



# Geo-Registration of Satellite Images

Bhaskaracharya Institute of Space Applications and Geoinformatics



Rajath S (2012B5A7589P)

July 13, 2014

Birla Institute of Technology and Science, Pilani

## Acknowledgments

I am highly indebted to BISAG for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support. I would like to express my gratitude towards Mr. Viraj Choksi, Mr. Manoj Pandey and Mr. Bharath for their kind co-operation and encouragement which helped me in completion of this report. I am also grateful to our Instructor, Mr. Sudeep Pradhan and co-instructor, Mr. Sumit Prajapati and would like to thank the industry persons for giving me such attention and time. The contribution of peer group, as reviewers is indispensable in making this report.

## PS Details

Station	Bhaskaracharya Institute of Space Applications and Geoinformatics
Duration	56 days
Start	23rd May, 2014
End	17th May, 2014
Project Title	Automatic Georeferencing of Satellite Images and Image Mosaicing
Project Team Members	Kartik Pitale (2012A7PS085P) , Nirant Kasliwal (2012C6PS694P)
Instructor	Mr. Sudeep Kr Pradhan
Student Co-Instructor	Akanksha Patnaik (2011B5A3359G)
Project Guide	Mr. Sumit Prajapati

## Abstract

The report examines the process of Geo-referencing of satellite images in detail. The technical details as well as the implementation is discussed along with a brief overview of the software tools used in this process. It attempts to clearly and concisely explain the mathematics involved in the process of image transformation and registration.

Further, it proposes a solution as to how the process of Geo-referencing can be automated to reduce human effort and error in the process. It also discusses how the currently existing soft-wares can be modified to automate the process of geo-referencing.

Apart from Geo-referencing, the report also has a discussion on Image Stitching. Two implementations of Image Stitching is explained. The fully automatic image stitching relies on pattern matching algorithms to find correspondences between the two images, whereas the semi-automatic algorithm requires user input to find reliable correspondences between the images.

In the end, our work during PS is demonstrated using screenshots and code samples. The features and specifications of our software is fully explained with regard to its utility to BISAG. Step by step explanation are presented in order to illustrate the algorithm used in the process. Relevant technical details such as the pattern matching algorithms are elaborated. Rationale behind the choice of different algorithms is explained. Finally, the software is evaluated based on the time and memory consumption.

# Contents

<b>I. Theory</b>	<b>6</b>
<b>1. Introduction: Registration and Stitching</b>	<b>7</b>
1.1. Image Registration . . . . .	7
1.2. Image Stitching . . . . .	8
1.3. Geo-referencing . . . . .	8
<b>2. Transformation Models</b>	<b>9</b>
2.1. 2D Planar transformations . . . . .	9
2.2. Non-linear Transformations . . . . .	11
<b>3. Feature Matching</b>	<b>13</b>
<b>4. Least Squares Approximation</b>	<b>17</b>
<b>5. Interpolation</b>	<b>18</b>
<b>II. Survey of Existing Softwares</b>	<b>21</b>
<b>6. Geo-spatial Data Abstraction Library</b>	<b>22</b>
<b>7. Quantum GIS</b>	<b>23</b>
7.1. Geo-referencer plugin . . . . .	23
<b>8. Auto-GR Toolkit</b>	<b>26</b>
8.1. Auto-GR SIFT . . . . .	26
8.2. GeoRef Filtering . . . . .	26
<b>9. NASA Vision Workbench</b>	<b>28</b>
<b>10. OpenCV</b>	<b>30</b>

<b>11. Qt</b>	<b>31</b>
<b>III. Project</b>	<b>32</b>
<b>12. Project Details</b>	<b>33</b>
12.1. Problem Statement . . . . .	33
12.2. Problem Description . . . . .	33
12.3. Methodology and Implementation . . . . .	33
12.3.1. Automatic Geo-referencing . . . . .	33
12.3.2. Image Stitching . . . . .	36
12.3.2.1. Automatic . . . . .	36
12.3.2.2. Semi-Automatic . . . . .	37
<b>IV. References</b>	<b>42</b>

Part I.

Theory

# 1. Introduction: Registration and Stitching

## 1.1. Image Registration

Image registration is the process of transforming different sets of data into one coordinate system. Data may be multiple photographs, data from different sensors, times, depths, or viewpoints. Registration is necessary in order to be able to compare or integrate the data obtained from these different measurements.

This process involves designating one image as the reference (also called the reference image or the fixed image), and applying geometric transformations to the other images so that they align with the reference. A geometric transformation maps locations in one image to new locations in another image. The step of determining the correct geometric transformation parameters is key to the image registration process.

Images can be misaligned for a variety of reasons. Commonly, the images are captured under variable conditions that can change camera perspective. Misalignment can also be the result of lens and sensor distortions or differences between capture devices. Two major types of distortions are distinguished. The first type are those which are the source of misregistration. Distortions which are the source of misregistration determine the transformation class which will optimally align the two images. The second type of distortion are those which are not the source of misregistration. This type usually affects the intensity values but they may also be spatial. Distortions of this type are not removed by registration but they make registration more difficult because exact match is no longer available.

Image registration is often used as a preliminary step in other image processing applications. For example, you can use image registration to align satellite images or to align medical images captured with different diagnostic modalities (MRI and SPECT). Image registration allows you to compare common features in different images. For example, you might discover how a river has migrated, how an area became flooded, or whether a tumor is visible in an MRI or SPECT image.

The essential problems in Image Alignment are: feature detection, feature description, feature matching, correspondence computation and interpolation.

## **1.2. Image Stitching**

Image Stitching is the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image. Commonly performed through the use of computer software, most approaches to image stitching require nearly exact overlaps between images and identical exposures to produce seamless results.

The first step in Image Stitching is Image Registration. Much of the theory related to Image Stitching is therefore same as Image Registration. Mosaicing is the process of making a panorama after images are aligned.

The details of Image Stitching and Registration are explained in detail in the later chapters.

## **1.3. Geo-referencing**

Georegistration is the alignment of an unreferenced image with a geodetically (latitude, longitude, and elevation based) calibrated reference image. In general, this process aligns two images (satellite is the most common, but is not limited to satellite images) and correlating the images to a physical location by examining a set of distinguishable points.

