Using Dense Visual Odometry to Create Path Overlay

1 Problem Definition

Motion estimation is one of the key problems in robotics. This project tackles estimating motion using visual odometry, specifically with dense methods rather than sparse feature tracking. An overlay of the rover's path was also created on the overhead image of the terrain. For the Mars rover and other robots, odometry and map construction are essential functionalities that can be accomplished or augmented using visual methods. Additionally, pose estimation allows plotting the path of the rover given an overhead image of the terrain. This provides a visual representation of explored and unexplored territory. Due to the usefulness of visual odometry and renewed interest in dense methods, I believe it is vital to understand dense visual odometry fully.

Motion is estimated using pairs of stereo images taken a small increment of time apart. Each motion estimate provides a transformation from the first pose to the second. Stitching together these transformations from a known starting pose, the UTM coordinates of the rover can be found at each increment.

Originally this project aimed to create a visual SLAM pipeline similar to the one used in LSD-Slam [1], optimizing for both the global map and each transform simultaneously. However, the project scope was reduced to dense visual odometry without mapping due to time constraints. Additionally, the Lucas-Kanade [2] approach to dense optical flow using Gauss-Newton non-linear optimization was used rather than the Levenberg-Marquardt approach in [1]. Several aspects of LSD-SLAM were maintained, specifically the concept of semi-dense depth maps.

I decided to test the motion estimation on a section of run #1 using the 4^{th} and 5^{th} omni-cameras. Run #1 was chosen because it has the lowest average speed of the rover, and visual odometry methods are known to be unreliable at high speeds. The section of run 1 from frame 1600 to frame 1825 was used because it has large curvature and a distinct shape which can challenge the algorithm. Omni-cameras 4 and 5 were chosen because they are the stereo pair that faces forward, in the direction of movement.

The code submitted runs the algorithm for frames 1750-1775 rather than the entire section used for evaluation and shortens the runtime by trading off accuracy. This is due to the very long run-time of the algorithm, discussed more in section **3.3 Performance.** Although the exact results will not be replicated, this shortened and sped-up run should demonstrate the efficacy of the algorithm.

2 Methodology

The algorithm takes in a start and end frame, and outputs the estimated position in UTM coordinates at every frame. It runs one main loop for the images at each frame:

```
For each frame from start_frame to end_frame:
   1. Get the omni-camera 4 and omni-camera 5 images at current frame
   2. Rectify and undistort the images
   3. Find the disparity map from the images
   4. Estimate motion using the disparity map from the previous frame
      and the omni-camera 4 images from both this frame and the
      previous frame.
```

2. 1 Rectify Images

Before the disparity map can be created, the stereo pairs of images need to be de-warped and rectified. Rectification transforms both images onto a common image plane so that the correspondences can be found by following epipolar lines. The epipolar lines in this project are vertical since the stereo cameras are mounted vertically. Rectification was done using OpenCV. Although the proposal did not state that OpenCV would be used, I did not account for the difficulty of rectifying and de-warping images using NumPy and SciPy alone.

