

The **lidar_image_align** application coregisters LIDAR and image data to form a consistent map. Specifically, it takes a set of Lunar Orbiter Laser Altimeter (LOLA) readings from the Lunar Reconnaissance Orbiter (LRO) and an Apollo metric camera image as input. It then aligns the two data sources to form a consistent map, and outputs a transformation from the original image coordinates to the adjusted image coordinates.

1 Files:

- `lidar_image_align.cc`, The main method, parses command line arguments and calls high level functions.
- `LidarImageAlign.cc` and `LidarImageAlign.h`, Gauss-Newton algorithm to align tracks to images. Also implements currently unused brute force search, and computes homographies between pairs of images, or two sets of LOLA track coordinates.
- `TracksGCP.cc` and `TracksGCP.h` implementaion of functions to update and save ground control points.

2 How to install:

1. *Install Prerequisites* - Install GDAL 1.9, OpenCV 2.4.9, ISIS 3.1, Eigen 3, Boost 1.5
2. *Build* - In the `lidar2image_processing/tests` directory, run “`cmake .`” followed by “`make`” to create the `tests/lidar2image` executable.

3 How to run

The command “`lidar_image_align -l lola_tracks.csv -i image.cub`” will align the LOLA tracks in the CSV file `lola_tracks.csv` to the image file `image.cub`. The transformation matrix and the ground control point files will be output in results directory under `filename_transf.txt` and `filename_trackIndex_shotIndex_gcp.txt` where `filename` is the stem of the image filename (filename with no path and no extension). The following options may be passed to `lidar_image_align` at runtime.

- `-l, -lidarFile filename`: a CSV file containing the LOLA shots to process or a file containing a list of LOLA CSV shots to process, separated by linebreaks
- `-i, -inputCubFile filename`: a cub image to align tracks to
- `-r, -results directory name`: optional parameter for the directory name where results are saved
- `-s, -settings filename`: optional parameter containing the settings filename
- `-t, -test mode`: optional parameter to run in test mode
- `-h, -help`: display a help message

`lidar_image_align.sh` is an script example to run data available in `lidar2image_processing/tests/data` directory.

4 The Algorithm

Many different spacecraft have succesfully returned various types of extraplanetary data to Earth, including imagery and elevation data. However, due to small uncertainties in space craft position, these different data sources have errors in alignment which makes forming a consistent multi-source map a challenging problem. Finding transformations from one data set to another to form a consistent map is the *coregistration* problem.

The problem of coregistering different sources of imagery data has been succesfully adressed with bundle adjustment. However, the problem of coregistering different classes of data, in particular aligning LIDAR measurements with images, has not been well-studied. This coregistration problem is particularly challenging because the types of data are fundamentally different. Furthermore, the available LIDAR data is much sparser than the image data.

The `lidar2img` program coregisters images from the Apollo 15 metric camera to data from the recently deployed Lunar Reconnaissance Orbiter’s Lunar Orbiter Laser Altimeter (LOLA). To do so, we first convert the LOLA data to a *synthetic image* by inputting measured surface normals, and estimated sun and spacecraft positions into a lunar reflectance model. Then, we find a planar homography which aligns the synthetic image to the actual image using the Gauss-Newton algorithm. However, the Gauss-Newton algorithm is susceptible to local minima, and a naive application will fail. Instead, we first apply Gauss-Newton to lower-resolution images which smooth over local minima. Then, we refine the transformation on successively higher-resolution layers of the image pyramid.

We first present and discuss the data sources in detail. Next, we introduce the Gauss-Newton algorithm as applied to the LIDAR to image coregistration problem, and its use in conjunction with the image pyramid. Finally, we present selected results demonstrating the effectiveness of the coregistration process.

4.1 Problem Statement

We are provided imagery from Apollo 15 and LIDAR scans from the Lunar Reconnaissance Orbiter (LRO). Our goal is to merge these two data sources to form one consistent map. Coregistering these two data sources is challenging due to significant uncertainties in both satellites’ positions.