In general, geo-referencing is registering a satellite image using earth as the reference image. All the theory relevant to Image Registration is applicable to geo-referencing as well. Georeferencing does not involve pattern matching as usually latitude and longitude details of some control points are available. Thus, the process of geo-referencing only involves finding a transformation using the provided control points.



## 2. Transformation Models

Before we can register and align images, we need to establish the mathematical relationships that map pixel coordinates from one image to another. A variety of such parametric motion models are possible, from simple 2D transforms, to planar perspective models, 3D camera rotations, lens distortions, and the mapping to non-planar (e.g., cylindrical) surfaces.

### 2.1. 2D Planar transformations

The simplest transformations occur in the 2D plane and are illustrated in Figure 2.1.

**Translation.** 2D translations can be written as  $\bar{\mathbf{x}} = \mathbf{x} + \mathbf{t}$ .

**Rotation+Translation.** This transformation is also known as 2D rigid body motion or the 2D Euclidean transformation (since Euclidean distances are preserved). It can be written as  $\bar{\mathbf{x}} = \mathbf{R}\mathbf{x} + \mathbf{t}$ , where  $\mathbf{R}$  is an orthonormal rotation matrix with  $\mathbf{R}\mathbf{R}^T = \mathbf{I}$  and  $|\mathbf{R}| = 1$ .

**Rotation(scaled)+Translation.** Also known as the similarity transform, this transform can be expressed as  $\bar{\mathbf{x}} = s\mathbf{R}\mathbf{x} + \mathbf{t}$  where  $s$  is an arbitrary scale factor.

**Affine.** The affine transform is written as  $\bar{\mathbf{x}} = \mathbf{A}\mathbf{x}$ , where  $\mathbf{A}$  is an arbitrary  $2 \times 3$  matrix. Parallel lines remain parallel under affine transformations.

**Projective.** This transform, also known as a perspective transform or homography. They are of the form:

$$\bar{x} = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}; \bar{y} = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

Perspective transformations preserve straight lines.

Figure 2.1.: Basic set of 2D planar transformations

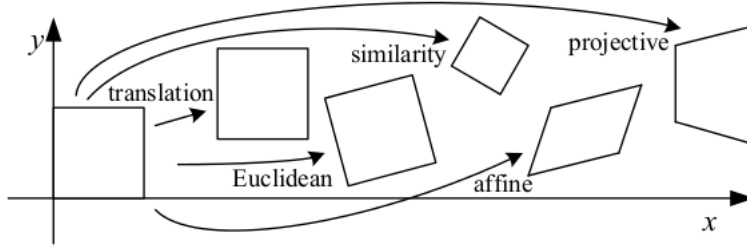


Figure 2.2.: Hierarchy of 2D coordinate transformations. #DOF denotes the number of determining factors.






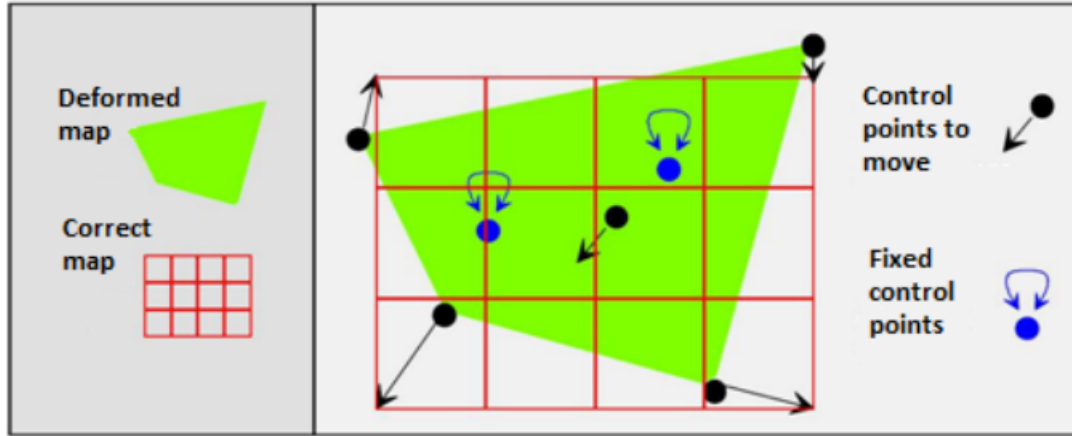
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation + ...	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths + ...	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles + ...	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism + ...	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	

Figure 2.3.: The Rubber-Sheeting transformation considers a deformed map as a sheet made of rubber which is stretched to some nails representing the correcting points.



## 2.2. Non-linear Transformations

Non-linear transformations involve non-linear terms in the correspondence equation. Different non-linear transformations are used to stretch images locally in order to align them with the reference image. Such transformations contain terms of order higher than 1 such as  $xy, x^2, y^2, x^3y$  etc.

**Rubber-Sheeting** The Rubber-Sheeting transformation is based on geometric and mathematical theories similar to those of the Homography transformation. This technique is called "Rubber-Sheeting" because figuratively it considers a deformed map as a sheet made of rubber which is stretched to some fixed nails representing the correcting reference points. This also means that the Rubber-Sheeting algorithm transforms a quadrilateral from one reference system into the corresponding quadrilateral in another reference system. The Rubber-Sheeting calculation is based on the following formulae:

$$\bar{x} = axy + bx + cy + d$$

$$\bar{y} = exy + fx + gy + e$$

**LensDistortions** When images are taken with wide-angle lenses, it is often necessary to model lens distortions such as radial distortion. The radial distortion model says that coordinates in the observed images are displaced away or towards the image

center by an amount proportional to their radial distance. The simplest radial distortion models use low-order polynomials such as:

$$\bar{x} = x(1 + k_1r + k_2r^2)$$

$$\bar{y} = y(1 + k_3r + k_4r^2)$$

where  $r^2 = x^2 + y^2$ .

Various other transformations exist which are not out of the scope of this document. Transformations which warp an image onto a cylindrical or a spherical surface exist which are essentially non-linear transformations.

These transformations are used to warp the image to align it with the reference image. Transformations can be determined using the control points, either entered manually by the user or found using feature detection.

