Optical Flow Odometry with Robustness to Self-shadowing

Neal Seegmiller and David Wettergreen

***Abstract*— An optical flow odometry method for mobile robots using a single downward-looking camera is presented. The method is robust to the robot's own moving shadow and other sources of error. Robustness derives from two techniques: prevention of feature selection on or near shadow edges and elimination of outliers based on inconsistent motion. In tests where the robot's shadow dominated the image, prevention of feature selection near shadow edges allowed accurate velocity estimation when outlier rejection alone failed. Performance was evaluated on two robot platforms and on multiple terrain types at speeds up to 2 m/s.**

I. Introduction

Accurate position estimation remains a challenge for planetary exploration robots in GPS-denied and high-slip environments. The Mars Exploration Rovers demonstrated the usefulness of visual odometry in these conditions, but also highlighted two challenges: computational cost and errors induced by shadows [1]. This paper presents an optical flow odometry method for resource-constrained planetary rovers. The method is uniquely designed for efficiency and robustness to the robot's own moving shadow.

A. Challenges and Related Work

The challenges of computational cost and self-shadowing errors in visual odometry are well documented.

1) Computational Cost: Many recent visual odometry methods use correspondences in stereo image pairs to triangulate points in 3D and estimate 6 DOF changes in pose [1]–[3]. These methods can be accurate up to 1% of the distance traveled [2], but can also be computationally expensive. On the Mars Exploration Rovers (MER) for example, limited computational resources meant that a single odometry update took up to 3 minutes to process [1].

A computationally cheaper approach is to estimate velocity from the optical flow of the visual texture of the ground, as viewed by a single downward-looking camera. Recent examples include [4]–[8].

2) Self-shadowing Errors: A common limitation in visual odometry is sensitivity to lighting conditions. Specifically, inadvertent tracking of the robot's own moving shadow can produce an erroneous velocity estimate [1], [2], [9]. Human operators had to take the MER's shadow into account when deciding whether or not to use visual odometry during a drive [1].

Manuscript received March 28, 2011.

N. Seegmiller is a Ph.D. candidate at the Carnegie Mellon University Robotics Institute nseegmiller@cmu.edu

D. Wettergreen is a Research Professor at the Carnegie Mellon University Robotics Institute dsw@ri.cmu.edu

One potential solution is to place the camera under the vehicle such that the robot's shadow covers the entire field of view (as suggested by [9]); however, shadow edges may still be visible at sunrise/sunset and the reduced camera height limits the top vehicle speed that can be tracked.

Another potential solution is the use of onboard illumination; however, such illumination wastes the rover's limited power. Furthermore, controlled lighting in outdoor environments is challenging because the position, shape, and intensity of the robot's shadow vary with the relative position of the sun.

A tracking algorithm impervious to moving shadow edges is the best solution, but common outlier rejection methods can be insufficient. Most visual odometry algorithms identify inliers as the largest set of feature matches that meet a rigidity constraint in world coordinates (e.g. [2]). Unfortunately, when using a downward-looking camera shadow features might dominate the image, forming the largest set that meets the rigidity constraint (see Section III-B).

B. Approach

The presented algorithm seeks to address both the challenge of computational cost and self-shadowing errors.

The efficiency of the optical flow odometry algorithm makes it well suited for planetary rovers with limited computing resources; however, additional applications could benefit from the algorithm's speed. The algorithm provides high-frequency, low-latency velocity measurements that could enable immediate detection of (and reaction to) wheel slip. Furthermore, the top vehicle speed which can be tracked is not limited by computation time (see Section II-A). The presented method was tested at faster vehicle speeds (2 m/s) than all cited methods in which vehicle speed is stated explicitly [1], [3]–[5], [7]–[9].

Robustness to shadows and other sources of error derives from two techniques: a dynamic mask that prevents features from being selected on or near shadow edges, and robust selection of the largest set of features (inliers) that track as a rigid pattern between frames. The primary contribution of the presented method is the prevention of feature selection near shadow edges. This, in combination with the commonly applied inlier selection step, enables accurate velocity estimation even in severe cases where the robot's shadow dominates the field of view.

