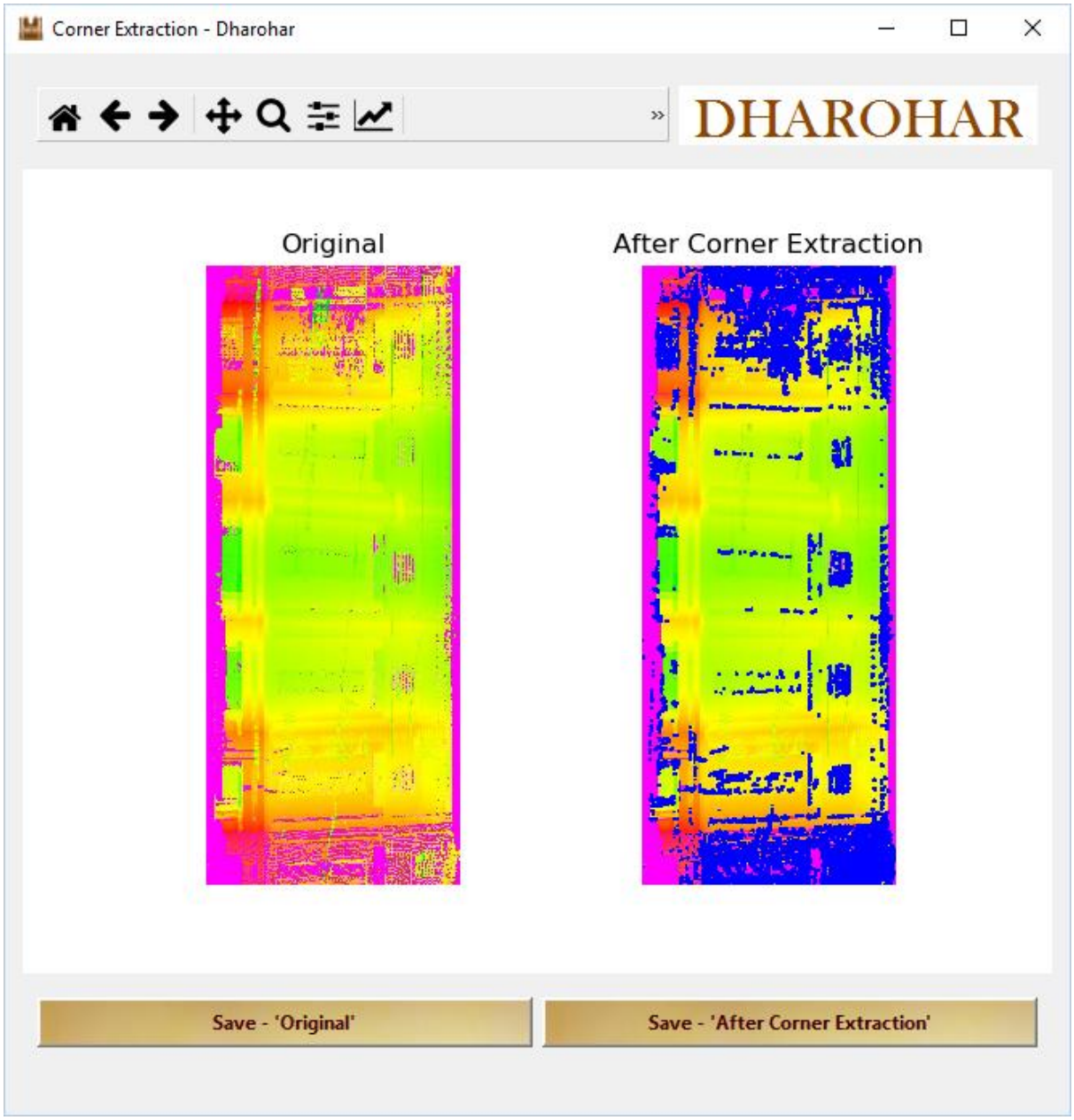


Fig 56: Corner Extraction output

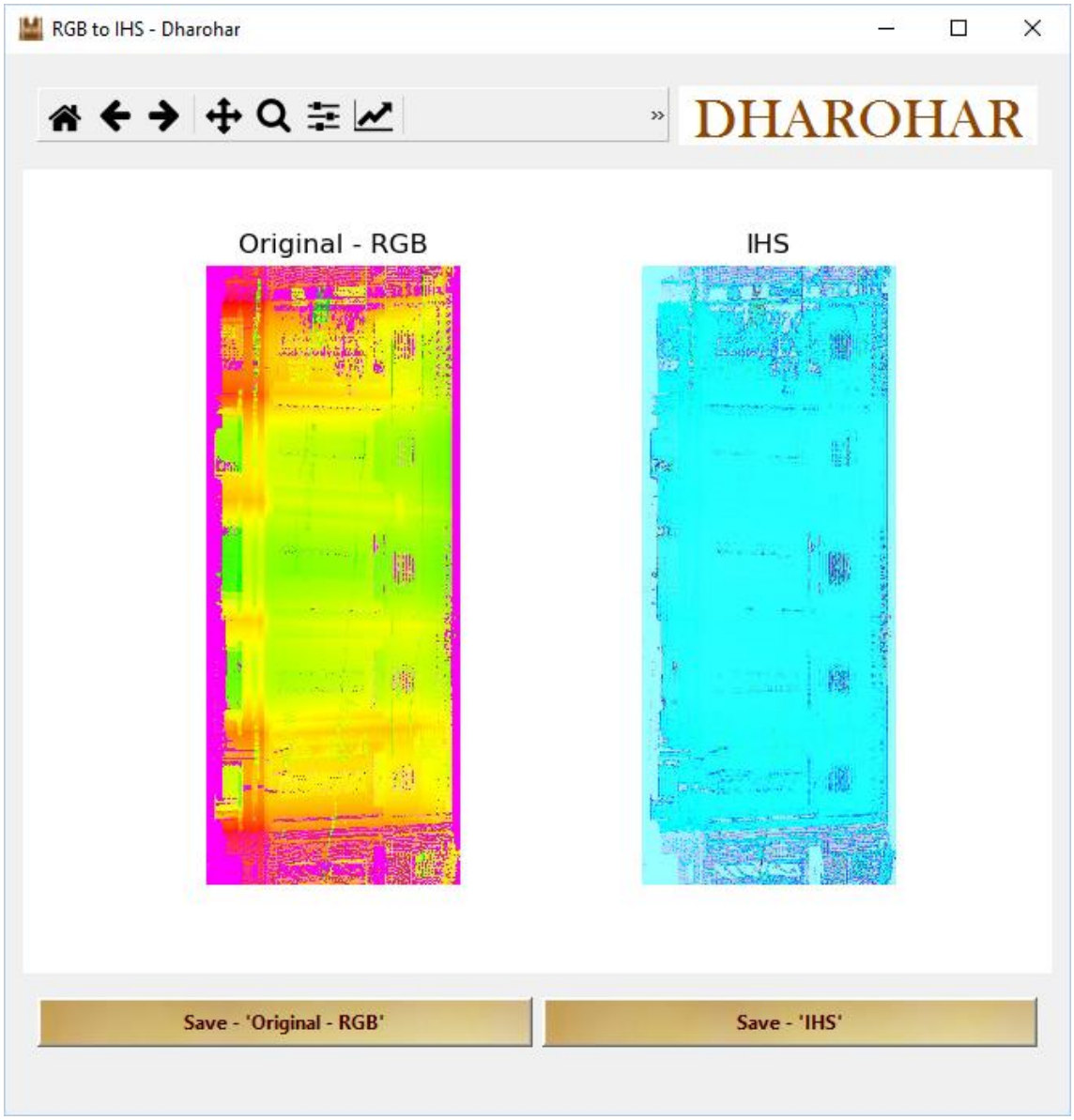


Fig 57: RGB to IHS converted output

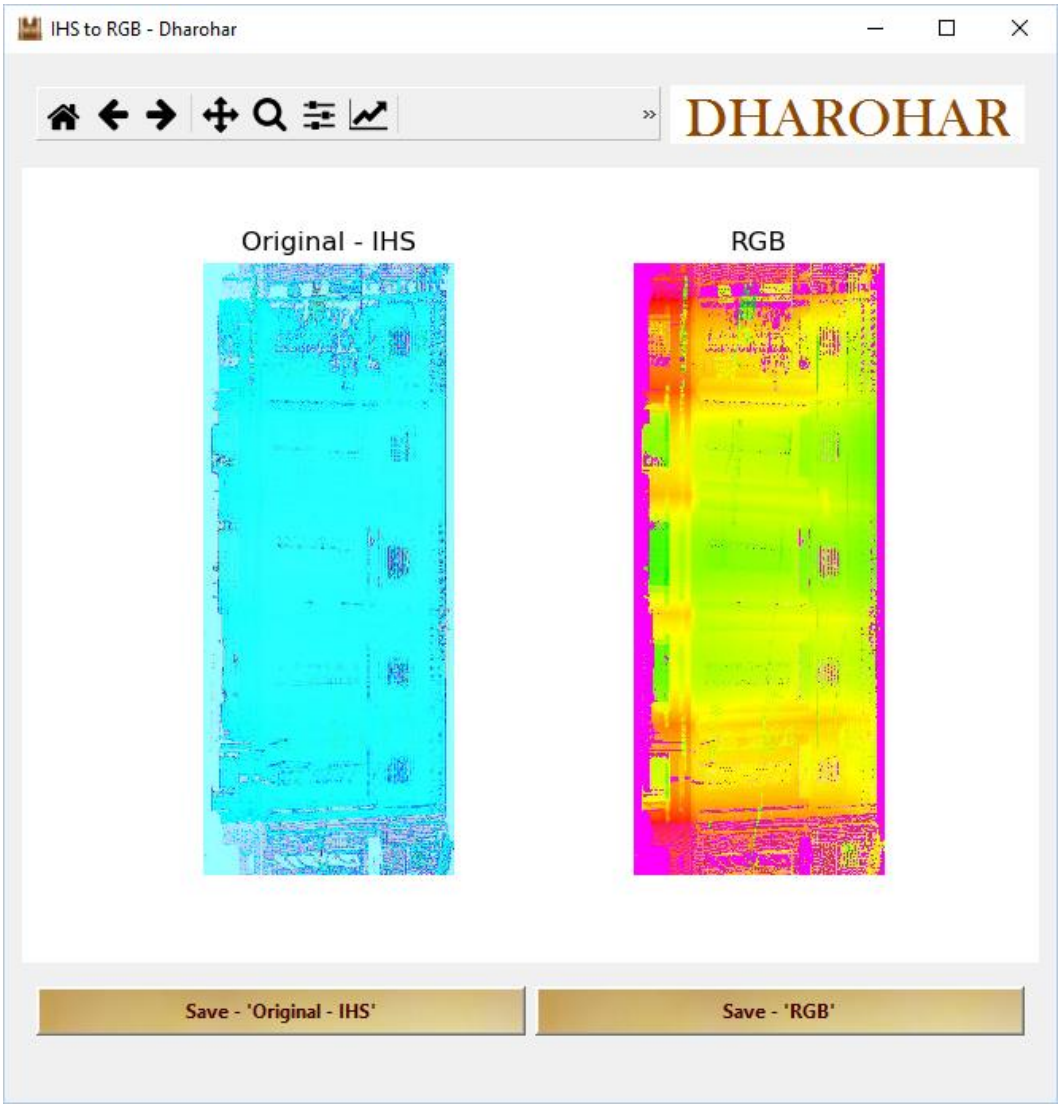


Fig 58: IHS to RGB converted output

