Real-Time Stereo Visual Odometry for Autonomous Ground Vehicles

Andrew Howard

***Abstract*— This paper describes a visual odometry algorithm for estimating frame-to-frame camera motion from successive stereo image pairs. The algorithm differs from most visual odometry algorithms in two key respects: (1) it makes no prior assumptions about camera motion, and (2) it operates on dense disparity images computed by a separate stereo algorithm. This algorithm has been tested on many platforms, including wheeled and legged vehicles, and has proven to be fast, accurate and robust. For example, after 4000 frames and 400m of travel, position errors are typically less than 1m (0.25% of distance traveled). Processing time is approximately 20ms on a 512x384 image. This paper includes a detailed description of the algorithm and experimental evaluation on a variety of platforms and terrain types.**

I. Introduction

Localization is a key capability for autonomous ground vehicles, and is typically performed using a combination of wheel odometry (from joint encoders) and inertial sensing (gyroscopes and accelerometers). This approach has two limitations, however: inertial sensors are prone to drift, and wheel odometry is unreliable in rough terrain (wheels tend to slip and sink). Visual odometry, which estimates vehicle motion from a sequence of camera images, offers a natural complement to these sensors: it is insensitive to soil mechanics, produces a full 6DOF motion estimate, and has lower drift rates than all but the most expensive IMUs.

This paper describes a real-time stereo visual odometry algorithm that is particularly well-suited to ground vehicle applications. Our motive for adopting a stereo, rather than monocular, approach derives from the observation that many ground vehicles are already equipped with stereo cameras for obstacle detection and terrain sensing, and that the stereo range data produced by these systems can be exploited for visual odometry. That is, the combination of stereo ranging and stereo visual odometry is generally faster and more reliable than stereo ranging followed by monocular visual odometry.

The basic algorithm is as follows: for a given pair of frames, (1) detect features in each frame (corner detection), (2) match features between frames (sum-of-absolute differences over local windows), (3) find the largest set of self-consistent matches (inliers), and (4) find the frame-to-frame motion that minimizes the re-projection error for features in the inlier set. The inlier detection step (3) is the key distinguishing feature of the algorithm. The feature matching stage inevitably produces some incorrect correspondences, which, if left intact, will unfavorably bias the frame-to-frame motion estimate. A common solution to this problem is to use a robust estimator that can tolerate some number of false matches (e.g., RANSAC [4]). In our algorithm, however, we adopt an approach described by Hirschmüller et al. [7], and exploit stereo range data at the inlier detection stage. The core intuition is that the 3D locations of features must obey a rigidity constraint, and that this constraint can be used to identify sets of features that are mutually consistent (i.e., a clique) prior to computing the frame-to-frame motion estimate.

This approach can be best described as *inlier detection* rather than *outlier rejection*, and offers a number of advantages. First, it is extremely robust to false matches, which relaxes some of the requirements on the matching stage of the algorithm. As a consequence, we can simplify this stage and trade accuracy for speed. Second, the algorithm is well-equipped to handle dynamic scenes, since each independent motion gives rise to a separate set of self-consistent features. While we will not discuss dynamic scenes in this paper, it should be noted that this algorithm is not confused when the robot is driving into its own shadow, a known problem for many algorithms.

There are other properties of the algorithm that should also be noted. First, it does not require an initial motion estimate, and, partly as a result, can handle very large image translations. Second, the feature descriptor is not invariant to rotation or scaling, so large motions around or along the optical axis will result in poor performance. This apparent limitation, however, is largely a design choice: for ground vehicle applications, rotations around the optical axis are generally small (since this corresponds to vehicle roll), while scale changes only become significant at high-speeds and/or low frame-rates. Furthermore, the non-invariant detectors used in our algorithm are at least an order of magnitude faster than their invariant cousins.

The visual odometry algorithm described in this paper has been employed in a number of programs, including DARPA LAGR (Learning Applied to Ground Robotics) and Biodynotics BigDog (BigDog is a quadruped dynamic walker built by Boston Dynamics) [3]. In the following sections, we describe the algorithm and implementation in detail, and consider its application to the BigDog and LAGR vehicles. We present experimental results from image sequences totaling over 24000 frames and 1700m of travel.