II. Explanation of the Algorithm

The optical flow odometry method is summarized in Algorithm 1. Sections II-A to II-F describe major steps of the algorithm in sequential order. Robustness to self shadowing derives from the steps described in II-B and II-E.

Algorithm 1 Optical Flow Odometry

1: **loop**
2: $I_{prev} \leftarrow$ grabImage
3: makeFeatureSelectionMask
4: selectFeatures
5: $I_{curr} \leftarrow$ grabImage
6: calculateOpticalFlow
7: transformFeaturesToRobotFrame
8: getInliers
9: **if** $numInliers > threshold$ **then**
10: estimateVelocity
11: **end if**
12: **end loop**

A. Grabbing Images at Fast Vehicle Speeds

The timing of frame capture is critical when tracking fast vehicle speeds. Many visual odometry algorithms capture one image per iteration (I_{curr} in one iteration is reused as I_{prev} in the subsequent iteration). The advantages are that one velocity estimate is produced per frame and features can be reused over multiple iterations. The key disadvantage is that frames may be dropped between I_{prev} and I_{curr} as all computation must occur between their capture times. If frames are dropped, the displacement of features between I_{prev} and I_{curr} may become too large to track when driving at high speeds.

If feature selection time is sufficiently bounded, the presented algorithm guarantees that I_{prev} and I_{curr} are captured at the maximum frame rate, thereby minimizing the risk of tracking failure. Feature selection, even though performed every iteration, does not contribute to latency because it is performed before rather than after the capture of I_{curr}. (see Algorithm 1, lines 2-5)

The top vehicle speed that can be tracked for a downward-looking camera (mounted normal to the ground) is given by:

$$maxVel = \frac{maxDisp \cdot f \cdot 2h \cdot tan(FOV/2)}{imsize} \quad (1)$$

$$imsize \geq \lambda \cdot maxDisp \quad (2)$$

In the above equations, $maxDisp$ is the maximum displacement (in pixels) that can be tracked, which depends on the terrain texture and $imsize$ (the image size in pixels in the direction of travel). In the LATUV test (Section III-B), displacements up to 25 pixels were tracked successfully on concrete. f is the framerate (in Hz), h is the height of the camera above the ground (in meters), and FOV is the camera's angular field of view. The constraint on $imsize$ in (2) ensures sufficient overlap between frames (for example, $\lambda = 10$ ensures at least 90% overlap). Because the algorithm guarantees image capture at the maximum framerate, computation time does not limit the top speed that can be tracked but only the update frequency.

B. Excluding Shadow Edges in Feature Selection

As stated in the introduction, tracked features on or near the robot's own shadow will cause error in the velocity estimate. We have developed a technique to dynamically mask shadow edges to prevent these features from being selected. (Algorithm 1 lines 3-4)

First, an image pyramid is constructed by filtering and downsampling I_{prev}. The base level in the pyramid is the full resolution image and each subsequent level is half the width and height of its parent level. The smallest image in the pyramid should be no less than 80×60 pixels. For smoothing and reduced computation, the smallest image is used for mask creation.

1) Determining the Presence of a Shadow: Binary segmentation is performed on the downsampled image, sorting the 1D pixel intensities into two clusters. We chose K-means as the clustering method for its simplicity and speed. A shadow exists in the image if pixel intensities have a bimodal distribution. The following test for bimodality is used:

$$shadowDetected = \begin{cases} 1 & if\ |c_1 - c_2| > \lambda(\sigma_1 + \sigma_2) \\ 0 & otherwise \end{cases} \quad (3)$$

c_1 and c_2 are the centers of the K-means clusters and σ_1 and σ_2 are their standard deviations. The scalar λ is tuned to prefer false positives over false negatives (see Fig. 1, Fig. 2). In the case of a false positive (shadow detected in a shadowless image), some edges in the image may be wrongly excluded, but sufficient features will still be found. The harmlessness of false positives is crucial, as multimodal distributions can occur for reasons other than shadows.