The images of the Moon we use were taken from the Apollo Lunar Mapping Camera, also known as the Apollo metric camera, onboard the lunar command module of Apollo 15.

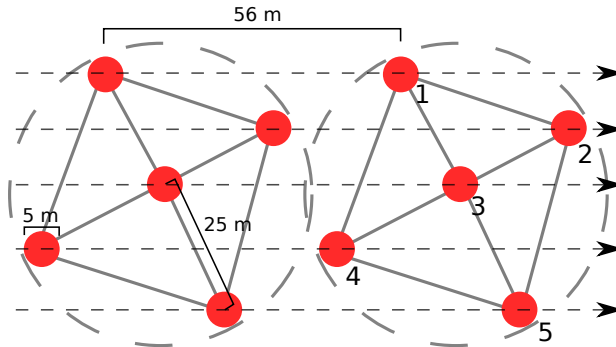


Figure 1: Two LOLA shots, part of a track. Each shot provides five distance measurements in a cross pattern. Shots are 56 m apart, and each laser beam reflects off a five meter diameter spot.

The laser data is acquired from the Lunar Orbiter Laser Altimeter (LOLA), an instrument of the Lunar Reconnaissance Orbiter (LRO). The LOLA splits a laser into five parts to take five distance measurements to the lunar surface, arranged in a cross shape. Each set of five measurements is called a *shot*. Each of the five laser beams illuminates a 5 m diameter spot on the moon’s surface (Figure 1).

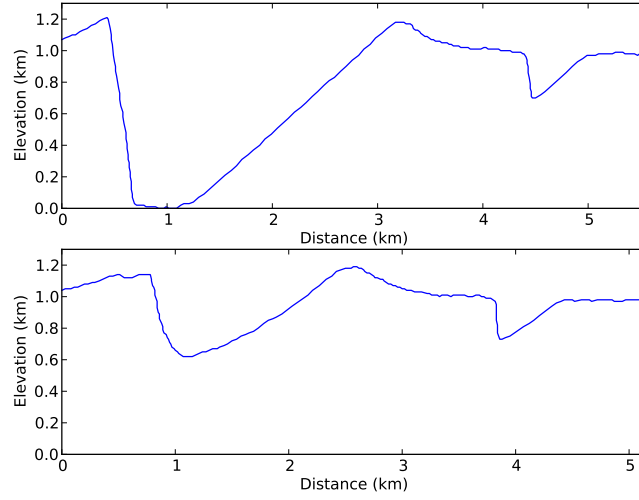
The LRO flies above the surface of the moon in a polar orbit, taking LOLA shots directly beneath it as it moves. Successive shots are approximately 56 m apart, and each sequence of shots from the same orbit forms a *track*. Figure 2(a) shows the elevation profile of a crater as measured by two LOLA tracks.