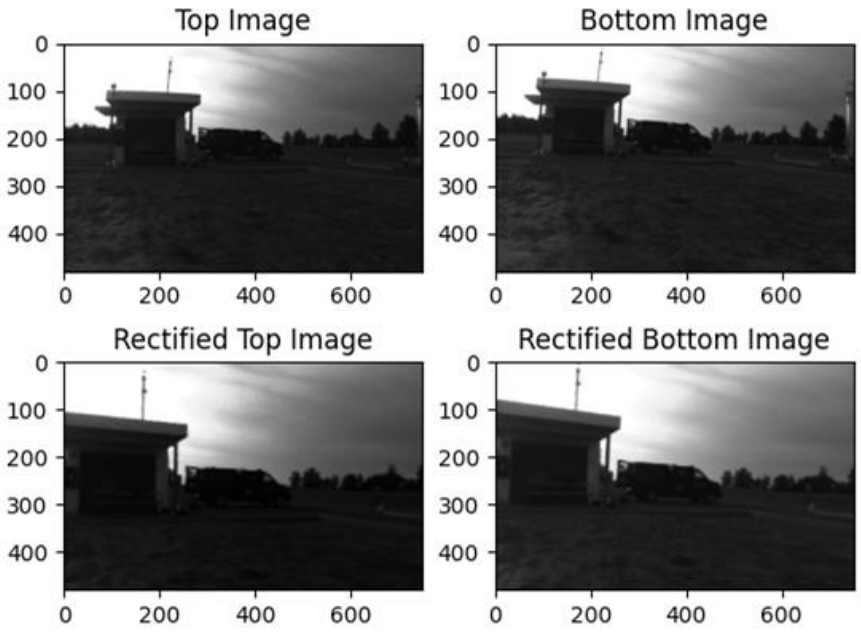

Figure 1: Stereo rectification using OpenCV

As can be seen, the rectification process cuts out significant portions of the image which are too warped or cannot be seen from both images. This leaves only the portion of the image suitable for calculating disparity.

2.2 Disparity

After the images are rectified, the disparity map can be calculated. To calculate disparity, I used the algorithm I implemented for Assignment 3, the Census Transform [3]. I chose this algorithm because it won the competition for Assignment 3 and was relatively fast. However, due to the images being mostly texture-less, the disparity was very noisy. To fix this, disparity was filtered to only include areas around high-gradient pixels. This was done by creating a mask of the high-gradient pixels and dilating it using OpenCV. The dilation was necessary because the there are extremely few high gradient pixels, but the area surrounding them tends to be equally accurate since the algorithm is window-based.

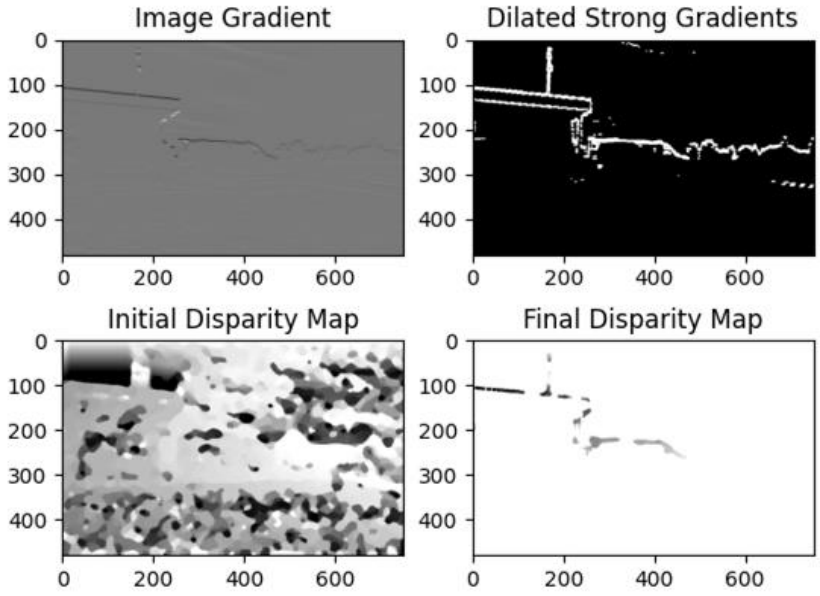

Figure 2: Comparison of the complete disparity map (lower left) with the disparity map after choosing only informative pixels (lower right). This is done by finding the gradient perpendicular to the epipolar lines (upper left) and dilating large gradients (upper right)

Very few pixels are used in the final disparity map (around 1.5% of pixels are used in the above example), but these pixels are much more accurate than using the entire image. This is in line with the approach used in LSD-SLAM [1] which called it semi-dense depth mapping.

2.3 Motion Estimation

Motion estimation is done by minimizing photometric error. Using the assumption of photometric consistency, that a pixel in one image will be the same intensity as the pixel in another image that is representing the same 3D point, the error residual is:

$$r_u = I_1(u) - I_2(\pi(p'))$$

Where: u is a pixel coordinate
r_u is the residual related to pixel coordinate u
$I_x(u)$ is the intensity of image x at pixel coordinate u
p is a point in 3D space
$\pi(p)$ is the pixel coordinate projection of point p