2) Detecting Shadow Edges: If a shadow is present, its edges are detected using a Canny detector with thresholds proportional to the difference between the K-means cluster centers ($cannyThresh \propto |c_1 - c_2|$). However, not all detected edges necessarily belong to the shadow. The raw Canny edge detection output is parsed as follows.

First, the labels produced by K-means are reshaped into a binary image shown in Fig. 3(a) (black denotes shadow, white denotes light).

Next, erroneous shadow segments are removed. Here, "segment" refers to groups of neighboring (4-connected) shadow pixels. The robot's shadow must reach the boundaries of the image; all isolated "shadow" spots must in fact be dark spots on the ground which are acceptable to track. Using an efficient algorithm that grows shadow segments seeded with boundary pixels, only those touching the image boundaries are preserved. The result of this step is shown in Fig. 3(b).

A *shadow edge mask* is created by finding the edges in the cleaned-up binary image (using a fast gradient method with no edge thinning) and dilating with a square 3×3 structuring element (Fig. 3(c)). The shadow edge mask is then applied to the raw Canny edge detection output. The raw and parsed edge detection images are shown in Fig. 3(d) and 3(e).

3) Creating the Dynamic Feature Selection Mask: Finally, the parsed shadow edge image is dilated to produce the feature selection mask (Fig. 3(f)). The radius of the dilation structuring element is given by:

$$radius \propto \frac{maxDisp + windowSize/2}{2^L} \quad (4)$$

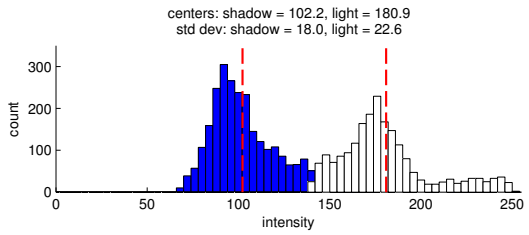

Fig. 1. Histogram of intensities corresponding to the image in Fig. 3. Bars are colored according to K-means labels (blue: shadow, white: light). K-means centers are drawn as vertical red lines

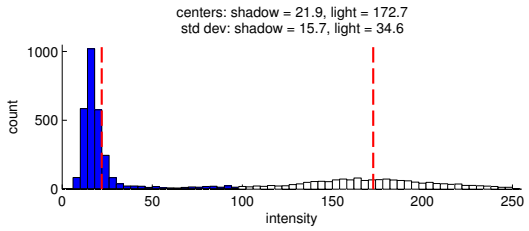

Fig. 2. Histogram of intensities corresponding to the image in Fig. 4. (refer to Fig. 1 caption for explanation.)

$maxDisp$ is the maximum feature displacement between I_{prev} and I_{curr} (in pixels), $windowSize$ refers to the Lucas-Kanade window (Section II-C), and L is the level in the pyramid at which the feature selection mask is created. Features must not be selected on the shadow edge or so near that the window tracks into the edge.

4) Feature Selection: Features are selected using a Harris corner detector. Features must not be selected where disallowed by the feature selection mask and a minimum distance between features should be enforced. A region of interest for feature selection is specified in the center of the image because features too near the boundaries are likely to track out of the field of view (see Fig. 3(g)).

Figures 1-4 illustrate two examples of the entire feature selection process. In Fig. 3, note how features are selected within and without the shadow but not on its edges. In Fig. 4(d), note how edges are detected on leaves but are removed from the feature selection mask, rightly allowing them to be tracked. Without parsing of the raw Canny edge detection output, the feature selection mask would be too restrictive on highly featured terrain.

C. Calculating Optical Flow

Pyramidal Lucas-Kanade tracking is used to track features over large displacements (which occur at high speeds) [10]. Parameters such as window size and the number of levels are limited primarily by the size of the full resolution image. (Algorithm 1 line 6)

D. Transforming Features to the Robot Frame

Accurate velocity estimates require accurate internal and external camera calibration. (Algorithm 1 line 7)

1) Internal Calibration: Internal parameters including focal length, principal point, and distortion coefficients are required to undistort pixel coordinates.