4.2 Aligning LIDAR to Images

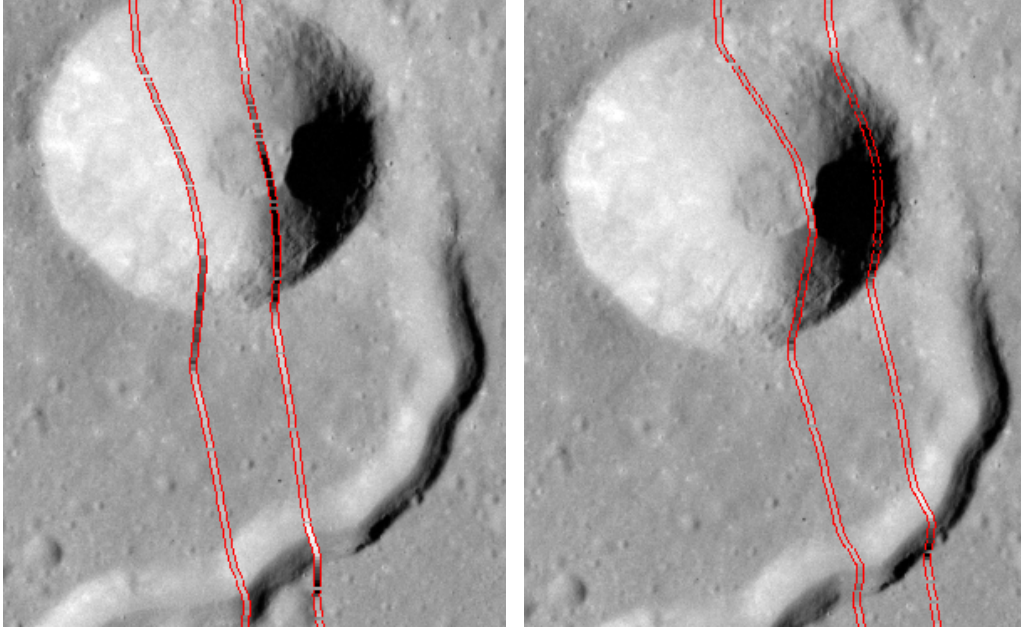
The data has two main sources of error: uncertainty in the position of the Apollo spacecraft’s position, and uncertainty in the position of the LRO when the LOLA shots were taken. As Apollo 15 launched in 1971, 38 years before the LRO, the Apollo images are subject to much larger position errors.

Uncertainty in Apollo 15’s position affects the image’s pose relative to all of the tracks, while error in the LRO’s position occurs on a per-track basis. We account for each of these error sources separately, both using the Gauss-Newton method. However, before we can align the two data sources, we form a *synthesized image* from the LOLA tracks that can be easily compared to the Apollo image.

Forming Synthesized Images



(a) LOLA Elevation Profile



(b) Before Coregistration

(c) After Coregistration

Figure 2: Two LOLA tracks that pass through the crater Hadley C and Rima Hadley are shown. (a) shows the LOLA tracks' elevation profiles. (b) shows the tracks' estimated reflectance (the synthetic image) against an Apollo 15 image with the original alignment. (c) shows the synthetic image and the Apollo 15 image after coregistration. The LOLA tracks are delineated by two bright lines, and the synthetic image is shown as the color between the lines.

To align the LOLA tracks to the Apollo images, we estimate the lunar reflectance r for each LOLA shot to form a *synthesized image*. The reflectance is a function of the angle of incidence of light, which depends on the angle to the sun and the surface normal. The surface normal is computed by adding the normals from each available triangle in the LOLA shot (see Figure 1) and normalizing the result. Some triangles' normals may not be available since LOLA does not always successfully return five surface readings.

Given the surface normal and the vector from the surface to the sun, we compute the expected reflectance with the photometric equation presented in by McEwen. These estimated shot reflectances form a synthetic image, which we expect to match the actual images taken during the Apollo mission.

The actual surface luminence seen in an image is given by the product of the reflectance and a constant factor, αr . The value of α depends on two factors: the parameters of the camera that took the image, and the properties of the reflecting surface. The reflectance properties of the lunar surface vary, particularly between the flat, low-lying maria and the highlands. However, they are locally consistent.

Given a set of tracks, an image, and a transformation from track coordinates to image coordinates, we estimate the scaling factor α . Let p be the list of all LOLA shots' transformed image coordinates, where p_i^x and p_i^y are the x and y coordinates of LOLA shot i transformed by a matrix M , let r be a list of all LOLA shots' estimated reflectances, and let I be an image. Then we compute the scaling factor as the value which makes the average value of the actual image equal the average value of the synthetic image, $\alpha = (\sum I(p_i^x, p_i^y)) / (\sum L_i^r)$. So α is a function of M .