### 3. Feature Matching

Image Registration involves finding matching control points in the given image and the reference image. This is done using feature detection algorithms. The common feature of feature detection algorithms is to find key points in images which are good points for matching, then feature descriptors are built for each point describing the neighborhood of each keypoint and then the keypoints from both reference image and the given image are matched to find correspondence function modeling any of the transformations described above.

Popular Feature detection algorithms that are used in our project are discussed below.

**SIFT** *Scale-Invariant Feature Transform* is an algorithm to compute and describe local features in images. The SIFT algorithm takes an image and transforms it into a collection of local feature vectors. Each of these feature vectors is supposed to be distinctive and invariant to any scaling, rotation or translation of the image. In the original implementation, these features can be used to find distinctive objects in different images and the transform can be extended to match faces in images. The features have been shown to be invariant to image rotation and scale and robust across a substantial range of affine distortion, addition of noise, and change in illumination. The approach is efficient on feature extraction and has the ability to identify large numbers of features. Important Steps in SIFT are: *Creating the Difference of Gaussian Pyramid, Extrema Detection, Noise Elimination, Orientation Assignment, Descriptor Computation*. Difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different  $\sigma$ , let it be  $\sigma$  and  $k\sigma$ . This process is done for different octaves of the image in Gaussian Pyramid. It is represented in Figure 3.1. Once this DoG are found, images are searched for local extrema over scale and space. For example, one pixel in an image is compared with its 8 neighbors as well as 9 pixels in next scale and 9 pixels in previous scales. If it is a local extrema, it is a potential keypoint. It basically means that keypoint is best represented in that scale. Once potential keypoints locations are found, they have to be refined to get more accurate results. Taylor series expansion of scale space is used to get more accurate location of extrema, and if the intensity at this

Figure 3.1.: Gaussian Pyramid

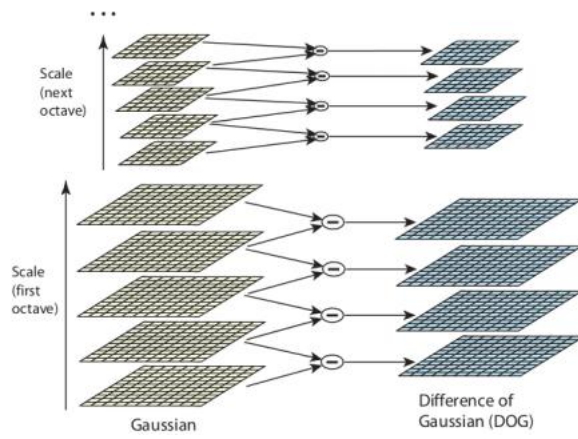
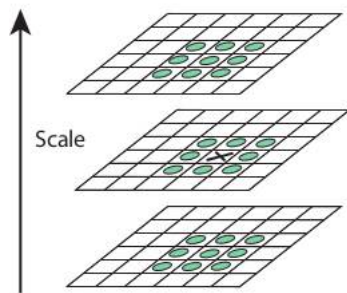


Figure 3.2.: Local extrema in DoGs



extrema is less than a threshold value, it is rejected. DoG has higher response for edges, so edges also need to be removed. If this ratio is greater than a threshold called `edgeThreshold`, that keypoint is discarded. So it eliminates any low-contrast keypoints and edge keypoints and what remains is strong interest points. Now an orientation is assigned to each keypoint to achieve invariance to image rotation. A neighborhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created (It is weighted by gradient magnitude and gaussian-weighted circular window with  $\sigma$  equal to 1.5 times the scale of keypoint) The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contribute to stability of matching. Now keypoint descriptor is created. A 16x16 neighborhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form keypoint descriptor. In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.

**SURF** *Speeded-Up Robust Features* is a modification of the SIFT algorithm. In SIFT, Laplacian of Gaussian is approximated with Difference of Gaussian for finding scale-space. SURF goes a little further and approximates LoG with Box Filter. One big advantage of this approximation is that, convolution with box filter can be easily calculated with the help of integral images and it can be done in parallel for different scales. For orientation assignment, SURF uses wavelet responses in horizontal and vertical direction for a neighborhood of size 6s. Adequate gaussian weights are also applied to it. Then they are plotted in a space as given in below image. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of angle 60 degrees. Interesting thing is that, wavelet response can be found out using integral images very easily at any scale. For feature description, SURF uses Wavelet responses in horizontal and vertical direction. In short, SURF adds a lot of features to improve the speed in every step. Analysis shows it is 3 times faster than SIFT while performance is comparable to SIFT. SURF is good at handling images with blurring and rotation, but not good at handling viewpoint change and illumination change.

Once the features are detected and descriptors are built as explained above, they have

to be matched to find correspondences in images. The most popular feature matching algorithms are explained below:

**Brute-Force** Brute-Force matcher is simple. It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation and the closest one is returned. It is a naive but effective approach to feature matching. It is time consuming and is not preferred if the number of features to be matched is too high.

**FLANN** FLANN (Fast Library for Approximate Nearest Neighbors) is a library that contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. We use the k-nearest neighbor search algorithm to find the matches in set of keypoints. k-nearest neighbor search identifies the top k nearest neighbors to the query. This technique is commonly used in predictive analytics to estimate or classify a point based on the consensus of its neighbors. k-nearest neighbor graphs are graphs in which every point is connected to its k nearest neighbors.

Once the points are matched, a transformation can be approximated using the least square technique.



## 4. Least Squares Approximation

Least square is an optimization technique that can be used to fit a data set to any given function. The method of least squares is a standard approach to the approximate solution of over-determined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation. It minimizes the quantity described by the following equation:

$$\chi^2 = \sum_{n=1}^N (y_n - f(x_n))^2$$

The most important application is in data fitting. The best fit in the least-squares sense minimizes the sum of squared residuals, a residual being the difference between an observed value and the fitted value provided by a model.

Multivariate least squares approximation is the extension of least squares technique to include equations with more than one variable. This technique is more complicated because

This technique is used to fit a transformation model to the given dataset of matching keypoints.

## 5. Interpolation

Image interpolation occurs in all digital photos at some stage. It happens anytime you resize or remap (distort) your image from one pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels, whereas remapping can occur under a wider variety of scenarios: correcting for lens distortion, changing perspective, and rotating an image.