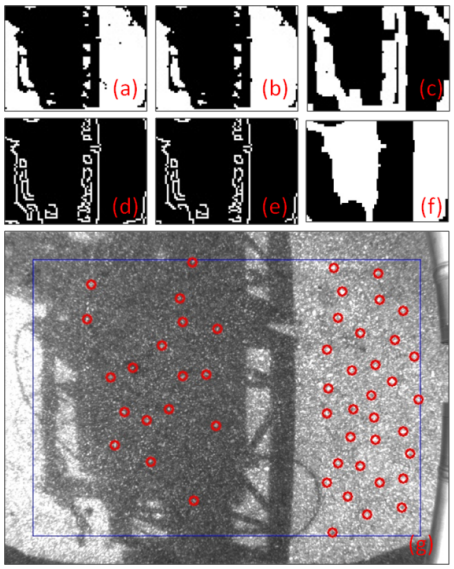

Fig. 3. Feature selection on concrete with a complex shadow cast by the LATUV. (a) Binary image of K-means labels. black: shadow, white: light. (b) Label image with shadows not touching borders removed. (c) Shadow edge mask. (d) Raw Canny edge detection output. (e) Shadow edge image after applying shadow edge mask. (f) Feature selection mask. (g) Original image with features circled. The blue box shows the region of interest for feature selection

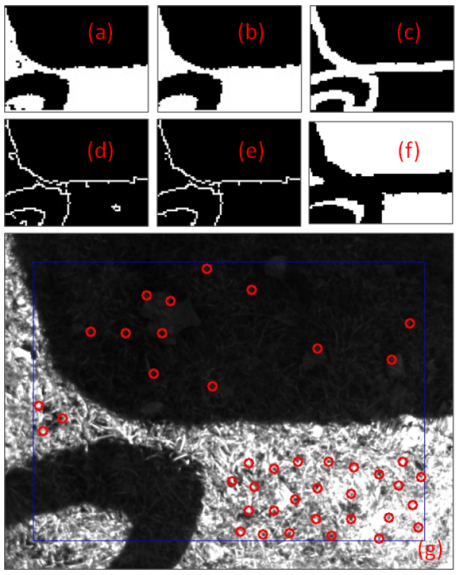

Fig. 4. Feature selection on grass with shadows cast by the wheel and body of the Zoë rover. (refer to Fig. 3 caption for explanation.)

2) External Calibration: Assuming the ground is locally flat (i.e. all features lie in a plane) then the conversion from undistorted pixel coordinates to the robot frame is a homography. The closed form equation for the homography can be computed from the pinhole camera model:

$$sp = KMP = K[R \mid t]P \tag{5}$$

where p and P denote the point coordinates in the camera and robot frame respectively. K is the camera matrix and s signifies the equality is up to scale. M is composed of a

rotation matrix and translation vector representing the 6 DOF pose of the robot frame with respect to the camera frame, $\underline{\rho}_R^C$. Because all points lie in a plane, point z coordinates are a linear function of x and y, and M can be converted into a 3×3 matrix:

$$MP = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z = ax + by + c \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} r_{11} + r_{13}a & r_{12} + r_{13}b & t_1 + r_{13}c \\ r_{21} + r_{23}a & r_{22} + r_{23}b & t_2 + r_{23}c \\ r_{31} + r_{33}a & r_{32} + r_{33}b & t_3 + r_{33}c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (6)$$

M^* denotes the 3×3 matrix and P^* denotes the 3×1 coordinate vector in (6). The homography equation is:

$$sp = HP^*, \;\; sP^* = H^{-1}p \quad (7)$$

$$H = KM^* \quad (8)$$

$\underline{\rho}_R^C$ (and corresponding H) are calibrated by least-squares minimization of reprojection error for a set of correspondences (for which p and P are known).

$$\underset{\underline{\rho}_R^C}{\operatorname{argmin}} \|p - HP^*\|^2, \;\; H = f\left(\underline{\rho}_R^C\right) \quad (9)$$

Using this technique, the transformation can be accurately calculated even if the camera is off-normal to the ground. This may be useful at high speeds as it increases the ground area within the field of view.

