Salar Hosseini
December 21, 2020

Stereo Visual Odometry Using Single-Scale and Transformation-Invariant Feature Detectors

Problem Definition:

The aim of this project is to implement a complete feature-based visual odometry pipeline and compare the performance from using the Harris corner detector [1] versus the more recently developed SIFT [2] and BRISK [3] feature detectors, to answer the question: would scale and transformation invariant feature detectors have been superior to the single scale Harris detector that was deployed on the Mars Exploration Rover (MER) for visual odometry [4]? It was hypothesized that, by using stereo imagery data collected at the Mars Emulation Terrain (MET) in Saint-Hubert, Quebec, BRISK and SIFT features would result in more accurate pose estimations from visual odometry than Harris corner detections due to their scale and transformation invariance properties.

This investigation is relevant to the design of future visual odometry pipelines for Mars rovers since it will provide insight to whether using transformation invariant detectors will result in much improved egomotion estimations, and if so, how much speed is sacrificed to achieve such estimations. Tackling this problem was very fulling for me since it gave me the opportunity to implement a full video odometry pipeline and better understand the effects of using different feature detectors. The problem investigated here is the same as that which was previously proposed.

Methodology:

As the goal of this project was to implement the visual odometry pipeline (using different feature detectors) and evaluate how it would perform on Mars, the dataset used was the *Canadian Planetary Emulation Terrain Energy-Aware Rover Navigation Dataset* [5] which was collected at the Canadian Space Agency's Mars Emulation Terrain. Listed below are the relevant files from the dataset that were used to perform the stereo visual odometry (note that for the purpose of this project only one stereo camera pair was used so only data pertaining to cameras 0 and 1 was relevant):

- "*omni_image{0-1}/frame{0000-9999}_YYYY_MM_DD_hh_mm_ss_uu.png*": the RGB stereo imagery collected from the omnidirectional sensor on the Husky robot
- "*camera_intrinsics.txt*": intrinsic and extrinsic calibration parameters of the stereo cameras
- "*rover_transforms.txt*": homogeneous transformation parameters between stereo camera frames

The performance of three feature detectors was evaluated for the video odometry pipeline: the Harris (single scale), SIFT (scale, rotation, and affine invariant), and BRISK (scale, rotation, and affine invariant) detectors. The Harris detector was chosen since it was the detector that was deployed for the MER [4] and it serves as a useful baseline for comparing against. The motivation for using the SIFT and BRISK detectors over other popular detectors such as SURF [6] and ORB [7] was based on recent experiments from Khan et al. [8]. They showed that from these detectors, SIFT and BRISK were found to have the highest overall image matching accuracies for all types of geometric transformations, and that BRISK was only second to ORB for matching speed, while ORB was the least scale invariant [8]. Furthermore, it is important to consider that these detectors produce features in different forms such as blobs and corners. Since SIFT is a blob-based detector while BRISK is corner-based, it was expected that SIFT would produce more accurate image matches in an environment potentially containing more blobs [8] such as the Mars Emulation Terrain.

The visual odometry approach that was implemented followed a similar methodology to that used by the MER [4] to emulate how it may have improved from better feature detectors. The only required computational resource was a CPU, and the necessary software libraries were NumPy, SciPy, and OpenCV (further details of usage are given later). Given the stereo image pair from the current and previous time steps (times *t* and *t-1*), the pose change is computed using the following steps:

- <u>Preprocessing</u> – the stereo image pair from time *t* is de-warped and rectified
- <u>Feature Detection</u> – Harris, SIFT, or BRISK features are detected for the preprocessed images
- <u>Stereo Matching</u> – feature correspondences are found using a 1D cross correlation search between the two frames (since they are vertically fronto-parallel)
- <u>Triangulation</u> – the 3-D positions of the features are found by finding the shortest distance gap between the corresponding rays projected through the camera model

- Feature Tracking – correspondences between features and 3D positions of landmarks in the stereo image pairs from times *t* and *t-1* are found by comparing feature descriptors (or surrounding pixel intensities in the case of the Harris corner detector)
- Scalar Weighted Point Cloud Alignment – Camera motion is estimated using a scalar weighted linear least squares approach
- Matrix Weighted Point Cloud Alignment – An iterative gauss-newton approach is used to minimize errors considering 3D position differences and the inverse covariance of the error