Even if the same image resize or remap is performed, the results can vary significantly depending on the interpolation algorithm. It is only an approximation, therefore an image will always lose some quality each time interpolation is performed.

Interpolation works by using known data to estimate values at unknown points. Image interpolation works in two directions, and tries to achieve a best approximation of a pixel's color and intensity based on the values at surrounding pixels. Figure 5.1 illustrates how resizing / enlargement works. The results quickly deteriorate the more you stretch an image, and interpolation can never add detail to your image which is not already present. Interpolation also occurs each time you rotate or distort an image.

Common interpolation algorithms can be grouped into two categories: adaptive and non-adaptive. Adaptive methods change depending on what they are interpolating (sharp edges vs. smooth texture), whereas non-adaptive methods treat all pixels equally. Non-adaptive algorithms include: nearest neighbor, bilinear, bicubic, spline, sinc, lanczos and others. Depending on their complexity, these use anywhere from 0 to 256 (or more) adjacent pixels when interpolating. The more adjacent pixels they include, the more accurate

Figure 5.1.: Image Interpolation during resizing

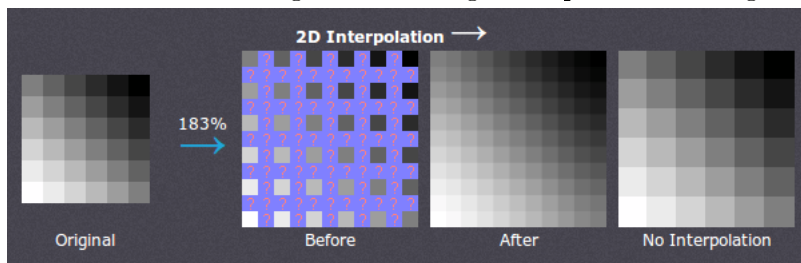
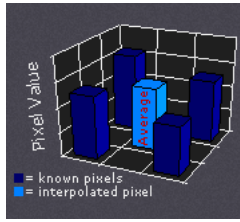


Figure 5.2.: Bilinear Interpolation



they can become, but this comes at the expense of much longer processing time. These algorithms can be used to both distort and resize a photo.

Adaptive algorithms include many proprietary algorithms in licensed software such as: Qimage, PhotoZoom Pro, Genuine Fractals and others. Many of these apply a different version of their algorithm (on a pixel-by-pixel basis) when they detect the presence of an edge — aiming to minimize unsightly interpolation artifacts in regions where they are most apparent. These algorithms are primarily designed to maximize artifact-free detail in enlarged photos, so some cannot be used to distort or rotate an image.

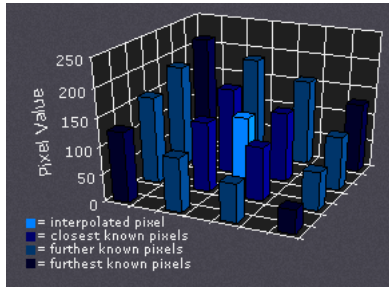
**NN** Nearest neighbor is the most basic and requires the least processing time of all the interpolation algorithms because it only considers one pixel — the closest one to the interpolated point. This has the effect of simply making each pixel bigger.

**Bilinear** Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value. This results in much smoother looking images than nearest neighbor. Figure 5.2 is for a case when all known pixel distances are equal, so the interpolated value is simply their sum divided by four.

**Bicubic** Bicubic goes one step beyond bilinear by considering the closest 4x4 neighborhood of known pixels — for a total of 16 pixels. Since these are at various distances from the unknown pixel, closer pixels are given a higher weighting in the calculation. Bicubic produces noticeably sharper images than the previous two methods, and is perhaps the ideal combination of processing time and output quality. For this reason it is a standard in many image editing programs (including Adobe Photoshop), printer drivers and in-camera interpolation.

**Spline** There are many other interpolators which take more surrounding pixels into consideration, and are thus also much more computationally intensive. These algorithms include spline, and retain the most image information after an interpolation. They are therefore extremely useful when the image requires multiple rotations /

Figure 5.3.: Bicubic Interpolation



distortions in separate steps. However, for single-step enlargements or rotations, these higher-order algorithms provide diminishing visual improvement as processing time is increased.

These algorithms are used when the image is transformed to align perfectly with the reference image. Transformations involve rotation, shear, projection etc, where interpolation is necessary to fill the blank data.

Part II.

## Survey of Existing Softwares

## 6. Geo-spatial Data Abstraction Library

Satellite images are stored either in raster or vector format in various file formats like GeoTIFF, L-3, etc. GDAL (Geospatial Data Abstraction Library) is a library for reading and writing raster geospatial data formats. As a library, it presents a single abstract data model to the calling application for all supported formats. Several software programs use the GDAL libraries to allow them to read and write multiple GIS formats. Several utility programs are distributed with GDAL. The most important ones are:

- `gdalinfo` - Report information about a file.
- `gdalwarp` - Warp an image into a new coordinate system.
- `gdal_merge.py` - Build a quick mosaic/stitching from a set of images.
- `gdaltransform` - Transform coordinates.
- `gdalcompare.py` - Compare two images and report on differences.

## 7. Quantum GIS

QGIS (also known as "Quantum GIS") is a cross-platform free and open source desktop geographic information systems (GIS) application that provides data viewing, editing, and analysis capabilities. Similar to other software GIS systems QGIS allows users to create maps with many layers using different map projections. Maps can be assembled in different formats and for different uses. QGIS allows maps to be composed of raster or vector layers. Typical for this kind of software the vector data is stored as either point, line, or polygon-feature. Different kinds of raster images are supported and the software can perform georeferencing of images.

QGIS provides integration with other open source GIS packages, including PostGIS, GRASS, and MapServer to give users extensive functionality. Plugins, written in Python or C++, extend the capabilities of QGIS. There are plugins to geocode using the Google Geocoding API, perform geoprocessing (fTools) similar to the standard tools found in ArcGIS, interface with PostgreSQL/PostGIS, SpatiaLite and MySQL databases, and use Mapnik as a map renderer.