E. Obtaining Inliers

In this step (Algorithm 1 line 8), the largest set of features that track consistently (as a rigid pattern) between I_{prev} and I_{curr} is obtained. We chose RANSAC for its simplicity, but other inlier selection methods are possible. The following sections explain the steps performed at each iteration of RANSAC.

1) Select Sample Set: The number of features selected and tracked is denoted by n. A sample of n_s features is randomly selected. In our current implementation n = 40-50 and n_s = 3. The set of sample indices is denoted by s.

2) Normalize about Centroids: Calculate the centroids of the sample points in I_{prev} and I_{curr} ($\bar{p}$ and $\bar{c}$ respectively):

$$\bar{x} = \frac{\sum_{i=1}^{n_s} x^{s(i)}}{n_s}, \quad \bar{y} = \frac{\sum_{i=1}^{n_s} y^{s(i)}}{n_s} \quad (10)$$

$$\bar{p} = \begin{bmatrix} \bar{x}_p \\ \bar{y}_p \end{bmatrix}, \;\; \bar{c} = \begin{bmatrix} \bar{x}_c \\ \bar{y}_c \end{bmatrix} \quad (11)$$

Express sample points (p) and the set of all points (P) in I_{prev} about the sample centroid in I_{prev} ($\bar{p}$). The "prime" symbol (e.g. p') is used to denote points with the sample centroid subtracted.

$$p' = \begin{bmatrix} x_p^{s(1)} - \bar{x}_p & \dots & x_p^{s(n_s)} - \bar{x}_p \\ y_p^{s(1)} - \bar{y}_p & \dots & y_p^{s(n_s)} - \bar{y}_p \end{bmatrix} \quad (12)$$

$$P' = \begin{bmatrix} x_p^1 - \bar{x}_p & \dots & x_p^n - \bar{x}_p \\ y_p^1 - \bar{y}_p & \dots & y_p^n - \bar{y}_p \end{bmatrix} \quad (13)$$

Do the same for the sample points (c) and the set of all points (C) in I_{curr} about the sample centroid in I_{curr} ($\bar{c}$).

3) Calculate the Rotation: First, calculate the covariance matrix Σ:

$$\Sigma = \frac{c' p'^T}{n_s} \quad (14)$$

Then calculate the least-squares rotation matrix from the SVD decomposition of the covariance matrix [11]:

$$USV^T = \text{SVD}(\Sigma) \quad (15)$$

$$R = UV^T \quad (16)$$

4) Calculate Error and Obtain Inliers: Error is calculated by:

$$Error = C' - RP' = \begin{bmatrix} x_{err}^1 & \dots & x_{err}^n \\ y_{err}^1 & \dots & y_{err}^n \end{bmatrix} \quad (17)$$

A point is classified an inlier if the square distance error ($x_{err}^2 + y_{err}^2$) is less than a threshold (e.g. a few millimeters). These steps are performed up to a maximum number of iterations, after which the best set of inliers is returned and outliers are eliminated. This RANSAC technique will reject erroneous features from any source (shadows, out of plane or moving objects) *as long as those features do not form the largest consistent set*.

F. Estimating Velocity

After all features are transformed to the robot frame and outliers are rejected, translation and rotation can be estimated using the following equation:

$$v_{pt} = V_{robot} + \omega \times r_{pt} \quad (18)$$

Point velocity (v_{pt}) is the *negative* of feature displacement from I_{prev} to I_{curr} (in meters) divided by the time difference between frames (in seconds). From this equation for a single feature, a matrix equation for all features can be derived:

$$\begin{bmatrix} 1 & 0 & -r_y \\ 0 & 1 & r_x \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ \vdots \end{bmatrix} \quad (19)$$

Note that each feature corresponds to two rows, and so at least two features are required or the system will be underdetermined. In practice the system is always overdetermined (5-50 features) and solved using the pseudoinverse. This equation was used by Lee and Song for velocity estimation using multiple optical mouse sensors [7]. Note that this equation assumes no non-holonomic constraints and, as a result, can detect lateral slip.