The main difference between this algorithm and the one implemented by the MER [4] is the exclusion of a RANSAC stepped which occurred in conjunction with scalar weighted point cloud alignment to reject outlier features and find a better initialization to use for the following non-linear least squares step. RANSAC was hypothesized to be unnecessary during the project since the matrix weighted point cloud alignment was converging without it, but the evaluation results will show that it may have been beneficial and should be considered in future work. The RANSAC algorithm used by the MER is summarized below:

- RANSAC and Scalar Weighted Point Cloud Alignment – For a fixed number of iterations:
    - A small number of features is selected
    - Camera motion is estimated using a scalar weighted linear least squares approach
    - The iteration is assigned a score proportional to the number of selected features that were accurately (based on a heuristic) projected using the camera motion estimate

  The camera motion from the iteration with the highest score is selected as the initialization for the next step, and only the features that are accurately projected using this motion will be used.

In the following subsections, details about the steps in the visual odometry implementation will be provided in addition to relevant libraries which were used in each.

Preprocessing
The first step in each iteration of the algorithm was to load the images from the stereo camera pair *omni0* (top image) and *omni1* (bottom image), and to undistort and rectify them (to become vertically fronto-parallel). The mapping from distorted and unrectified to undistorted and rectified image space was computed using OpenCV's *stereoRectify()* function. This function was inputted the camera intrinsic and distortion parameters, which were parsed from the *camera_intrinsics.txt* file, as well as the rotation and translation from the coordinates of camera one to zero which was computed using the pose quaternion in the *rover_transforms.txt* file. The mapping was then applied to each image. The *stereoRectify* function additionally returned region of interest bounding boxes which indicated what regions of the images should be searched within during the later 1D stereo matching step.

Since from this point on only rectified images were used, the homogeneous transformation from unrectified to rectified camera space, as well as the new 'rectified' camera intrinsic matrix, were also extracted from *stereoRectify* and applied when computing world to top/bottom image transformations.

Feature Detection
To extract Harris/SIFT/BRISK key points from the rectified top/bottom images, OpenCV's *cornerHarris()*, *SIFT_create()*, and *BRISK_create()* functions were used. For SIFT and BRISK, descriptors were also extracted for each key point, and the key points were rounded to the nearest pixel, so that a 1D stereo search could be employed along integer coordinates in the next step.

Stereo Matching
To find image point correspondences between top and bottom stereo images, a similar algorithm was implemented to that in Assignment 3. The main difference here was that the stereo image pair was for vertically adjacent cameras as opposed to horizontally adjacent cameras, and only certain features in the top image were used as queries, instead of performing a dense search for the full image. For each key point in the top image, the following steps are performed:

1. Pad the top and bottom images with zeros in case a sliding window would need to cover edges.
2. Skip the key point if it is outside the top image's region of interest (produced from *stereoRectify*).
3. Slide a square window of size 13x13 (experimentally chosen) over y coordinates of the bottom image (within the bottom image's region of interest) at the same x value as the key point.

4. At each position of the window, compute the sum of absolute difference (SAD) of pixel intensities (with a fixed window at the top image's key point), and store the coordinate with the lowest SAD value encountered so far during the vertical search.
5. After the vertical search, check if the best corresponding bottom image point does not have the same coordinates as the key point in the top image. If it does, reject the match (this edge case occurs for very few features because of small camera motions and due to key point pixel rounding), so that light rays later produced by triangulation are not parallel.
6. If using Harris, store the pixel intensities in the fixed window at the key point in the top image as a pseudo-descriptor which can be used during the next feature tracking step.

Feature Tracking

Next, it was required to find correspondences between features in the current time step's top image and the previous time step's top image. These correspondences would later be used to determine 3D point correspondences for point cloud alignment. To do this, descriptors were matched using OpenCV's *BFMatcher* (brute force matcher) which finds the closest descriptor match in a corresponding image to a descriptor in a query image by comparing distances between all descriptors in the corresponding image (n^2 time complexity). Only matches which were mutually accepted by both images were used. For Harris and SIFT descriptors (Harris descriptors were generated manually during stereo matching as the pixel intensities in the flattened window surrounding the key point), Euclidean distance was used as the distance metric. For BRISK features, Hamming distance was used.