Point p' is the pixel coordinate u transformed into 3D space and then transformed by the motion of the camera between the two images:

$$p' = T_\xi \pi^{-1}(u, D_1(u))$$

Where T_ξ is the homogenous transformation matrix related to twist ξ
$D_x(u)$ is the disparity at pixel coordinate u corresponding to image x
$\pi^{-1}(u, D_x(u))$ is the inverse projection of $\pi(p)$

Thus, the error we are trying to minimize is:

$$E(\xi) = \sum_{u \epsilon D_1} {r_u}^2$$

This is the Lucas-Kanade [2] formulation for photometric error. Using Gauss-Newton optimization explained in [4], the optimal twist ξ can be found and thus recover the homogenous transformation of the camera pose from the first image to the second, T_ξ.

3.4 Final Transformation

Once all transformation between every consecutive frame has been calculated, these transformations need to be converted to UTM coordinates. To do this, first we use the ground truth position and orientation to define a transformation from the origin of the UTM zone 18T to the rover body frame at position i, T_{Ri}^{18T}. Then, using the transformations provided in the CPET dataset, we transform this frame into the frame of the top camera, omni-camera 4, T_{C4}^{R}. Next, we transform this camera from the current frame to the next frame using the calculated transform, T_ξ**.** Finally, we transform this

back into the rover frame using the inverse of the previous transform, T_R^{C4}**.** This gives us the transform from the origin of 18T to the rover body frame at the next pose, T_{Ri+1}^{18T}.

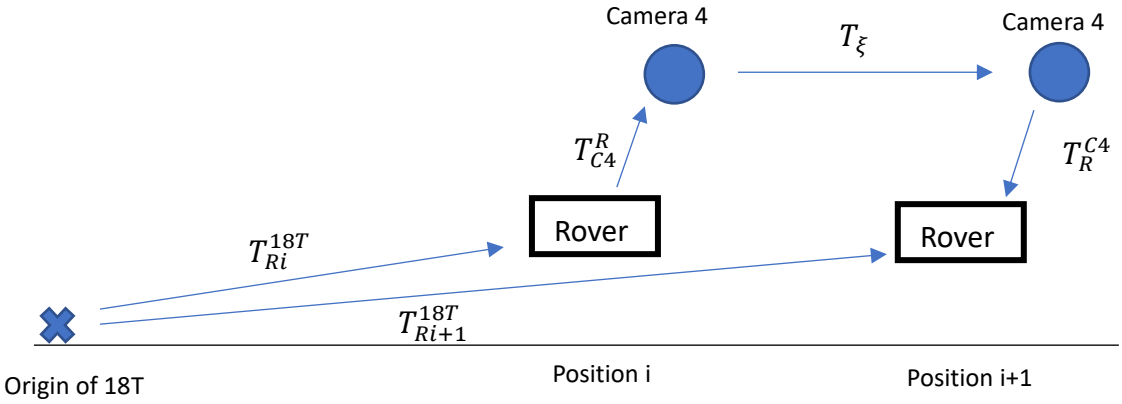

Figure 3: All the transformations necessary to compute UTM coordinates of the rover given successive movements in the omni-camera 4 frame.

Thus, the final equation for the pose of the rover in the second frame (position 1) in the frame set at the origin of zone 18T is:

$$T_{Ri+1}^{18T} = T_{Ri}^{18T} * T_{C4}^{R} * T_\xi * T_R^{C4}$$

From this, the UTM coordinates are simply the x and y translation of the T_{Ri+1}^{18T} homogenous transform, which are the easting and northing respectively. This can be repeated for all the frames to get the UTM coordinates of the rover at each frame.

3 Evaluation and Results

3.1 Quantitative Evaluation

The accuracy was measured using Absolute Trajectory Error (ATE). The ground truth was the rover pose calculated using by the VINS-Fusion package, provided in the dataset specifically for comparison to other motion estimation algorithms. The root mean squared error of my algorithm's position estimates compared to the ground truth was then calculated. However, the ground truth is only provided at specific times that do not align with when each frame was captured. Thus, to find the true position of the rover at the time of each frame, the two closest true poses were linearly interpolated. Only the position was used since error in the rotation would also affect the calculated positions, and this project was dedicated to finding the path of the rover.