III. Experimental Results

A. Experimental Setup

Tests were performed on two robots (the LATUV and the Zoë rover) on multiple terrain types in outdoor sunlit conditions. The LATUV, or Lunar All-terrain Utility Vehicle, was designed for speed and mobility on the Moon [12]. Zoë is an exploration robot that surveyed microbiological life in the Atacama Desert [13].

A Point Grey Flea2 black and white camera was selected with a maximum resolution of 640×480 pixels and a maximum frame rate of 60 Hz. The lens field of view was 77×57°. The camera was mounted facing downward, but 15° off-normal on Zoë. OpenCV's `goodFeaturesToTrack()` and `calcOpticalFlowPyrLK()` were used for feature selection and tracking. Processing times were recorded on a Dual-core 2.2 GHz laptop computer.

B. Lunar All Terrain Utility Vehicle (LATUV) Test

In this test, the LATUV was commanded to drive a 5 meter radius arc at 2 m/s. The test was performed on a paved surface where the robot repeatedly drove into and out of direct sunlight. Because of the LATUV's open frame construction it cast a large, complex shadow in the center of the camera's field of view (see Fig. 3). Note that with no robustness technique enabled all velocity measurements fail continually (see Fig. 5 (top)). With only RANSAC enabled (middle) the velocity estimate jumps between zero and 2 m/s, depending on whether more features lie on or off the shadow edges. Jumps in V_y also correspond to large jumps in the angular velocity. With both the dynamic feature selection mask and RANSAC enabled (bottom) the algorithm consistently tracks the correct V_y velocity at 2 m/s. V_x and the angular velocity are also stable.

Clearly, performance is greatly improved by supplementing RANSAC with the dynamic feature selection mask in the presence of the robot's shadow. Feature selection clocks at 9 ms, including 3 ms for creation of the feature selection mask. Computation is sufficiently fast to select features between the capture of I_{prev} and I_{curr} at 60 Hz without risk of dropping frames. Velocity updates are produced at 45 Hz.

C. Zoë Rover Test

Even though the camera was mounted underneath Zoë's solar panels (see Fig.7), wheel and body shadow edges dominated the field of view (see Fig. 4). Eight random test paths (150-186 m in length) were driven on a grassy field. Ground truth was provided by a Novatel SPAN GPS/INS unit.

Table I presents the mean and standard deviation position error for 20, 50, 100, and 150 m path segments. These values are calculated using the same method as [9]. First, time-corresponding odometry and ground truth path segments are extracted. Then the odometry path segment is translated and rotated to align with the starting ground truth position and heading, and the Euclidean distance error of the endpoint is calculated. Next, we move one meter forward in the ground truth data and repeat the process until the end of the path is reached.

When using both the linear and angular optical flow velocity estimates, odometry position error is approximately 4-6% of distance travelled (see the "using odometry yaw" columns in Table I). Mean heading error after 150 m is 5.4° with a standard deviation of 3.0°. Alternatively, when using optical flow linear velocity measurements but obtaining heading