Triangulation, Scalar Weighted Point Cloud Alignment, and Matrix Weighed Point Cloud Alignment

To perform triangulation of the 3D landmark points, scalar weighted point cloud alignment for motion estimate initialization, and matrix weighted point cloud alignment (non-linear least squares, i.e., NLS) for refining the motion estimate, the code from Assignment 4 was used.

However, an important improvement had to be made to the NLS algorithm, which was to check a convergence criterion (norm of the change in parameters between iteration) instead of using a fixed number of iterations (100 iterations were used here). Furthermore, a novel improvement was made to label the regressed homogenous motion matrix as viable based on a distance threshold if the algorithm did not converge. In summary, the resulting motion estimate was deemed acceptable by the visual odometry if NLS converged, or if it did not converge but the change in parameters between the last two iterations was still less than a small, fixed threshold. If neither of these criteria were met, a motion estimate of identity was used (estimate no change in body frame). These changes resulted in significantly improved trajectory estimates, since returning invalid motion estimates would typically cause the trajectory estimates to explode.

Project Evaluation and Results:

To evaluate the respective performances of the three feature detectors for stereo visual odometry, the global poses from the rover navigation data (available in "*global-pose-utm.txt*" in the form of easting, northing, altitude, and orientation) were used as references for comparison. The resulting trajectory from the visual odometry was determined by computing the global pose after every time step using the predicted homogenous body transformations between time steps (starting from the true initial position). The coordinates in the predicted and reference trajectories could then be compared using the mean squared error (MSE). Additionally, the average processing time for computing camera poses at each time step using the different feature detectors was measured to compare relative speed. These results as well as intermediate results from pre-processing, feature detection, matching, and tracking are provided in the following subsections.

Pre-processing

To verify that the un-distortion and vertical rectification was working as desired, an example pair of stereo images was plotted before and after the mapping, as shown in Figure 1. As shown by the vertical red lines drawn on the rectified images, it can be qualitatively seen that the x values of corresponding image feature in the top and bottom images is the same, and therefore a vertical 1D search can be used for stereo matching.

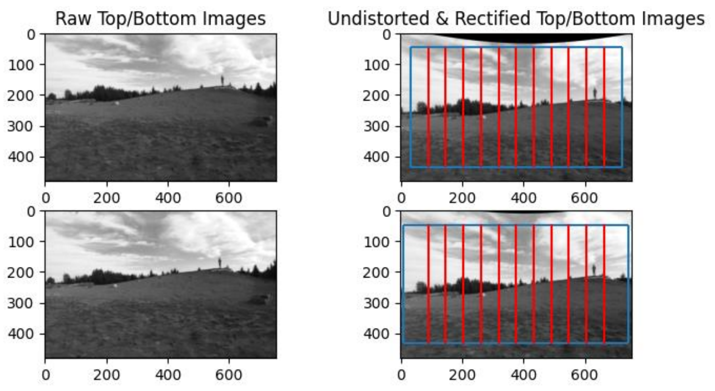

Figure 1: Un-distortion and rectification on a top/bottom stereo image pair. The blue boxes indicate regions of interest for 1D stereo matching, and the red lines show that the new images are fronto-parallel in the vertical direction.

Feature Detection and Stereo Matching

In Figure 2, the red dots on the top/bottom images indicate the features that were detected by each respective feature detector. As shown, Harris features are the densest and are solely concentrated on the tree line while there are some SIFT and BRISK key points on the ground and sky (mostly for SIFT). This follows intuition since Harris and BRISK are corner-based and are less likely to select rounder key points. To qualitatively verify that the stereo matching was working, the matches were plotted as the green dots. The green dots are aligned with the red dots everywhere in the region of interest of the top image since those are the query points. In the bottom images, most green dots are located at the appropriate y coordinate. A few exceptions in the bottom image can be seen in the sky or ground for all feature types.