The Gauss Newton Algorithm

Let L be a list of LOLA shots, where L_i^x and L_i^y denote the initial image coordinates and L_i^r denotes the estimated reflectance. Our goal is to find

$$\arg \min_M S(M) = \arg \min_M \sum_i r_i(M)^2,$$

where $r_i(M)$ is the error for a single LOLA shot's synthetic image given the transformation matrix M , the 3×3 homography that minimizes the error between the synthetic and the observed images.

$$r_i(m) = I \left(M \begin{bmatrix} L_i^x \\ L_i^y \\ 1 \end{bmatrix} \right) - \alpha L_i^r.$$

The true error in the satellites' positions cannot be accounted for with a planar homography, since the surface of the moon and the LRO's motion are both non-planar. However, we have found that planar homographies are an effective approximation across limited areas.

Minimizing this objective function is a non-linear least squares problem. A common tool for solving such problems is the Gauss-Newton algorithm, a variant of Newton's method which doesn't require the computation of second derivatives.

In each step of the algorithm, we find a change Δ to the parameters β (the entries of M written as a vector) such that the error decreases, where $\beta_{t+1} = \beta_t + \Delta$.

$$-J_I(\beta)^T r = J_I(\beta)^T J_I(\beta) \Delta$$

We solve this system of linear equations for Δ with an application of singular value decomposition.

Then, we continue taking steps with new Δ until the error doesn't decrease. The full Gauss-Newton algorithm is shown in Algorithm 4.2.

$err_{old} = \infty$

loop

$$\forall i \in \{1, \dots, |L|\} \begin{bmatrix} p_i^x \\ p_i^y \\ 1 \end{bmatrix} = M \begin{bmatrix} L_i^x \\ L_i^y \\ 1 \end{bmatrix}$$

$$\alpha = (\sum I(p_i^x, p_i^y)) / (\sum L_i^r)$$

$$\forall i \in \{1, \dots, |L|\} r_i = I(p_i^x, p_i^y) - \alpha L_i^r$$

$$err = \sum_i r_i^2$$

if $err \geq err_{old}$ **then**

return M_{prev}

end if

$$err_{old} = err$$

$$J_I = \begin{bmatrix} \frac{dI(p_1^x, p_1^y)}{dM_{1,1}} & \frac{dI(p_1^x, p_1^y)}{dM_{1,2}} & \frac{dI(p_1^x, p_1^y)}{dM_{1,3}} & \frac{dI(p_1^x, p_1^y)}{dM_{2,1}} & \frac{dI(p_1^x, p_1^y)}{dM_{2,2}} & \frac{dI(p_1^x, p_1^y)}{dM_{2,3}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{dI(p_n^x, p_n^y)}{dM_{1,1}} & \frac{dI(p_n^x, p_n^y)}{dM_{1,2}} & \frac{dI(p_n^x, p_n^y)}{dM_{1,3}} & \frac{dI(p_n^x, p_n^y)}{dM_{2,1}} & \frac{dI(p_n^x, p_n^y)}{dM_{2,2}} & \frac{dI(p_n^x, p_n^y)}{dM_{2,3}} \end{bmatrix}$$

$$D = (J_I^T J_I)^{-1} J_I^T E$$

$$\Delta = \begin{bmatrix} D_1 & D_2 & D_3 \\ D_4 & D_5 & D_6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_{prev} = M$$

$$M = M + \Delta$$

end loop

Search Over the Image Pyramid

The Gauss-Newton algorithm, like Newton's method, is highly susceptible to local minima. In our problem, the LOLA tracks begin highly misaligned, and a naive application of Gauss-Newton will converge to nearby local minima rather than the global minima.

To prevent this, we first search on downsampled, lower-resolution layers of the image pyramid (see Fig. 3). By downsampling we find only a coarse alignment, but the lower resolution image allows us to avoid local minima and coregister the tracks in the presence of large initial errors.

Once a coarse alignment is found, we align the LOLA tracks on progressively higher resolution layers of the image pyramid using Algorithm 4.2. Note that we must scale the translational component of the transformation matrix between layers of the image pyramid to account for the changed image coordinate system.