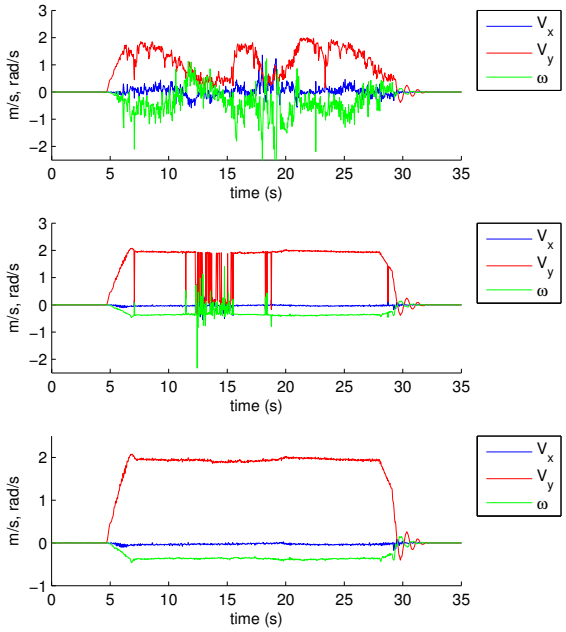

Fig. 5. Calculated linear and angular velocities with various robustness techniques enabled. (Top) No robustness technique enabled. (Middle) Only RANSAC enabled. (Bottom) Both dynamic feature selection mask and RANSAC enabled

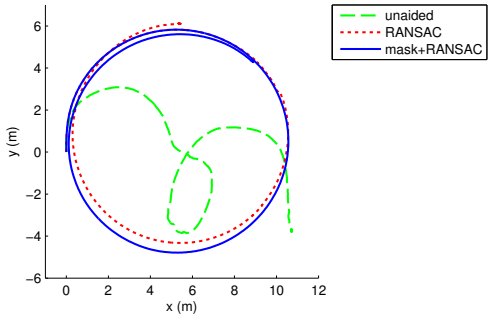

Fig. 6. Integrated path based on velocities in Fig. 5. Note that with both the dynamic feature selection mask and RANSAC enabled the path is a correct 5 m radius arc

from an Inertial Measurement Unit, odometry position error drops to 2-3% of distance traveled (see the "using IMU yaw" columns). Because the current objective is to detect wheel slip on planetary rovers and not to replace their IMU, use of the more accurate inertial yaw measurement is preferred. The largest source of odometry error is likely breaking of the planar terrain assumption.

Finally, despite extreme self-shadowing (as seen in Fig. 4) no erroneous zero-velocity measurements occurred in any of the experiment paths with the feature selection mask enabled (see Fig. 9).

IV. Conclusions and Future Work

The presented algorithm addresses two challenges in visual odometry: computational cost and self-shadowing errors. While the accuracy of the presented optical flow algorithm is slightly lower than stereo visual odometry methods (e.g.

TABLE I

OPTICAL FLOW ODOMETRY POSITION ERROR

Distance traveled	# of data	Pos. error using IMU yaw		Pos. error using odometry yaw[a]	
		μ (m)	σ (m)	μ (m)	σ (m)
20 m	847	0.88	0.50	1.09	0.60
50 m	839	1.43	0.62	2.40	1.35
100 m	457	2.37	0.59	6.05	2.73
150 m	74	3.35	0.99	9.61	3.49

[a] yaw obtained by integrating optical flow angular velocity, ω

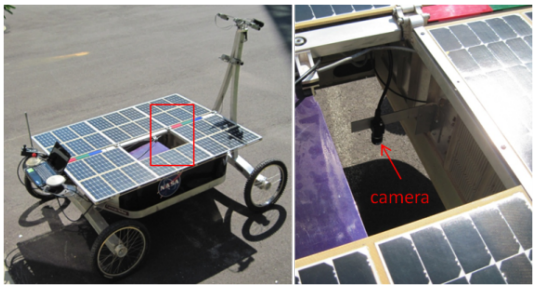

Fig. 7. Photograph of camera placement on the Zoë Rover. The missing solar panel was replaced during testing.

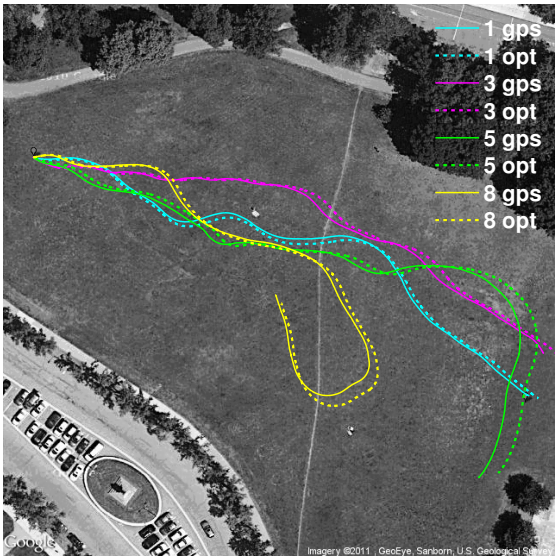

Fig. 8. Plot of four representative test paths on a scaled satellite image. Solid lines denote GPS ground truth, dashed lines denote optical flow odometry path (using inertial yaw measurement).