### 7.1. Geo-referencer plugin

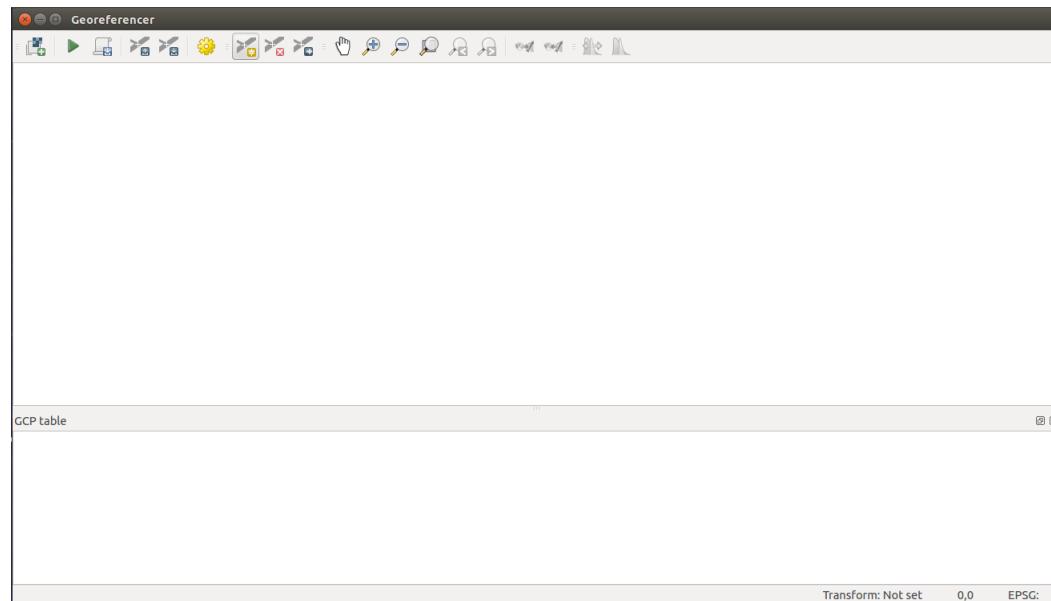
Quantum GIS (QGIS) has a very powerful image georeferencing module. The Georeferencer Plugin is a tool for generating world files for rasters. It allows us to reference rasters to geographic or projected coordinate systems by creating a new GeoTiff or by adding a world file to the existing image. The basic approach to georeferencing a raster is to locate points on the raster for which we can accurately determine their coordinates.

The usual procedure for georeferencing an image involves selecting multiple points on the raster, specifying their coordinates, and choosing a relevant transformation type. Based on the input parameters and data, the plugin will compute the world file parameters. The more coordinates we provide, the better the result will be.

This plugin implements all the algorithms required to transform the unregistered image based on the control points to produce the registered image.

The available transformation algorithms as explained in the theory section are:

Figure 7.1.: QGIS GeoReferencer Plugin



- The Linear algorithm is used to create a world-file, and is different from the other algorithms, as it does not actually transform the raster. This algorithm likely won't be sufficient if we are dealing with scanned material.
- The Helmert transformation performs simple scaling and rotation transformations.
- The Polynomial algorithms 1-3 are among the most widely used algorithms introduced to match source and destination ground control points. The most widely used polynomial algorithm is the second order polynomial transformation, which allows some curvature. First order polynomial transformation (affine) preserves collinearity and allows scaling, translation and rotation only.
- The Thin plate spline (TPS) algorithm is a more modern georeferencing method, which is able to introduce local deformations in the data. This algorithm is useful when very low quality originals are being georeferenced.
- The Projective transformation is a linear rotation and translation of coordinates.

The type of resampling we choose will likely depending on the input data and the ultimate objective of the exercise. If we don't want to change statistics of the image, we might want to choose 'Nearest neighbor', whereas a 'Cubic resampling' will likely provide a more smoothed result.

It is possible to choose between five different resampling methods as explained earlier.



1. Nearest neighbor
2. Linear
3. Cubic
4. Cubic Spline
5. Lanczos

## 8. Auto-GR Toolkit

AutoGR-Toolkit is a set of 4 scripts(GGRAB, AuttoGR-Sift, GeoRef Filtering, Geo-Tiff Converter) and 2 algorithm libraries (ASift andGDAL) to assist the user in georeferencing one image on another one according to the specific geographical projection in an easy, fast and accurate way.

A few modules relevant to our project is discussed below:

### 8.1. Auto-GR SIFT

AutoGR-SIFT is the core and the most interesting part of the Toolkit. This script“prepares” two input images (essentially byscaling and saving a copy of them in PNG format) to be processed by the SIFT algorithm (Scale-invariant feature transform) and converts the output for a GIS environment.

It extracts keypoint from two images(in PNG, JPG or TIFF/GeoTIFF format) to provide a "feature description" of the object depicted in each of them. Such descriptions can then be used to locate the same object in both images. Once the relation between x and y coordinates of the key points in both images has been found, a structured text file and visual preview of (horizontal and vertical) of matching points connected by vectors are created. The conversion of common points in first and second image from relative XY pixel coordinate into geographical Easting Northing information, according to the input projection, is carried out by AutoGR without any user interaction.

Profiting of the powerful GDAL library, an automated rectification of the second image is attempted and the result presented for the evaluation of the user. Selected matches are then fed to gdalwarp to obtain the transformation.

### 8.2. GeoRef Filtering

The hundreds of points (usually) produced in few seconds by AutoGR-SIFT often need to be decimated in order to be processed by most of the GIS applications. Indeed, to load more than 200 matching points into the georeferencing utility of QGIS may result

in software crashes due to lack of memory. For this reason the user is provided with the possibility to filter the points to a lower number. This function is provided by this software.

## 9. NASA Vision Workbench

The NASA Vision Workbench (VW) is a general purpose image processing and computer vision library developed by the Autonomous Systems and Robotics (ASR) Area in the Intelligent Systems Division at the NASA Ames Research Center.

The Vision Workbench was implemented in the C++ programming language and makes extensive use of C++ templates and generative programming techniques for conciseness of expression, efficiency of operation, and generalization of implementation.