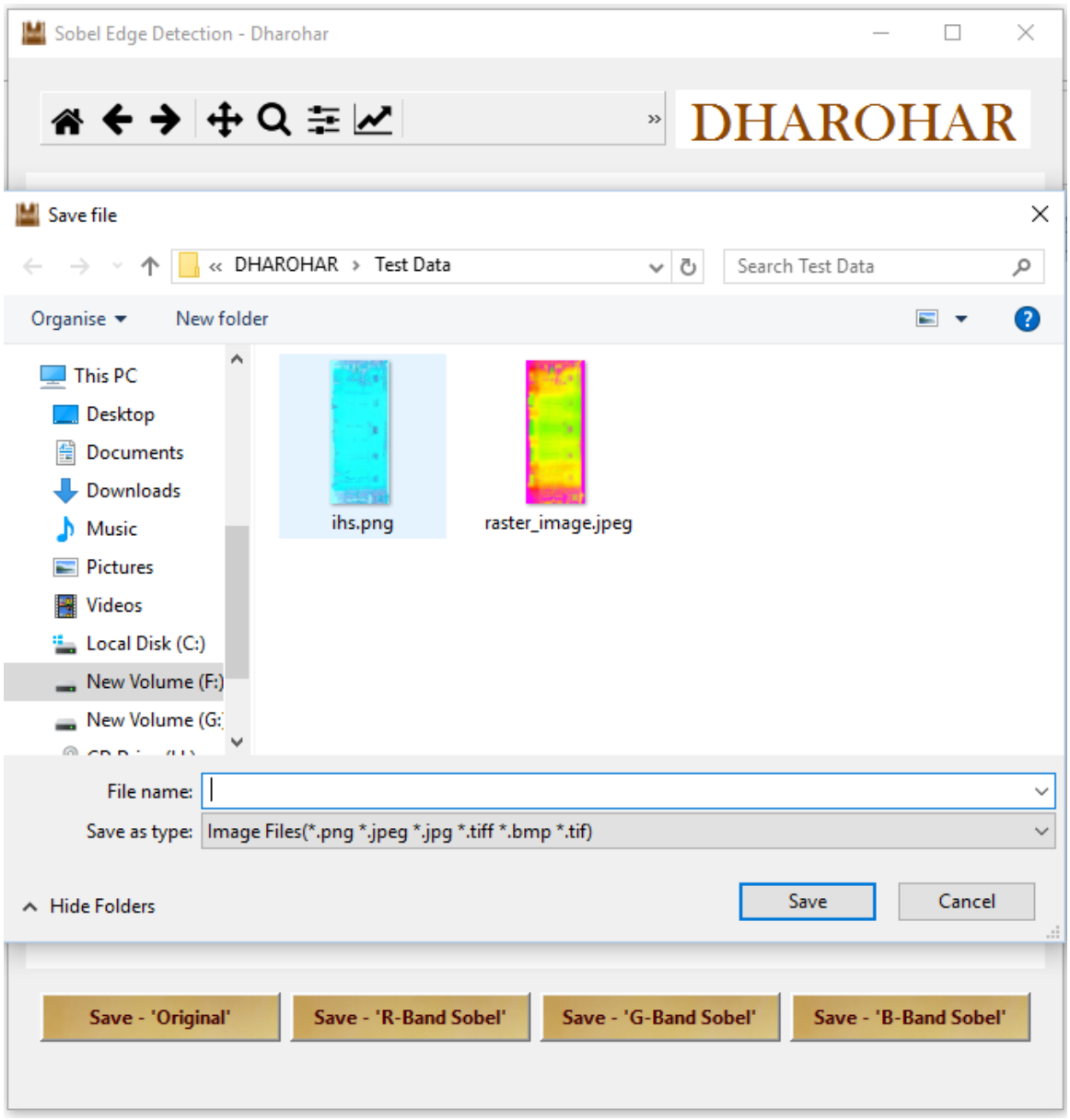


Fig 59: Saving the Individual Images from the obtained output

Fig 60: Change Detection

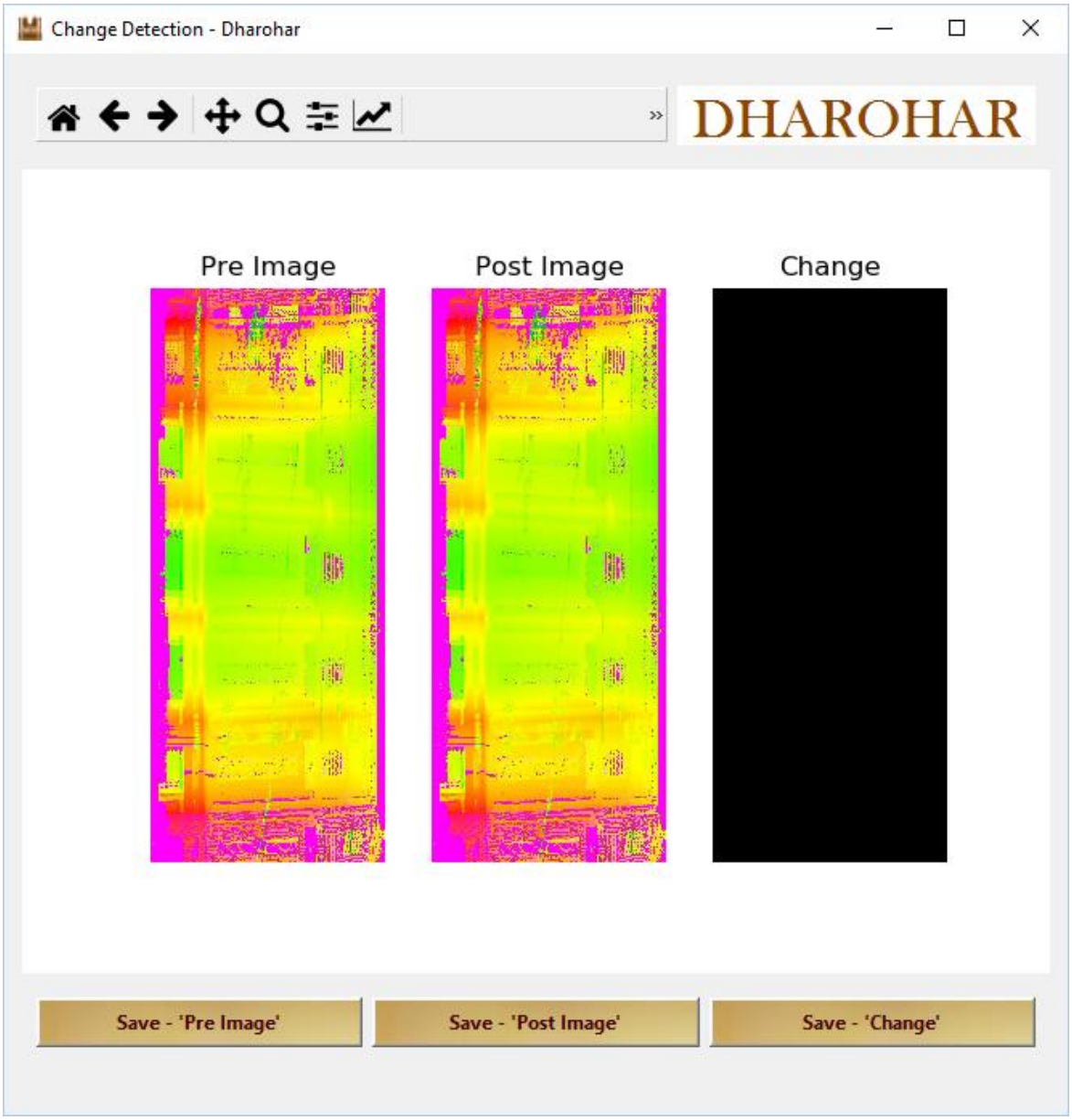


Fig 61: Output of change detection – No changes detected on loading same image