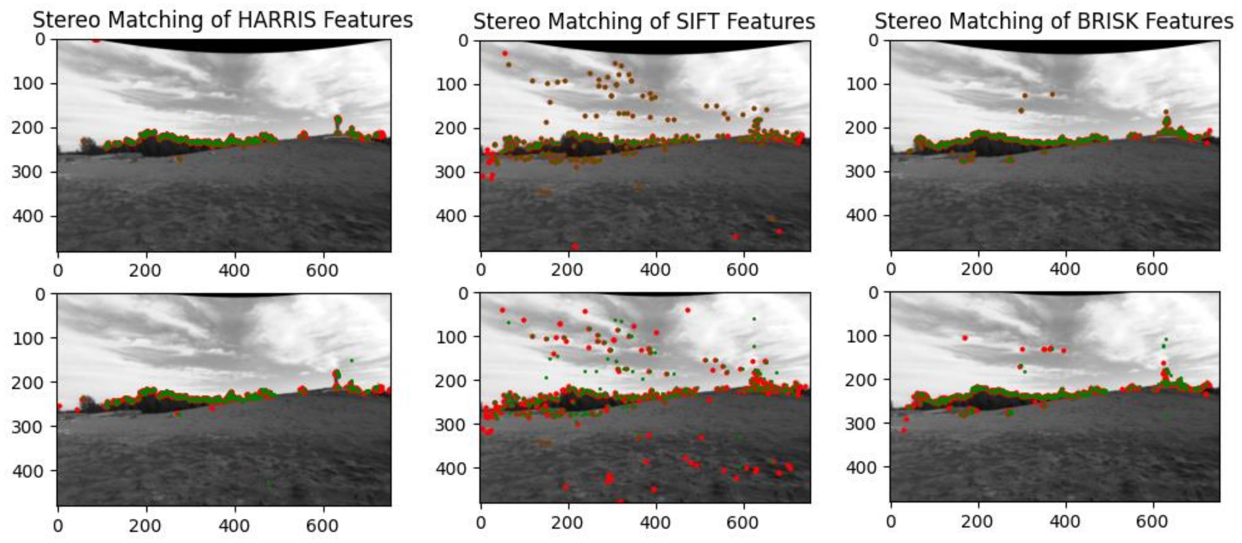
Stereo Matching of HARRIS Features
Stereo Matching of SIFT Features
Stereo Matching of BRISK Features

Figure 2: Feature detection and stereo matching using (a) Harris features (b) SIFT features and (c) BRISK features. The red dots indicate the features that were detected in the top and bottom images, and the green dot indicate all the 2D point correspondences after stereo matching.

Feature Tracking

The functionality of feature tracking was also visually confirmed by examining example matches in the first 2 time-steps. As shown in Figure 3, the descriptor matching (of key points in the top stereo image) was successful for most points, and there were very few outliers, mainly because of the mutual check that was employed (only descriptor matches that were agreed on by both images were retained).

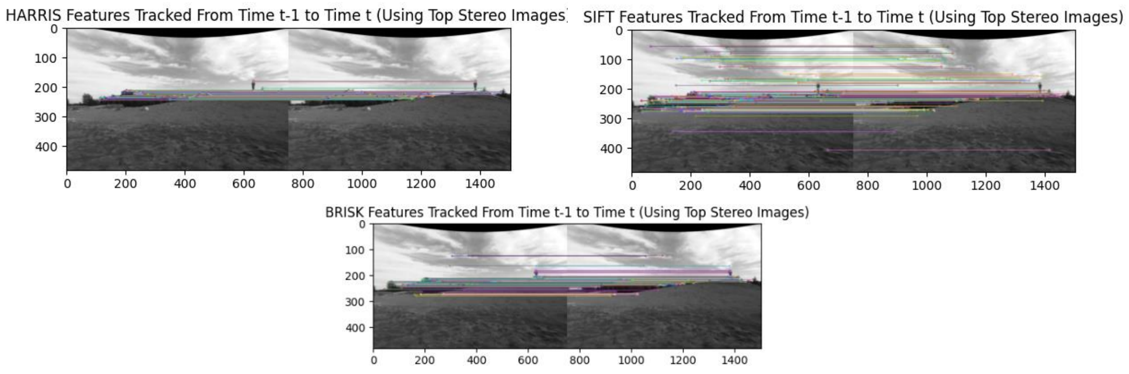
HARRIS Features Tracked From Time t-1 to Time t (Using Top Stereo Images)
SIFT Features Tracked From Time t-1 to Time t (Using Top Stereo Images)
BRISK Features Tracked From Time t-1 to Time t (Using Top Stereo Images)

Figure 3: Matched features between top stereo images at time steps t-1 and t for (a) Harris (b) SIFT and (c) BRISK.