Below you can see a graph of the ATE as a function of the length of the trajectory, in frames. The ATE increases with the number of frames, as the pose

estimates begin to drift. A closed-loop SLAM algorithm that optimizes the map and localization simultaneously, as well as loops around to observe previously mapped locations, would be capable to eliminating this drift.

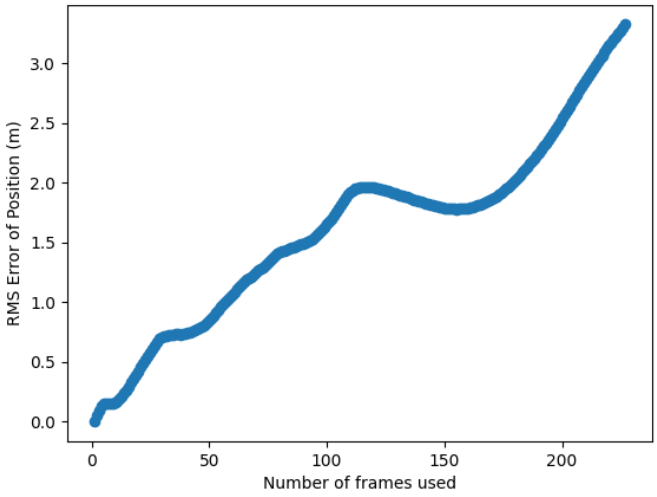

Figure 4: RMS error as a function of the number of frames the algorithm operates on

3.2 Qualitative Results

Finally, the calculated path was plotted on the overhead image of the terrain, alongside the path calculated from the ground truth positions. The calculated path clearly drifts significantly compared to the true path, but generally follows the same shape.

Figure 5: The full path (blue), the interpolated positions of the section used (green), and the estimated path (red). The size of the red circles represents the photometric error of the calculated transform at that point.

This result is expected since visual odometry is known to drift. As can be seen in *Figure 5*, the error increases significantly around the sharp turn, and in fact the algorithm completely fails to account for this turn. It instead turns slightly in the opposite direction. However, on flat straightaways the algorithm performs well. The initial estimate of the algorithm is set to a 10cm forward movement, so it is expected to perform well when the estimate is accurate. Additionally, the error is quite high in the initial part of the run before the sharp turn. This is likely because this part of the run is quite bumpy, and the rover is at a significant slant.

3.3 Performance

The runtime of the algorithm is immense. It takes approximately 1 minute to get the disparity map for each frame, and another minute to calculate the optimal transformation for that frame. For the 225 frames used in for the final results, the algorithm took approximately 8 hours to complete. This is far from real-time, and greatly limited my ability to tune, or add any additional computation such as a coarse-to-fine strategy. A much faster disparity calculation method along with a vectorized motion estimation algorithm is necessary to lower the run-time. It is also possible that down-sampling the images would significantly lower run-time while only minimally reducing accuracy, as was found in LSD-SLAM [1]. A non-linear optimizer that converges faster than Gauss-Newton, such as Levenberg-Marquardt, as well as a more efficient photometric residual such as the methods outlined in [4] could also reduce run-time.

4 References

[1] J. Engel, J. Stueckler, D. Cremers, “Large-Scale Direct SLAM with Stereo Cameras”, *International Conference on Intelligent Robots and Systems (IROS)*, 2015. Available: https://vision.cs.tum.edu/_media/spezial/bib/engel2015_stereo_lsdslam.pdf

[2] B. Lucas, T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision”, *International Joint Conference on Artificial Intelligence (IJCAI)*, 1981 Available: https://www.ijcai.org/Proceedings/81-2/Papers/017.pdf

[3] R. Zabih, J. Woodfill, “Non-parametric Local Transforms for Computing Visual Correspondence”, *European Conference on Computer Vision (ECCV),* 1994. Available: http://www.cs.cornell.edu/~rdz/Papers/ZW-ECCV94.pdf

[4] S. Baker, I. Matthews, “Lucas-Kanade 20 Years On: A Unifying Framework”, *International Journal of Computer Vision (IJCV),* 2004. Available: https://www.ri.cmu.edu/pub_files/pub3/baker_simon_2002_3/baker_simon_2002_3.pdf