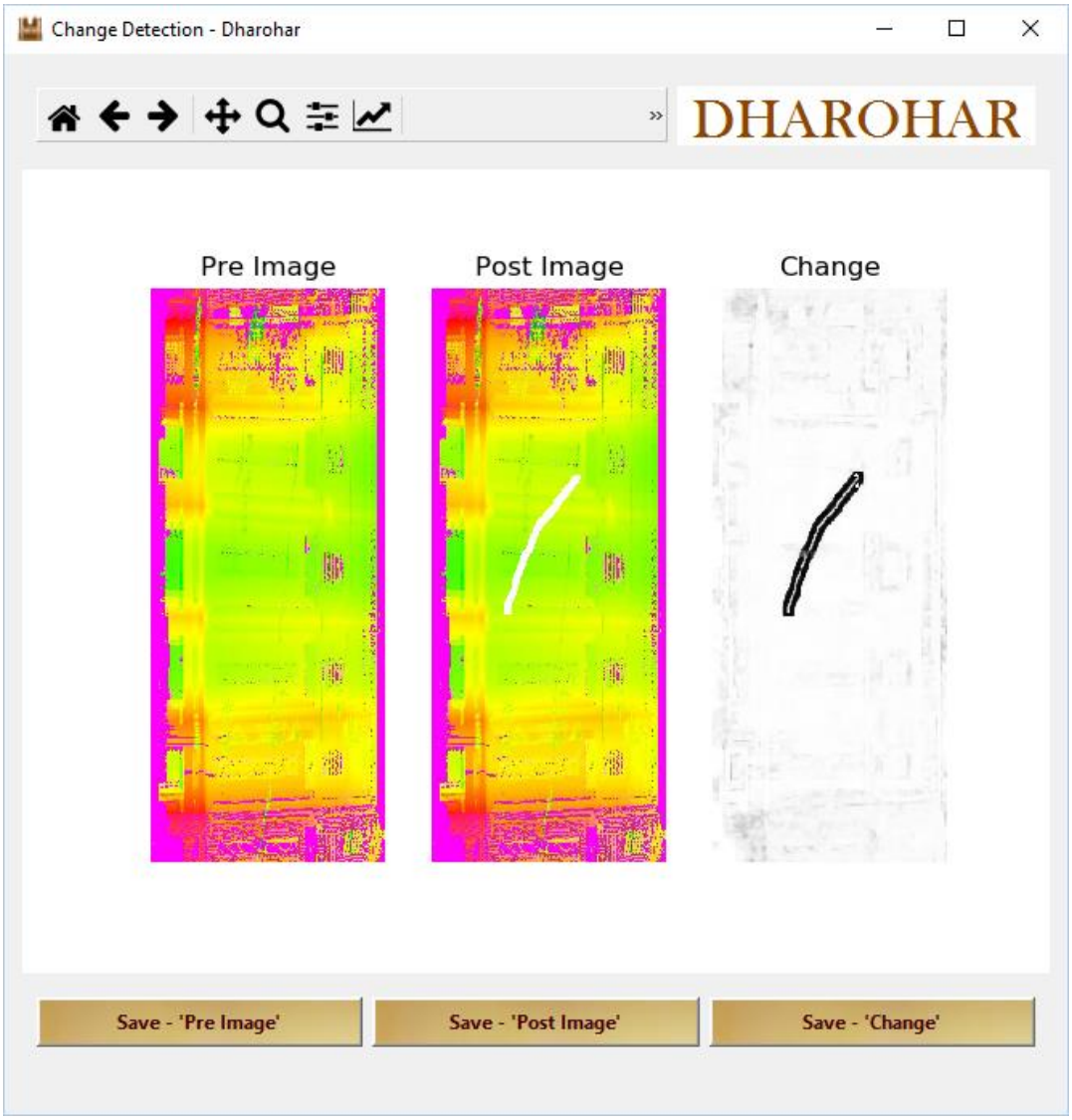


Fig 62: Output of change detection – Changes detected on loading different image

Fig 63: Texture analysis

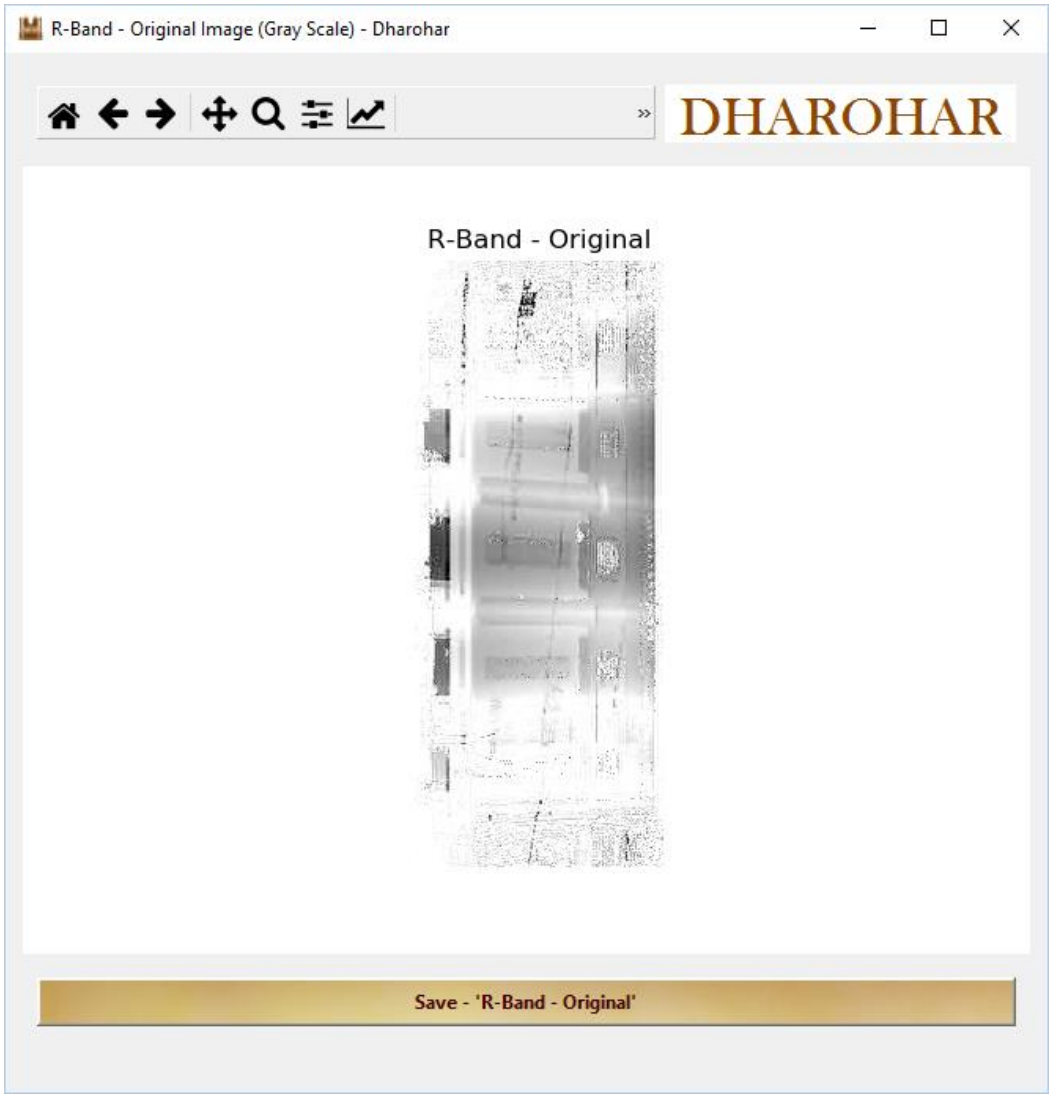


Fig 64: R-Band- Original Image

Fig 65: R-Band Pixels