While substantial functionality is implemented in the Vision Workbench, the goal of the library is not to provide advanced cutting-edge image processing or computer vision capabilities. Rather, the intent is to provide a solid efficient foundation implementing well known techniques and a common framework for doing advanced research and development in collaboration with others.

C++ templates are used extensively throughout the Vision Workbench. This enables compile time expansion and optimization of expressions, and in general terms, compile-time computation of checks and quantities that are typically performed at run-time with more traditional object oriented approaches. In addition, extensive parameterization by type of functions and classes together with careful design has the well known advantage of enabling code reuse across a variety of applications.

The Vision Workbench has a hierarchical layered architecture where each implementation layer is composed of a set of modules that are independent of modules in higher level layers. Dependencies between modules in a given layer and circular dependencies (direct or indirect) between modules are not permitted.

The following modules are provided:

1. Cartography: transformation between map projections, geo-referenced file IO (Geo-TIFF, etc.).
2. FileIO: reading and writing JPEG, PNG, TIFF, PDS, OpenEXR files, image paging support for large files.
3. Image: convolution — convolve images with linear filters, algebra — image addition, subtraction, and multiplication by a scalar, math — perform per pixel math

operations on images (e.g., abs, hypot, sqrt, pow, exp, log, sin, cos, asin, acos, atan2, etc.), transformations — translation, rotation, scale, resampling, arbitrary warps, image statistics.

4. Math: vector, matrix, quaternion and linear algebra, optimization — levenberg-Marquardt, conjugate gradient, homographies — estimation of relations between geometric entities, bounding boxes — Determination of geometric extent.
5. Mosaic: assembling large image composites from many source images, multi-band blending.
6. Stereo: correlating stereo pairs of images, stereo camera models for 3D reconstruction, outlier rejection.

## 10. OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel Russia research center in Nizhny Novgorod, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing.

It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

## 11. Qt

Qt is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI) (in which cases Qt is classified as a widget toolkit), and also used for developing non-GUI programs such as command-line tools and consoles for servers.

Qt uses standard C++ but makes extensive use of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform application programming interface (API) for file handling.

Starting with Qt 4.0 the framework was split into individual modules. With Qt 5.0 the architecture was modularized even further. Qt is now split into essential and add-on modules.

PyQt is a python Qt module which can be used to create GUIs using python. Qt Creator, a cross-platform IDE for C++ and QML. Qt Designer's GUI layout/design functionality is integrated into this relatively new IDE, although Qt Designer can still be called as a standalone tool.

Part III.

Project



## 12. Project Details

### 12.1. Problem Statement

To automate the selection of control points in the process of georeferencing satellite images and to stitch two georeferenced images with overlap area to form a mosaic.

### 12.2. Problem Description

Satellite images provided to BISAG are not georeferenced. Along with the satellite images, a text file containing the geographic locations of the four corners of the image is provided. Georeferencing is done using a software called ArcGIS by manually selecting the four corners as control points. This process is prone to human errors and can be easily automated to reduce the human effort. In the case of Satellite Images, even a small error can have a large impact as each pixel represents 2.5m on earth. The second part of the project is to create a seamless mosaic of two overlapping satellite images. Stitching of satellite images is easier because of the geographic data available along with the image. An example is shown in Figure 12.3.

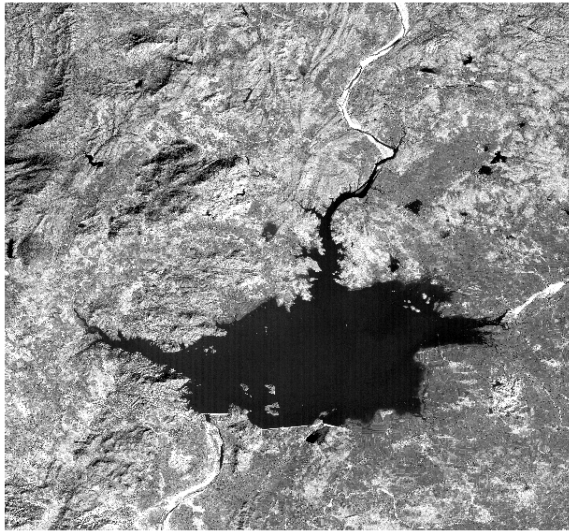
### 12.3. Methodology and Implementation

#### 12.3.1. Automatic Geo-referencing

The steps involved are:

1. Reading a raster image file.
2. Reading the text file associated with it.
3. Searching for Upper Left, Upper Right, Lower Left and Lower Right coordinates from the text file.
4. Using Least Squares technique to fit the selected transformation and the selected control points.

Figure 12.1.: Unregistered image



5. Warping the input image using the transformation obtained.
6. Using the selected interpolation algorithm to fill the empty pixels produced after the transformation, to obtain the output.

**Implementation:**

As discussed above, QGIS has a georeferencer plugin which already has the code for finding the transformation coefficients using least square method, warping and interpolation. Moreover, QGIS is opensource and source code is readily available. So, the only feature to add is to read the text file for the coordinates and add it as control points.

The implementation of this feature involved:

1. Checking out the QGIS Source Code from github.
2. Installing the dependencies of QGIS to setup a development environment.
3. Locating the source code pertaining to the Georeferencer plugin.
4. Modifying the Qt based .ui file using QtCreator to add a button to the edit menu.
5. Modifying the .hpp file to add a function header to be linked with the button.
6. Adding the function definition to read the .txt file for Control Points in the .cpp file.
7. Linking the function with the button using SLOT - SIGNAL Qt concept.

Figure 12.2.: Sample Text file provided with the satellite image

```
ImageFormat=GeoTIFF
ProcessingLevel=STD
ResampCode=CC
NoScans=12000
NoPixels=12000
MapProjection=NONE
Ellipsoid=WGS_84
Datum=WGS_84
MapOriginLat=0.00000000
MapOriginLon=0.00000000
ProdULLat=24.23268159
ProdULLon=72.80976599
ProdURLat=24.18210139
ProdURLon=73.07036204
ProdLRLat=23.91278532
ProdLRLon=73.00721703
ProdLLLat=23.96331574
ProdLLLon=72.74716672
ProdULMapX=0.00000000
ProdULMapY=0.00000000
ProdURMapX=0.00000000
ProdURMapY=0.00000000
ProdLRMapX=0.00000000
ProdLRMapY=0.00000000
ProdLLMapX=0.00000000
ProdLLMapY=0.00000000
SceneCenterLat=24.07276725
SceneCenterLon=72.90862689
SatAltitude=624517.42100000
SunAzimuth=158.22480231
SunElevation=39.60703803
SatelliteHeading=192.22581100
AngleIncidence=29.07670729
DEMCorrection=NONE
SourceOrbitvalues=1
SourceQs=3
```

Figure 12.3.: Mosaic Example



8. Running the CCMake script to generate the machine specific Make files.
9. Running Make files to compile and link the C++ files to produce the QGIS binaries.