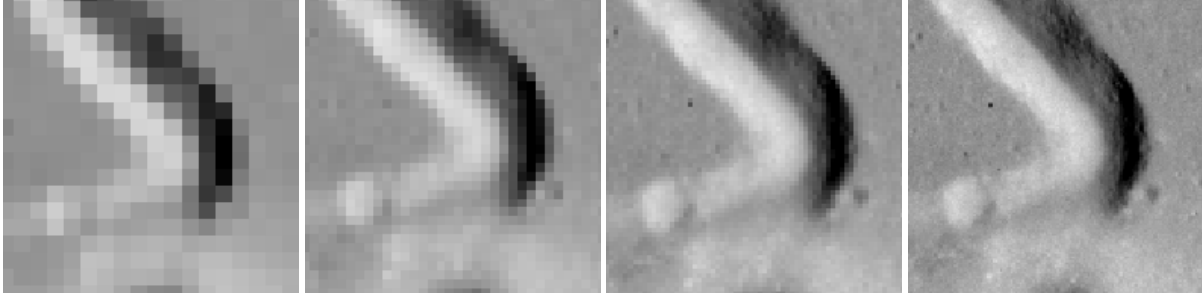


Figure 3: Hadley Rille, as seen in multiple layers of the image pyramid. From left to right, downsampled by factors of 8, 4, 2, and 1.

```

P = image_pyramid(I, n)
T0 =  $\begin{bmatrix} 1/2^n & 0 & 0 \\ 0 & 1/2^n & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 
G = gauss_newton(L, Pn, T0)
∀ i ∈ {1, ..., |T|} Mi = G
for i from n - 1 to 0 do
  for j ∈ {1, ..., |T|} do
    Mj =  $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  Mj  $\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 
    Mj = gauss_newton(L, Pi, Mj)
  end for
end for
return M

```

Computing the Jacobian

Recall that the Gauss-Newton algorithm requires us to compute the Jacobian $J_I(\beta)$ of the error with respect to the parameters of the transformation matrix. We skipped over this previously, but now we discuss this in detail.

We aim to find the homography H from track coordinates (x, y) to image coordinates (x', y') .

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}$$

$$\begin{bmatrix} H_{11}x + H_{12}y + H_{13} \\ H_{21}x + H_{22}y + H_{23} \\ H_{31}x + H_{32}y + H_{33} \end{bmatrix} = \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}$$

$$x' = \frac{H_{11}x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}}$$

$$y' = \frac{H_{21}x + H_{22}y + H_{23}}{H_{31}x + H_{32}y + H_{33}}$$

For the Gauss-Newton algorithm, we require the partial derivatives of the image with respect to each entry of the homography. For the first two rows of H , we compute this by taking a finite difference with three points on the image, either over a row in the image (for the first row of the homography) or over a column (for the second row). We use the finite difference with six points.

$$\begin{aligned} \Delta I_x(x', y') &= -\frac{1}{60}I(x' - 3, y) + \frac{3}{20}I(x' - 2, y') - \frac{3}{4}I(x' - 1, y) + \\ &\quad \frac{3}{4}I(x' + 1, y) - \frac{3}{20}I(x' + 2, y') + \frac{1}{60}I(x' + 3, y) \end{aligned}$$

Once we find the change in the horizontal direction of the image, we find the change in the entries of H with induces a change of a single pixel in the image coordinates.

$$\begin{aligned} \frac{(H_{11} + \Delta H_{11})x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}} - \frac{H_{11}x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}} &= 1 \\ \frac{H_{31}x + H_{32}y + H_{33}}{H_{31}x + H_{32}y + H_{33}} &= \Delta H_{11}x \\ \frac{H_{31}x + H_{32}y + H_{33}}{H_{31}x + H_{32}y + H_{33}} &= \Delta H_{11} \\ \frac{x}{H_{31}x + H_{32}y + H_{33}} &= \Delta H_{12} \\ \frac{y}{H_{31}x + H_{32}y + H_{33}} &= \Delta H_{13} \end{aligned}$$