Fig 66: R-Band statistical

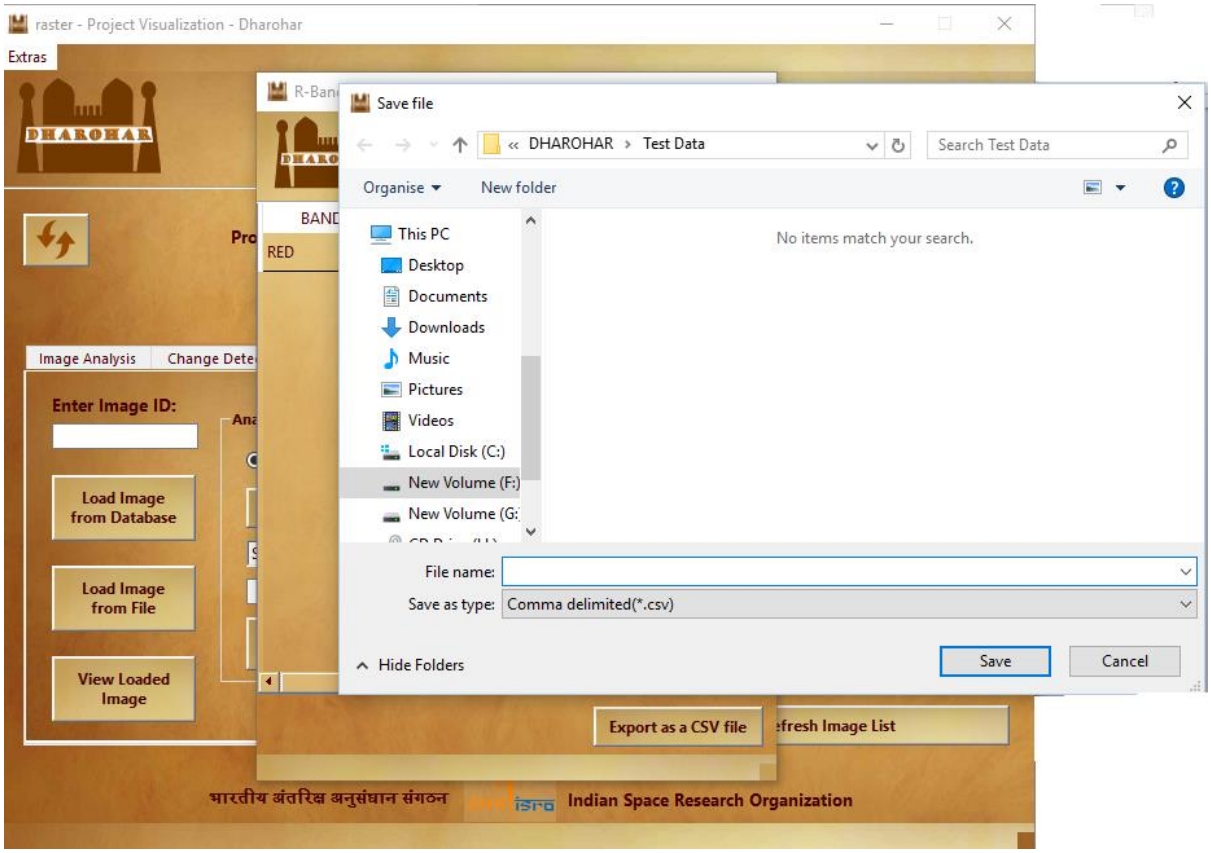


Fig 67: Exporting the table as .csv file

Fig 68: G-Band- Original Image

Fig 69: G-Band Pixels

Fig 70: G-Band statistical

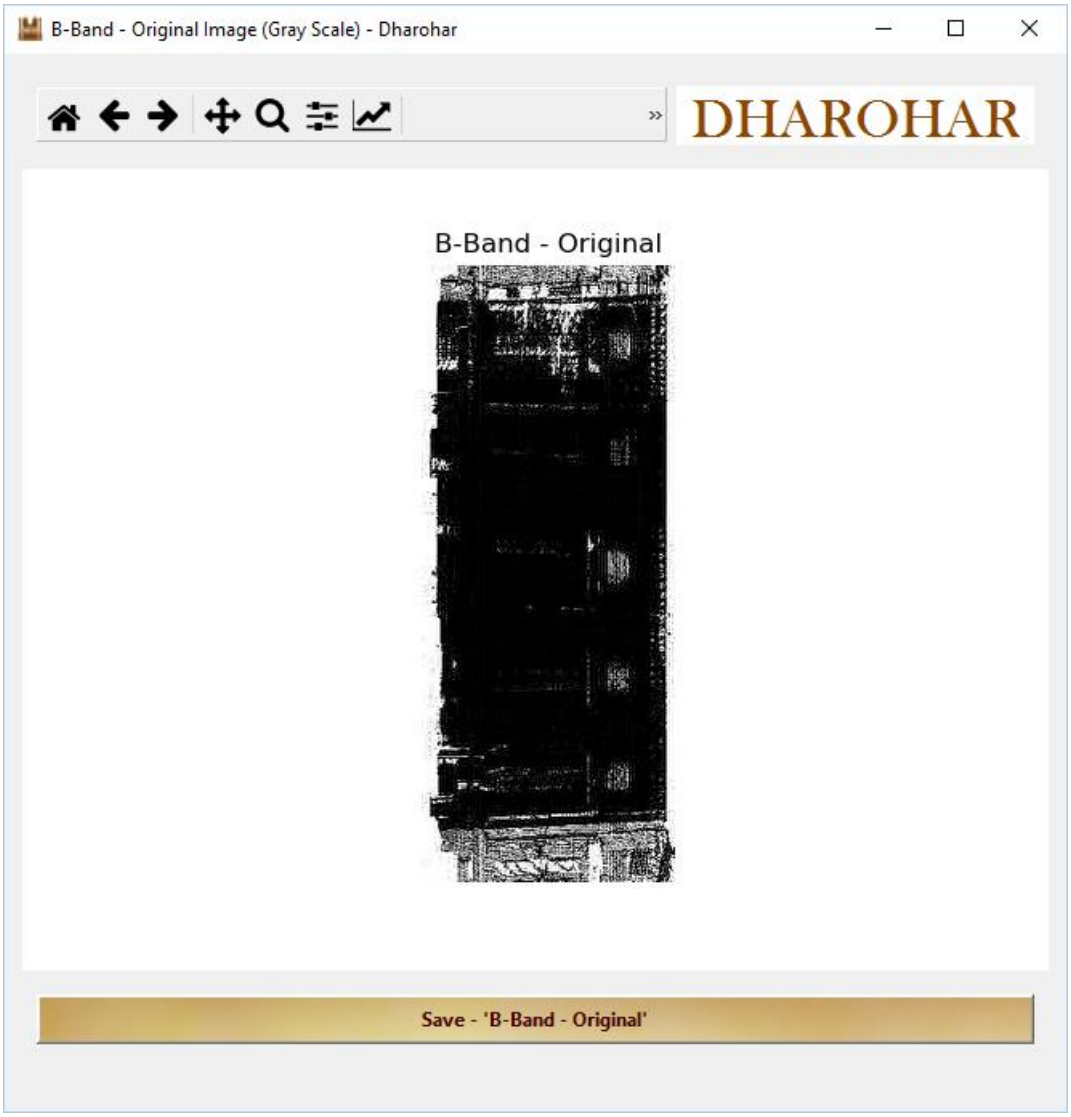