<u>Trajectory Comparisons</u>

The main evaluation of the visual odometry was performed for each feature descriptor by comparing the estimated trajectories with the ground truth trajectories from *global-pose-utm.txt* for the first 80 seconds of Run 1. Figure 4 displays the x vs. y plot (trajectory) and the coordinate plots vs. time. Also, Table 1 shows the resulting MSE for each coordinate. Qualitatively, the Harris feature detectors resulted in the closest trajectory estimates to the reference when considering direction changes, especially for y and z coordinates. However, all feature detectors struggled to approximate the x coordinate. It is hypothesized that this is due to the rapid change in x-direction at around t=35, which all the estimated trajectories missed. Harris and BRISK somewhat followed the direction change at around t=75. SIFT results in poor estimated trajectories, apart from the z-coordinate which it follows the closest. It is hypothesized that this is a result of SIFT detecting more round features in the ground and sky (shown previously), and that those features lead to more potential outlier matches that result in less frequent convergence from NLS.

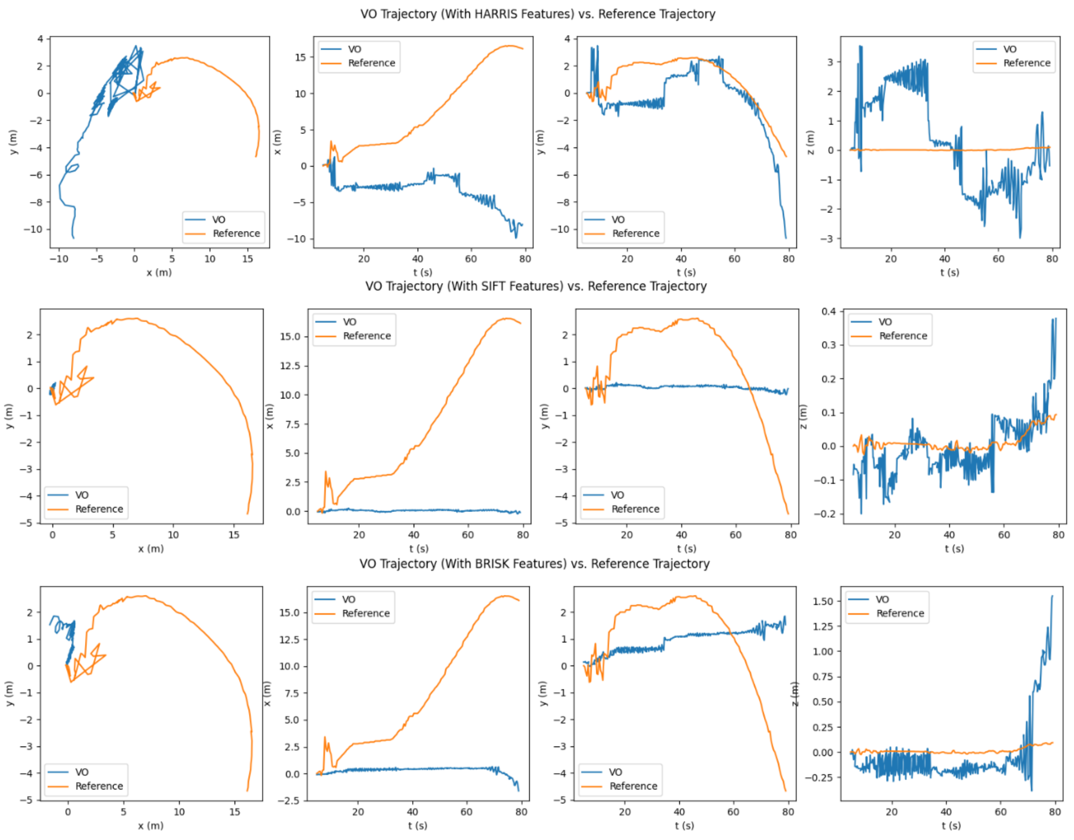

Figure 4: Resulting trajectories from the visual odometry using the different feature detectors, and the comparisons to the ground truth trajectory.