A. Howard is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109; andrew.howard@jpl.nasa.gov.

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work was supported by the DARPA Learning Applied to Ground Robotics Program (LAGR), the DARPA Biodynotics Program (BigDog) and the NASA Mars Technology Program (MTP).

II. Related Work

Contemporary approaches to visual odometry can be roughly categorized as being either monocular or stereo, and employing either feature tracking or feature matching. Nister [12], for example, describes a monocular approach that emphasizes feature tracking over small image motions, and demonstrates the effectiveness of this algorithm for real-time motion estimation on a variety of ground vehicles. Matthies [11] describes a stereo approach that also makes use of feature tracking. This approach was later evolved into the visual odometry algorithms used on the JPL Mars Exploration Rovers [10], and on the soon-to-be-launched Mars Science Lab [8]. The approach described in this paper, however, does not derive from the Matthies/MER/MSL lineage. Instead, it follows from the approach described by Hirschmüller [7], which uses feature matching rather than tracking and employs stereo range data for inlier detection. Our approach both generalizes and simplifies this earlier work, introducing a more complete inlier detection scheme (based on the notion of cliques) and simplifying the point-to-point inlier test to permit faster comparisons.

Finally, we note that Agrawal [1] has recently described an approach based on stereo feature-matching and used in the DARPA LAGR program. The results for this algorithm are comparable with those reported here (see Section V).

In the taxonomy above, we explicitly exclude "batch" techniques, such as structure-from-motion, bundle-adjustment and visual SLAM. That is, our interest lies in incremental, real-time, low-latency methods for estimating camera motion. As an aside, however, we note there has been some recent progress on sliding window techniques for stereo and visual odometry [14], and that these approaches could be applied to the algorithm presented here.

III. Algorithm Description

For our stereo visual odometry algorithm, we assume that the robot is equipped with a stereo camera pair, and that the images have been processed using a dense stereo algorithm (such as [11], [5]). Real-time stereo algorithms typically proceed through the following steps:

1) Rectification: warp the original left/right images such that epipolar lines are aligned with the image rows. These rectified images correspond to the images one would obtain from a virtual pair of perfectly aligned perspective cameras.
2) Pre-filtering: smooth the rectified images with an edge-preserving filter to remove high-frequency components. While this is ideally performed using a Laplacian-of-Gaussian or a bilateral filter [15], a difference-of-boxes filter is often preferred for the sake of speed.
3) Correlation: construct a disparity image by locating matching pixels in the left/right pre-filtered images. Typical scoring methods for the matching step include sum-of-absolute differences (SAD) and CENSUS [16], both of which consider a small window around the pixel of interest. The disparity at each pixel is proportional to the inverse range.

The inputs available to the visual odometry algorithm, therefore, are the raw, rectified, pre-filtered and disparity images. The algorithm is as follows.

A1. Inputs

Let J_a, D_a denote the pre-filtered and disparity images for frame a, acquired at some time t_a. Let J_b, D_b denote the pre-filtered and disparity images for frame b, acquired at some later time $t_b > t_a$.

A2. Feature detection: $(J_a, D_a; J_b, D_b) \rightarrow (F_a; F_b)$

Detect features in the pre-filtered image J_a using a standard corner detector (such as Harris [6] or FAST [13]) and compute the corresponding world coordinates from the disparity image D_a. Discard features with unknown disparity. For each feature, construct an $m \times m - 1$ descriptor from the surrounding pixels in J_a. We use odd values for m and omit the middle pixel, such that the feature descriptor length is a multiple of 8; e.g., a 7x7 window yields a descriptor of length 48. Let $f_a = (j, w, s)$ denote a feature with image location j, world location w and descriptor s; let F_a denote the set of all such features.

Repeat for images J_b and D_b to find the feature set F_b.

In practice, the first detection step (on frame a) can be omitted: we simply retain the feature set F_b from the previous frame, such that it can become F_a for the current frame. With this optimization, the inputs to the algorithm become the feature set $F_b \rightarrow F_a$ from the previous frame and the new image pair (J_b, D_b).