Fig 71: B-Band- Original Image

Fig 72: B-Band Pixels

Fig 73: B-Band statistical

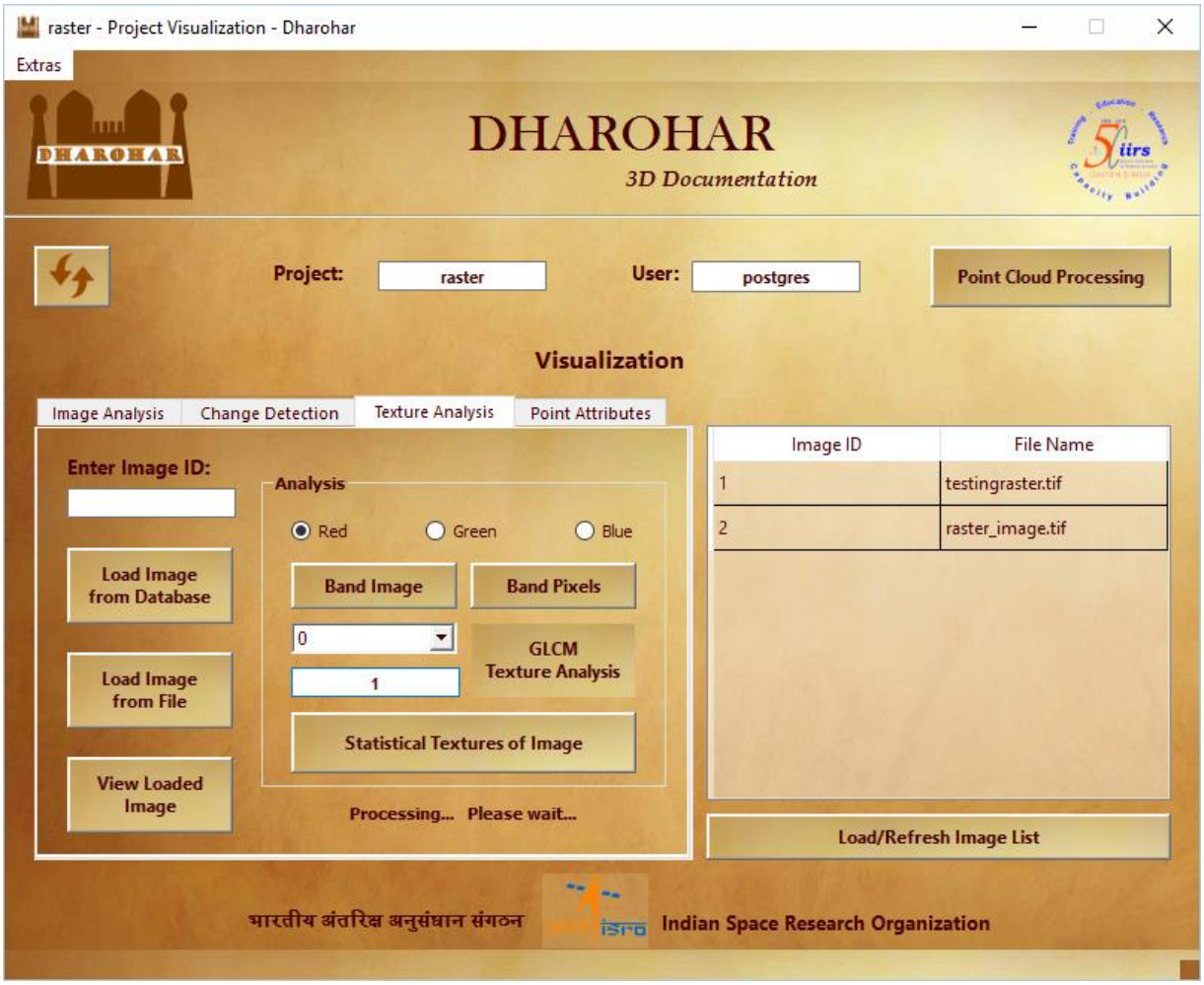


Fig 74: Statistical texture of Image – R-Band

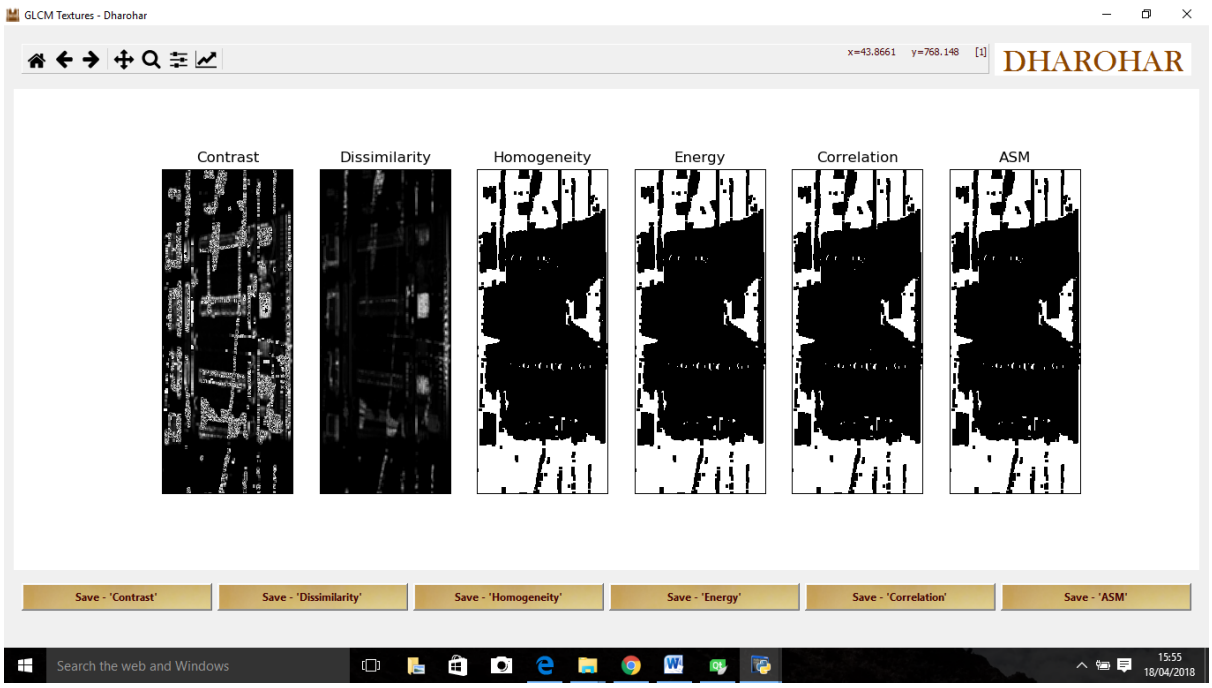


Fig 75: GLCM output – R-Band