Then we can approximate the partial derivatives as

$$\frac{dI}{dH_{11}} = \frac{\Delta I_x(x', y')}{\Delta H_{11}}, \quad \frac{dI}{dH_{12}} = \frac{\Delta I_x(x', y')}{\Delta H_{12}}, \quad \frac{dI}{dH_{13}} = \frac{\Delta I_x(x', y')}{\Delta H_{13}}$$

The equations are the same for the second row of H , except we take a finite difference in the vertical direction instead:

$$\frac{dI}{dH_{21}} = \frac{\Delta I_y(x', y')}{\Delta H_{21}}, \quad \frac{dI}{dH_{22}} = \frac{\Delta I_y(x', y')}{\Delta H_{22}}, \quad \frac{dI}{dH_{23}} = \frac{\Delta I_y(x', y')}{\Delta H_{23}}$$

The third row of H determines the scaling factor, which will multiply both x and y by the same amount. If $x' > y'$, we find the change in H which brings us to the next pixel in the x direction, if $y > x$ we find the change to bring us to the next pixel in the y direction. We interpolate between neighboring pixels for non-integral image coordinates.

Let $m = \frac{y'}{x'}$. If $m \leq 1$, we sample find the divided difference using pixels with an even vertical offset.

$$\begin{aligned} \Delta I(x', y') &= -\frac{1}{60}I(x' - 3, y - 3m) + \frac{3}{20}I(x' - 2, y' - 2m) - \frac{3}{4}I(x' - 1, y - m) + \\ &\quad \frac{3}{4}I(x' + 1, y + m) - \frac{3}{20}I(x' + 2, y' + 2m) + \frac{1}{60}I(x' + 3, y + 3m) \end{aligned}$$

Likewise, if $m \geq 1$, we find the divided difference from pixels with a horizontal offset.

$$\begin{aligned} \Delta I(x', y') = & -\frac{1}{60}I(x' - \frac{3}{m}, y - 3) + \frac{3}{20}I(x' - \frac{2}{m}, y' - 2) - \frac{3}{4}I(x' - \frac{1}{m}, y - 1) + \\ & \frac{3}{4}I(x' + \frac{1}{m}, y + 1) - \frac{3}{20}I(x' + \frac{2}{m}, y' + 2) + \frac{1}{60}I(x' + \frac{3}{m}, y + 3) \end{aligned}$$

Next, we once again find the changes in H which produce this x or y offset. First, if it is an x offset:

$$\begin{aligned} \frac{H_{11}x + H_{12}y + H_{13}}{(H_{31} + \Delta H_{31})x + H_{32}y + H_{33}} - \frac{H_{11}x + H_{12}y + H_{13}}{H_{31}x + H_{32}y + H_{33}} &= 1 \\ \frac{H_{11}x + H_{12}y + H_{13}}{(H_{31} + \Delta H_{31})x + H_{32}y + H_{33}} &= 1 + x' \\ H_{11}x + H_{12}y + H_{13} - (H_{32}y + H_{33})(1 + x') &= (1 + x')(H_{31} + \Delta H_{31})x \\ \frac{1}{x} \left(\frac{H_{11}x + H_{12}y + H_{13}}{(1 + x')} - H_{32}y - H_{33} \right) - H_{31} &= \Delta H_{31} \\ \frac{1}{y} \left(\frac{H_{11}x + H_{12}y + H_{13}}{(1 + x')} - H_{31}x - H_{33} \right) - H_{32} &= \Delta H_{32} \\ \frac{H_{11}x + H_{12}y + H_{13}}{(1 + x')} - H_{31}x - H_{32}y - H_{33} &= \Delta H_{33} \end{aligned}$$

If it is a y offset ($y' > x'$), the results are the same, except the entries from the first row of H are replaced with entries from the second row. Furthermore, x' becomes y' . Then, once again, we find the derivatives.

$$\frac{dI}{dH_{31}} = \frac{\Delta I_y(x', y')}{\Delta H_{31}}, \quad \frac{dI}{dH_{32}} = \frac{\Delta I_y(x', y')}{\Delta H_{32}}, \quad \frac{dI}{dH_{33}} = \frac{\Delta I_y(x', y')}{\Delta H_{33}}$$

Now we are able to generate the Jacobian $J_r(\beta)$.