### **12.3.2. Image Stitching**

This problem is approached in two ways. The first method is to build a completely automatic stitching algorithm based on Feature Matching and the other method is to use human involvement to provide with some matching control points to stitch the images. The Geographic data associated provides with images provides extra information necessary to align the images. The source of misalignment is the error in the process of georeferencing.

#### **12.3.2.1. Automatic**

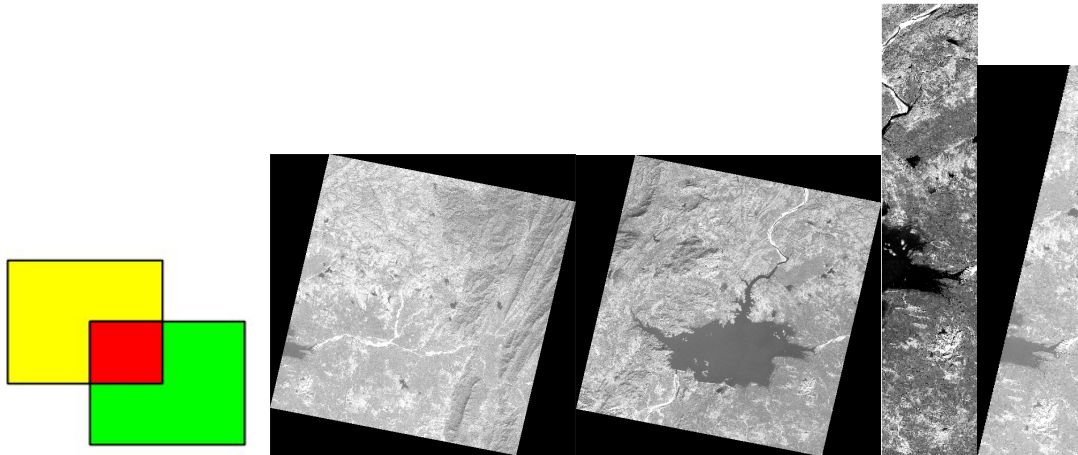
Steps in automatic image stitching software:

1. Find the overlap area of the images using geodata.
2. Extract the overlap area from both the images.
3. Further crop the overlap area to remove areas without data.
4. Scale down the images from 16bit float to 8bit unsigned integer.
5. Use SURF / SIFT to process both the images to find keypoints.
6. Run the SURF/SIFT keypoint descriptor to describe the features.
7. Use BruteForce/FLANN based k-nearest neighbor matcher to find the matches.
8. Construct a homography on the basis of matched keypoints.
9. Transform the images based on homography to stitch them.

#### **Implementation:**

During Literature review it was found that Auto-GR tools and NASA Vision Workbench had the features of automatic stitching using SIFT. On installation of Auto-GR tools, it was understood that the software has many limitations in terms of the format and size of the images that can be used in Auto-GR SIFT. The satellite images used in BISAG exceed the limitations therefore rendered Auto-GR tools unusable. Moreover, Auto-GR tools is not an opensource software. Therefore, it is not possible to adapt it to

Figure 12.4.: Overlap area, Georeferenced Input Images, Overlap Area from both images



the requirements of BISAG. NASA Vision Workbench is poorly documented and maintained. The makescript is not clearly explained and failed to produce a clean build. The instructions to build it for Windows is unavailable. Therefore, NASA Vision Workbench also failed to work.

The next option was to use OpenCV to run use SIFT/SURF and find the matches. SIFT is a time and memory consuming operation. The overlap area is extracted from both the images using geodata. Figure 12.1 shows the overlap data as intersection of two rectangles. A Python GDAL Script was written to extract the overlap area and trim the black space. Using OpenCV SIFT/SURF funtions correspondences are found. Correspondences are filtered based on Lowe's test. RANSAC algorithm is used to fit the large number of datapoints to a homography(8 coefficients). Images are warped to form the mosaic. The output produced is in the JPEG format with all the data scaled down to 8bit integer involving data loss. The output is not very satisfactory.

#### 12.3.2.2. Semi-Automatic

This method uses user input to minimize the error in the stitching process. As explained in the theory section, `gdal_merge.py` script can be used to merge two satellite images. The wrong output produced by `gdal_merge.py` script is shown in Figure 12.6.

Problems in `gdal_merge.py` was identified and the script was re-written with the following algorithm:

1. Find the dimensions and coordinates of the output image using the given images. (Figure 12.7)

Figure 12.5.: Output produced by using automatic geo-registration

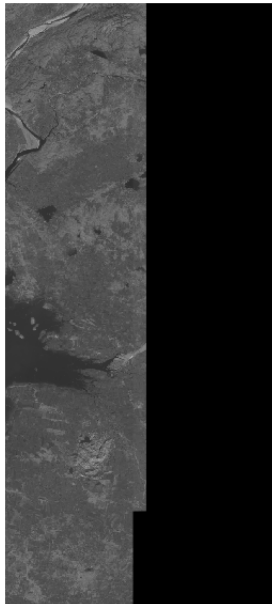


Figure 12.6.: Wrong output produced by gdal\_merge.py

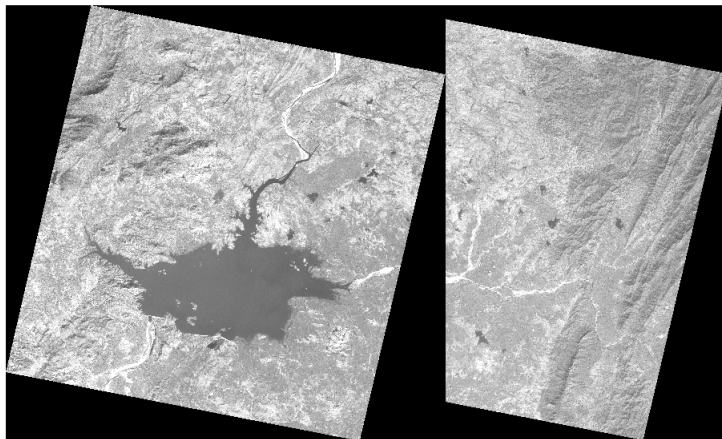


Figure 12.7.: Dimensions of Output Image, Configuration of Images

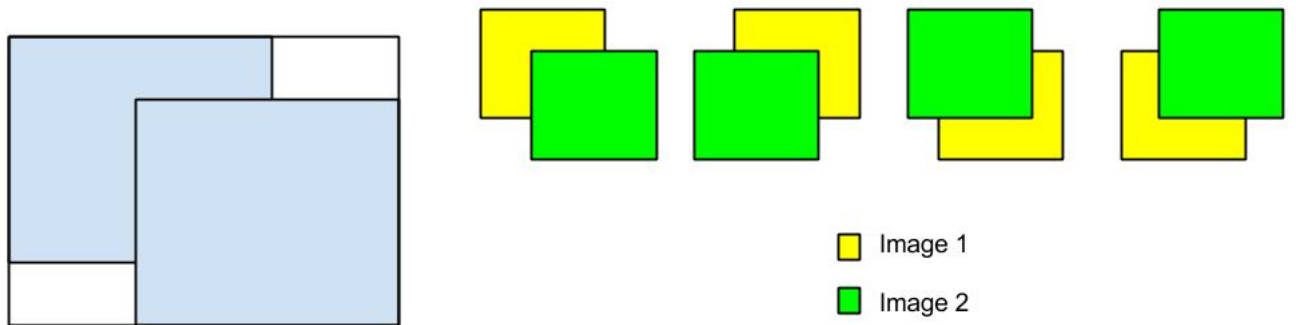
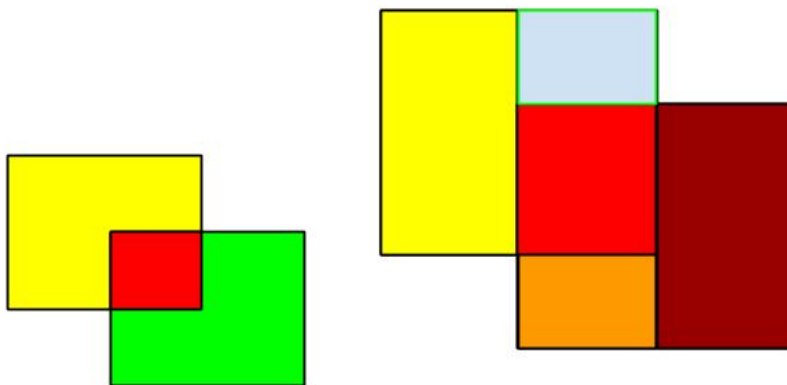


Figure 12.8.: Overlap Area, five parts of the output image



2. Using GDAL, create the empty output image.
3. The images can exist in 4 configurations. Identify the orientation in which these images exist. (Figure 12.7)
4. Identify the dimensions and coordinates of overlap area. (Figure 12.8)
5. The output image is divided into 5 parts, one of them being the overlap area. Copy everything except the overlap area. (Figure 12.8)
6. For the overlap area, compare the value pixel by pixel of the two input images. If one of them is a zero, copy the value from the other image. If both are non-zero always copy the pixel value from Image 1.

The output produced by this script is shown in Figure 12.9. This output also contains

Figure 12.9.: Output produced by modified gdal\_merge.py

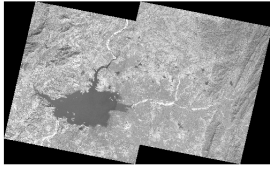


Figure 12.10.: Zoomed in merge image at the boundary



some error which is carried forward from the process of georeferencing. An image showing the error in the boundary of the merge image is shown in Figure 12.10. We modelled this error using a linear translation. A `translate.py` GDAL script was written to correct the error by translating the image with the user input of number of pixels to shift to left and up. After translating the second image, merging is done again using the same script to produce a seamless mosaic shown in Figure 12.11.

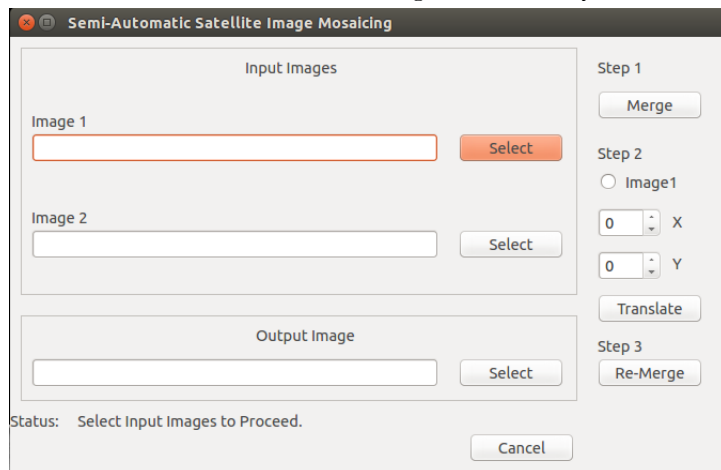
A Qt Based interface was built to support all these features under one software. It is shown in Figure 12.12. The semi-automatic software produces reliable and satisfactory output.



Figure 12.11.: Final Output of semi-automatic merging



Figure 12.12.: Qt-based GUI



Part IV.

## References

The following sources were referred during the project:

- Distinctive Image Features from Scale-Invariant Keypoints, D.Lowe.
- A Survey of Image Registration techniques, LG Brown.
- Georegistration of remotely sensed imagery, Stuart Ness.
- Image Alignment and Stitching: A Tutorial, Richard Szeliski.
- OpenCV official documentation, [docs.opencv.org](https://docs.opencv.org)
- GDAL Official Documentation, [www.gdal.org](http://www.gdal.org)