Table 1: MSE along x, y, z coordinates and average feature usage time per time step

Feature Detector	**MSE-x (m^2)**	**MSE-y (m^2)**	**MSE-z (m^2)**	**Avg. time/iter (s)**
Harris	166.97	3.55	2.34	0.61
SIFT	86.6	3.9	0.0	1.24
BRISK	83.5	4.0	0.1	1.39

From Table 1, Harris has the largest error for the x coordinate, and all feature detectors have similar errors for the y and z coordinates. However, these numbers do not indicate the direction changes which Harris was more accurately able to follow. Furthermore, Harris features resulted in the fastest iterations.

These results were extremely surprising as it was expected that SIFT and BRISK features would lead to more accurate trajectory estimates because of them being scale and affine-transformation invariant. On the contrary, they were not able to produce trajectories that were significantly more accurate than Harris. This is believed to be a result of the greater number of meaningful features that were detected. It was demonstrated earlier that the majority of the Harris features were located at distinct object corners while some SIFT and BRISK features were in more sparse regions like the ground or sky. Since RANSAC was not employed in the pipeline, it is possible that many of those features led to outlier matches that should have been eliminated. Furthermore, it was also surprising that BRISK did not have the fastest iteration time since it was expected that the use of the hamming distance would speed up the iterations for BRISK. This can be explained from the fact that the limiting speed factor was the stereo matching, and that SIFT and BRISK consistently produced 200-400 features, while Harris would vary between 50-500 features depending on the image. On average, SIFT and BRISK potentially had more features to match.

Besides piecing together all the different components of the visual odometry pipeline, the most significant challenge was producing trajectories that did not grow exponentially in magnitude (or the NLS not converging for most iterations). This was resolved using the technique described in the methodology section of forcing the homogenous motion estimate matrix to be identity when NLS did not converge (it did converge for the vast majority of iterations however) to a parameter change below a fixed threshold. As a result, pose updates which were not reasonable did not result in significant accumulated errors, and the convergence rate was significantly higher.

References:

[1] C. Harris & M. Stephens, “A Combined Corner and Edge Detector”, in *Alvey Vision Conference*, Plessey Research Roke Manor, United Kingdom, 1988.

[2] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 2, no. 60, pp. 91–110, Nov. 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.

[3] S. Leutenegger, M. Chli, and R. Y. Siegwart, “BRISK: Binary Robust Invariant Scalable Keypoints,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV’11)*, Barcelona, Spain, Nov. 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[4] Y. Cheng, M. W. Maimone, and L. H. Matthies, “Visual Odometry on the Mars Exploration Rovers,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, Big Island, Hawaii, USA, Oct. 2005, pp. 903–910. DOI: 10.1109/ICSMC.2005.1571261.

[5] O. Lamarre, O. Limoyo, F. Maric, and J. Kelly, “The Canadian Planetary Emulation Terrain Energy-Aware Rover Navigation Dataset,” *The International Journal of Robotics Research*, p. 0278364920908922, 2020.

[6] H. Bay, T. Tuytelaars, and L. V. Gool, “SURF: Speeded Up Robust Features,” in *Computer Vision — ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7–13, 2006, Proceedings, Part I*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951/2006, Berlin, Germany: Springer, Jun. 2006, pp. 404–417. DOI: 10.1007/11744023_32.

[7] E. Rublee et al., “ORB: An efficient alternative to SIFT or SURF,” in *IEEE International Conference on Computer Vision*, Barcelona, ICCV, 2011, pp. 2564-2571.

[8] Khan, S.; Saleem, Z. A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. In *Proceedings of the International Conference on Computing, Mathematics and Engineering Technologies*, Sukkur, Pakistan, 3–4 March 2018.