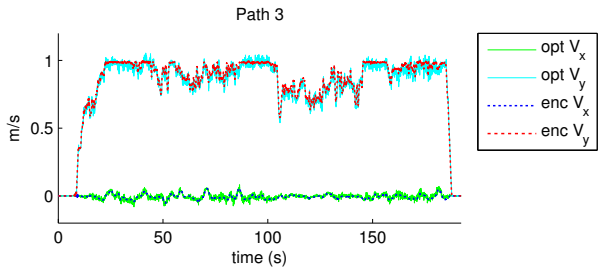

Fig. 9. Plot of optical flow V_x and V_y velocity estimates for test path 3. Velocities were also computed using wheel encoder and steer angle measurements for comparison. The two velocity measurements match closely in this low slip test.

[2]), the algorithm's computational efficiency makes it well suited for resource-constrained planetary exploration robots, as highlighted by the Mars Exploration Rovers. Furthermore, the algorithm is capable of tracking vehicle speeds at (and potentially above) 2 m/s on sufficiently textured terrain.

In multiple tests on two robots and varied terrain the presented algorithm demonstrated robustness to the robot's own shadow and other sources of error. In tests where the robot's shadow dominates the image, prevention of feature selection on shadow edges allowed accurate velocity estimation when outlier rejection alone failed. Robustness to self-shadowing allows great freedom in camera placement and eliminates the need to spend limited power on artificial illumination.

Future work will focus on utilizing the algorithm's high-frequency velocity updates to immediately quantify and respond to wheel slip.

ACKNOWLEDGMENT

This research was made with Government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a. The Zoë rover was built with support from NASA, NAG5-12890. The LATUV was designed and built by and ProtoInnovations LLC.

REFERENCES

[1] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars exploration rovers," *Journal of Field Robotics*, vol. 24, pp. 169–186, Mar. 2007.

[2] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 3946–3952.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, 2006.

[4] M. Dille, B. Grocholsky, and S. Singh, "Outdoor downward-facing optical flow odometry with commodity sensors," in *Proc. Field and Service Robotics*, Jul. 2009.

[5] X. Song, Z. Song, L. Seneviratne, and K. Althoefer, "Optical flow-based slip and velocity estimation technique for unmanned skid-steered vehicles," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 101–106.

[6] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision," in *Proc. IEEE International Conference on Robotics and Automation*, Apr. 2005.

[7] S. Lee and J. Song, "Mobile robot localization using optical flow sensors," *International Journal of Control, Automation, and Systems*, vol. 2, pp. 485–493, Dec. 2004.

[8] C. McCarthy and N. Barnes, "Performance of optical flow techniques for indoor navigation with a mobile robot," in *Proc. IEEE International Conference on Robotics and Automation*, Apr. 2004, pp. 5093–5098.

[9] N. Nourani-Vatani, J. Roberts, and M. Srinivasan, "Practical visual odometry for car-like vehicles," in *Proc. IEEE International Conference on Robotics and Automation*, May 2009, pp. 3552–3557.

[10] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm," 2000.

[11] J. Challis, "A procedure for determining rigid body transformation parameters," *Journal of Biomechanics*, vol. 28, pp. 733–737, Jun. 1995.

[12] L. Pedersen, M. Allan, V. To, H. Utz, W. Wojcikiewicz, and C. Chautems, "High speed lunar navigation for crewed and remotely piloted vehicles," in *Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Aug. 2010.

[13] D. Wettergreen *et al.*, "Second experiments in the robotic investigation of life in the Atacama desert of Chile," in *Proc. 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Sep. 2005.