4.3 Selected Results

Next, we show selected results with Gauss-Newton on an image pyramid aligning LOLA tracks to Apollo-15 images. Figures 4 and 5 show the position of the LOLA tracks on an Apollo image both before and after finding a transformation. The red lines indicate the tracks, and the color between them indicates the expected luminence of the moon based on the surface normal, sun position and camera position. The initial error in both cases is large, as seen by the large mismatch between the color of the LOLA track and the background image.

After Gauss-Newton on the image pyramid is applied, a successful transformation is found. The expected luminence closely matches the actual luminence.

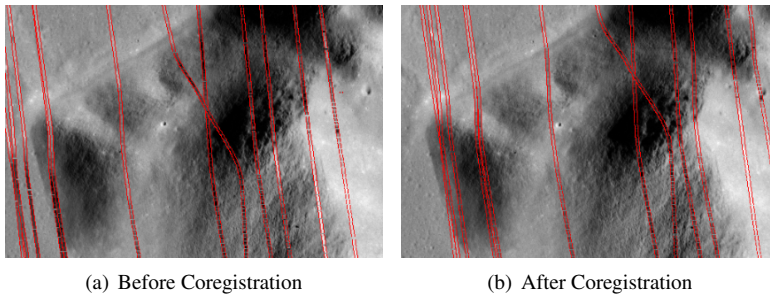


Figure 4: LOLA tracks passing through the western part of Mons Hadley are shown (a) before coregistration, and (b) after coregistration. Note how after coregistration, the synthetic image matches the actual image much more closely.

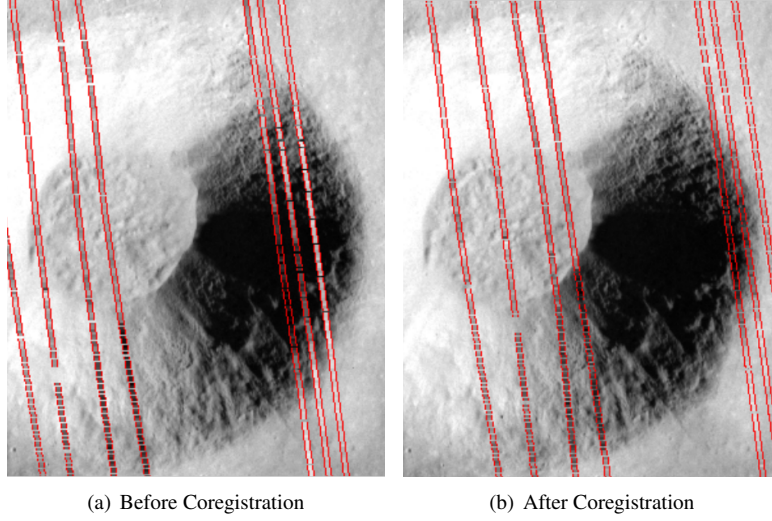


Figure 5: LOLA tracks over Aratus crater (a) before coregistration, and (b) after coregistration. The sides of the crater form distinctive visual features.

Although our algorithm allows us to find general planar homographies from the track coordinates to image coordinates, in practice the transformations that are found are affine transformations. They have a large translational component and a slight rotational component.

4.4 Conclusion

We have presented the algorithm that aligns orbital LIDAR data to image data in the presence of substantial position errors implemented by the `lidar2img` program. The algorithm applies Gauss-Newton on an image pyramid to avoid local minima. We have demonstrated the algorithm’s success on lunar data, and successfully coregistered LIDAR and image data of the moon.