Fig 76: Statistical texture of Image – G-Band

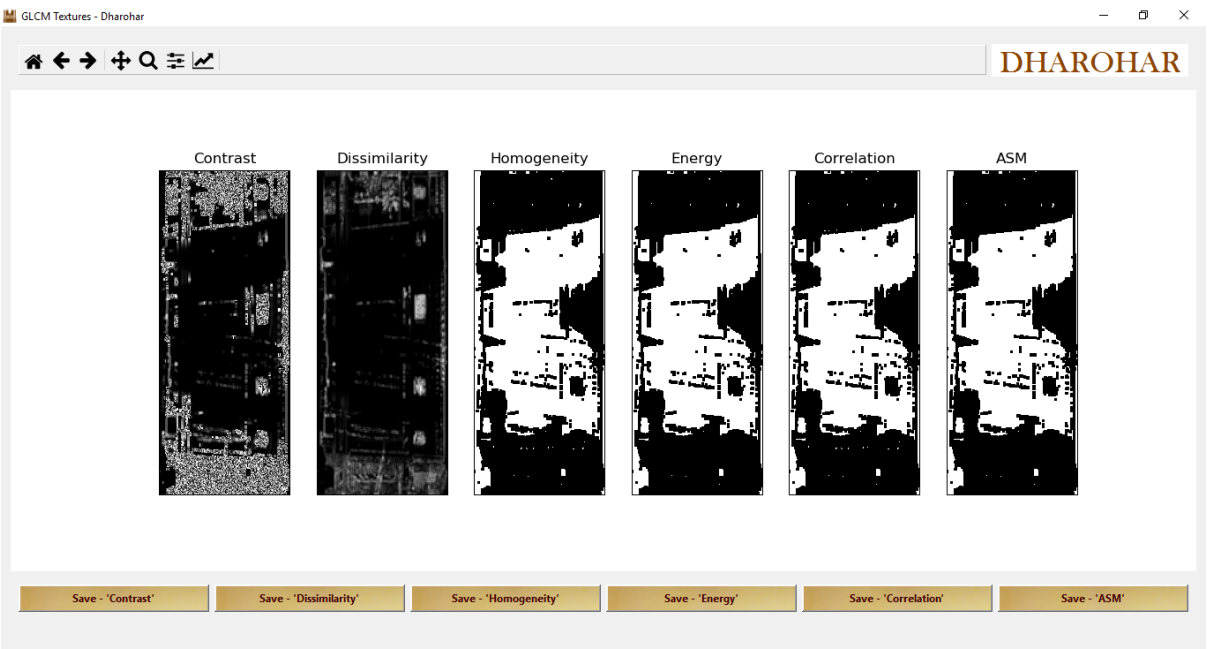


Fig 77: GLCM output – G-Band

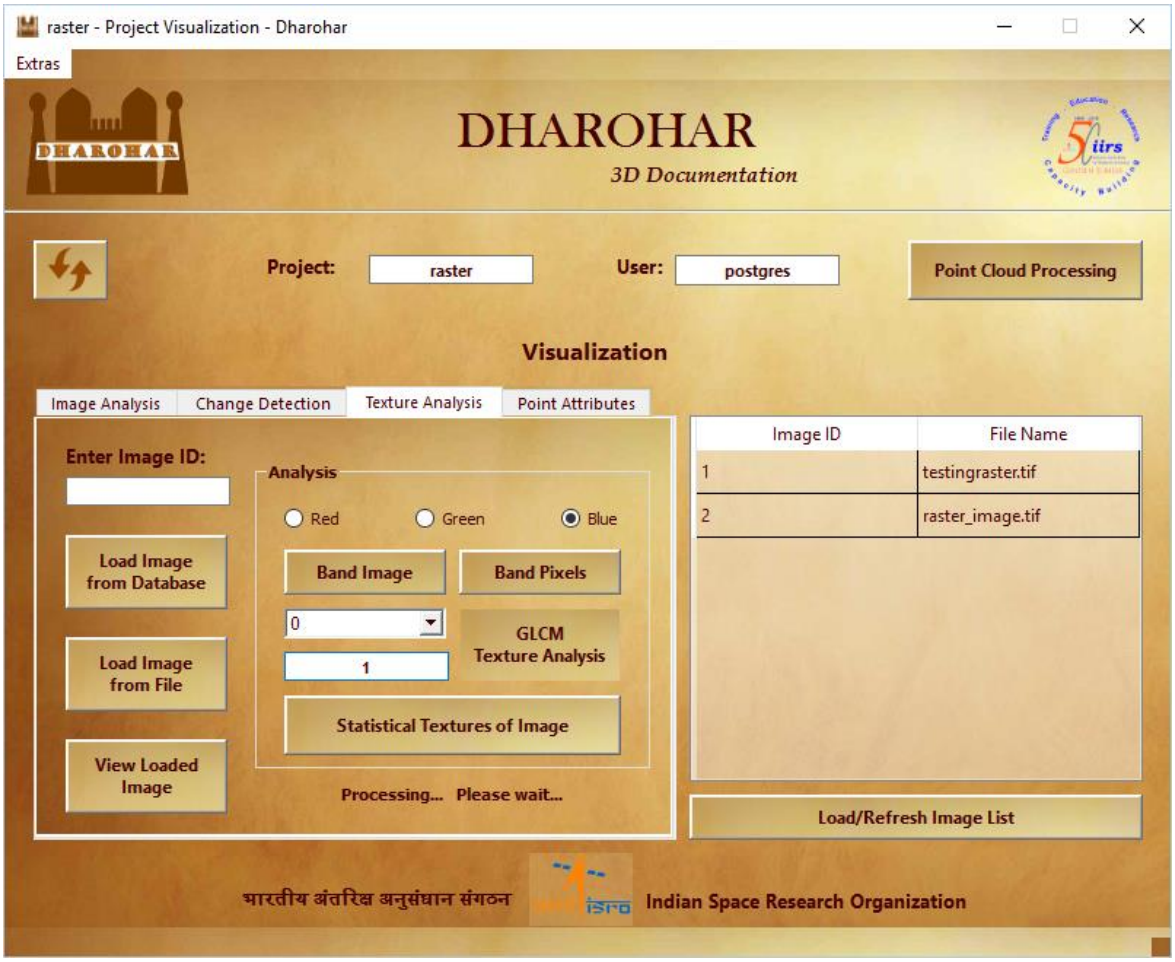


Fig 78: Statistical texture of Image – B-Band

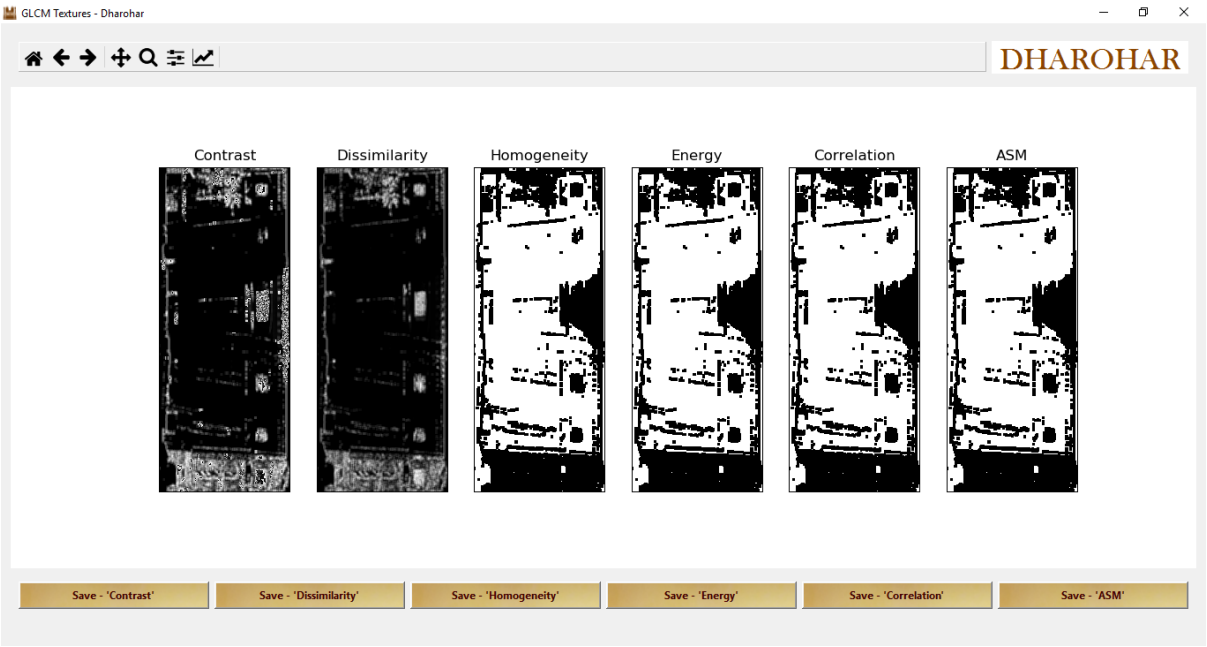


Fig 79: GLCM output – B-Band

Fig 79: Point attributes

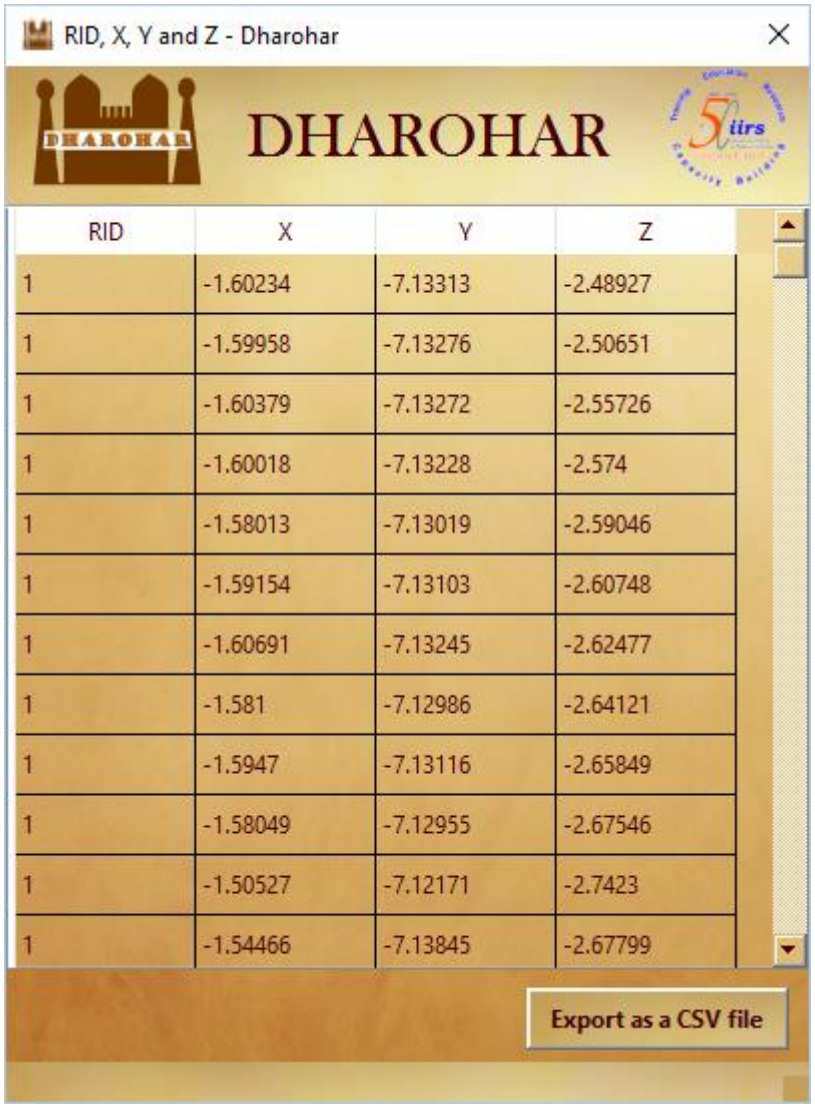


Fig 80: X,Y and z attribute value

Fig 81: Radius, Depth and Theta

Fig 82: Radius, Depth and Theta value

DATABASE DESIGN:

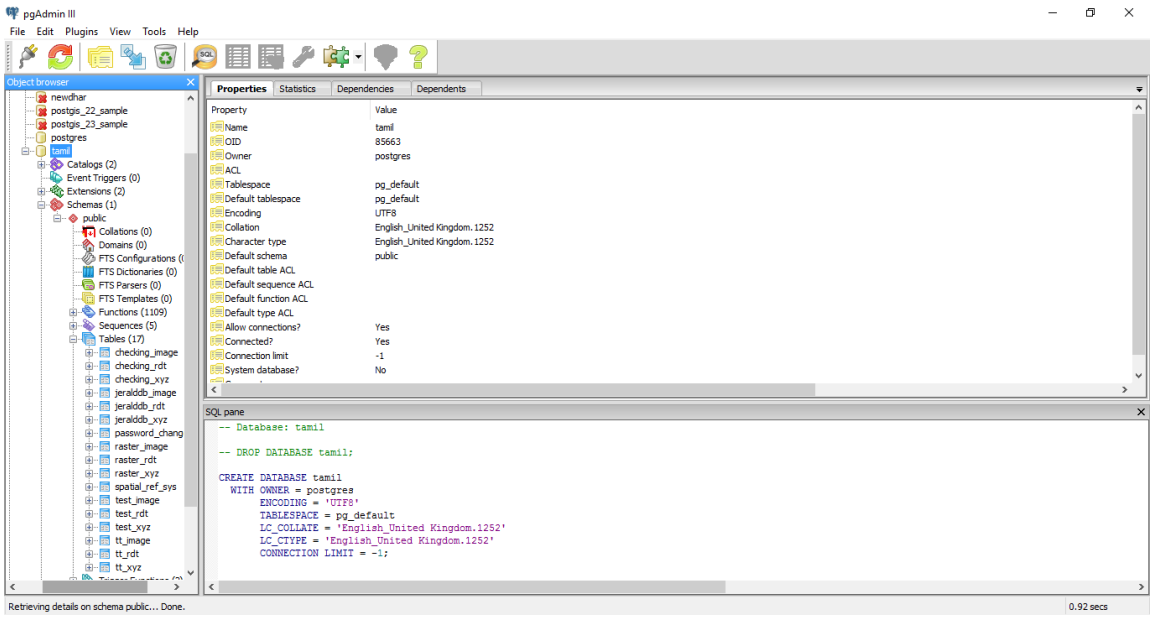


Fig 83: Database

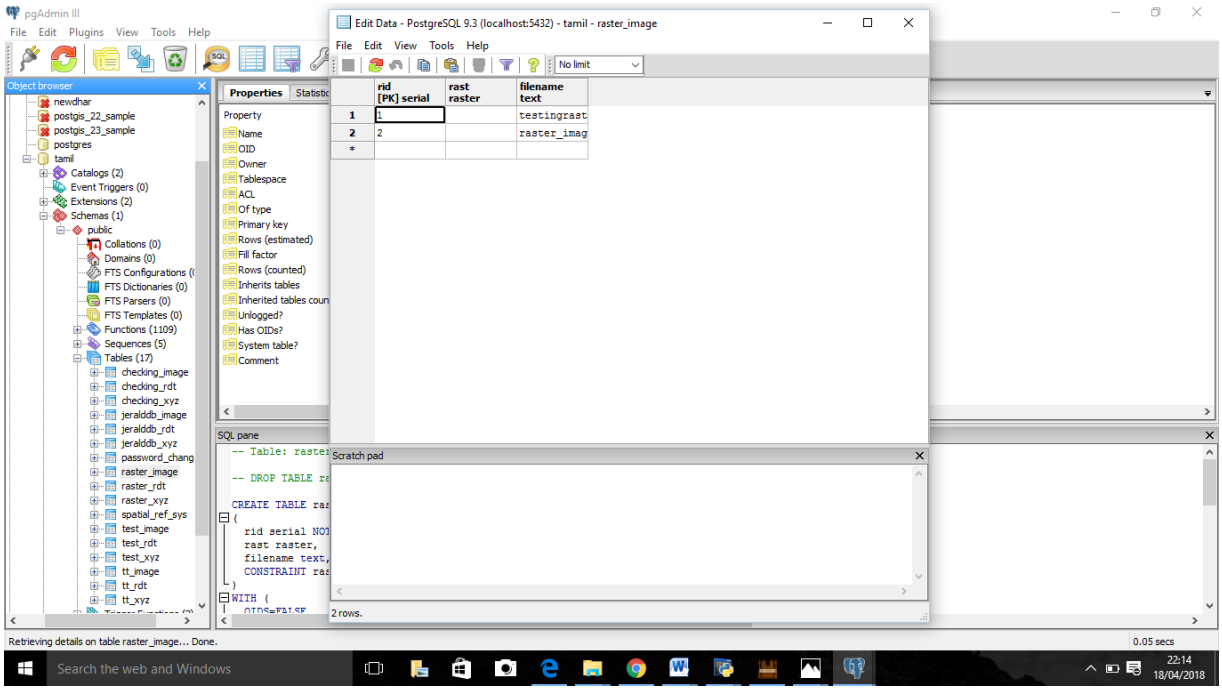


Fig 84: Table containing Raster Image

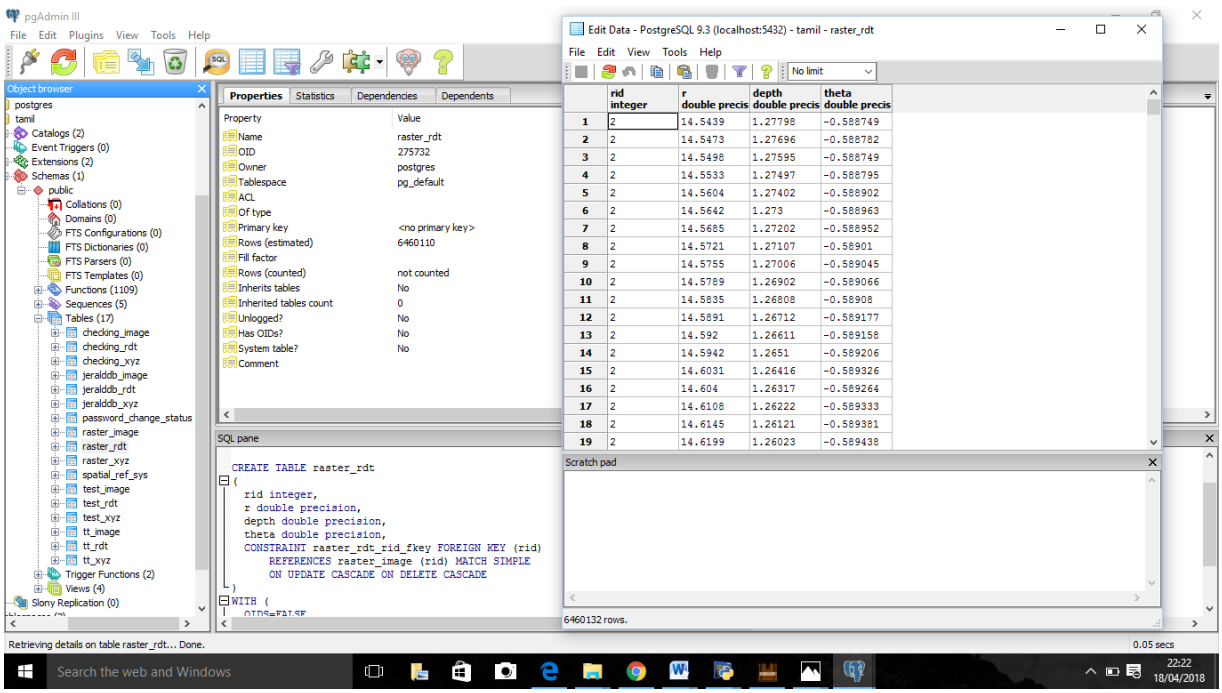


Fig 85: Table containing radius, depth and theta values

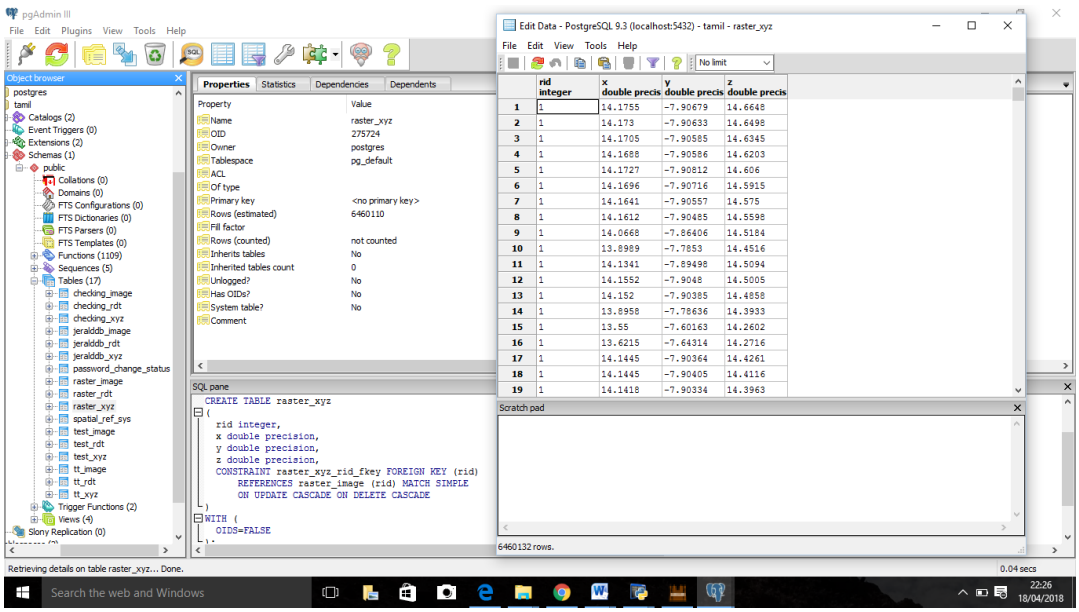


Fig 86: Table containing x,y,z values