A2. Construct the score matrix: $(F_a, F_b) \rightarrow S$

Compute the score matrix S for all pair-wise combinations of features in F_a and F_b using the sum-of-absolute differences (SAD) between the feature descriptors. Low values in the scoring matrix indicate that two features are very similar; high values indicate that they are very different. On machines with vector instruction sets (such as Intel SSE2), the SAD operation can be performed on eight bytes simultaneously.

A3. Match features: $(F_a, F_b, S) \rightarrow M$

Lucas Kanade

Using the score matrix S, match features F_a in frame a with features F_b in frame b. This is equivalent to the linear assignment problem, which is typically solved using either the greedy algorithm (for sub-optimal solutions) or the Hungarian method (for optimal solutions). However, since both of these algorithms are relatively expensive ($O(n^2 \log n)$ and $O(n^3)$ in the number of features, respectively), we use a faster $O(n^2)$ assignment algorithm that selects co-occurring minima in the score matrix. This algorithm can be stated as follows: for each feature $f_a \in F_a$, find the feature f_b with the minimum SAD score; for each feature $f_b \in F_b$, find the feature $\bar{f}_a$ with the minimum SAD score. If $\bar{f}_a = f_a$ and $\bar{f}_b = f_b$, the features f_a and f_b are declared a match.

Let (f_a, f_b) denote a match between features in frames a and b, and let M be the set of all matches.

A4. Find the maximum inlier set: $M \rightarrow Q$

Compute a consistency matrix W for all pairwise combinations of matches in M, using a simple rigidity constraint on the world coordinates. A pair of matches is consistent if the distance between two features in frame a (measured in world coordinates) is identical to the distance between the corresponding features in frame b. Any pair of matches for which this is not the case must contain at least one incorrect match, or must contain features from some independent mover. At the simplest level, a pair of matches (f_a, f_b) and (f'_a, f'_b) are consistent iff they satisfy the inequality:

$$|w_a - w'_a| - |w_b - w'_b| < \delta \quad (1)$$

where w denotes the world coordinates of a feature and δ is a fixed threshold. The corresponding entry in the consistency matrix W contains a 1 if the constraint is satisfied, and 0 otherwise. This test can also be generalized to better handle the quadratic range errors arising from stereo triangulation (see [7], for example), but the analysis is beyond the scope of this paper.

Using the matrix W, find the largest set of mutually consistent matches. This is equivalent to finding the maximum clique on a graph with adjacency matrix W. Since the maximum clique problem is known to be NP-complete, we use the following sub-optimal algorithm:

1) Initialize the clique to contain the match with the largest number of consistent matches (i.e., choose the node with the maximum degree).
2) Find the set of matches compatible with all the matches already in the clique.
3) Add the match with the largest number consistent matches.

Steps 2 and 3 are repeated until the set of compatible matches is empty.

Let Q denote the set of matches (f_a, f_b) in the inlier set discovered by this algorithm.

A5. Estimate motion: $Q \rightarrow \Delta_{ab}, \epsilon$

Estimate frame-to-frame camera motion by minimizing the image re-projection error for all matches in the clique Q. We seek the homogeneous transform Δ_{ab} that minimizes the re-projection error:

$$\epsilon = \sum_{(f_a, f_b) \in Q} (j_a - P\Delta w_b)^2 + (j_b - P\Delta^{-1} w_a)^2 \quad (2)$$

where j and w are the homogeneous image and world coordinates, respectively, and P is the camera projection matrix. The solution is found using the standard Levenberg-Marquardt least-squares algorithm, with one small modification: after finding the initial solution Δ_{ab}, we discard any matches whose re-projection error exceeds a set threshold and re-run the optimization. This second pass discards any outliers that survive the inlier detection step.

A6. Validate the solution: $(Q, \epsilon) \rightarrow \{pass|fail\}$

Validate the solution against three criteria:

- The number of points in the clique Q. At a minimum, we require three points to generate a unique motion estimate. In practice, however, we typically demand at least ten points to ensure that the clique captures the camera egomotion (as opposed to some other motion in the scene).
- The co-linearity (or otherwise) of features in the image. This is done be computing the eigenvalues of the feature distribution and computing the maximal ratio; values close to 1 indicate a good spread of features.
- The re-projection error ϵ, which must be below some threshold (e.g., 0.3 pixels).

A7. Outputs: $\Delta_{ab}, \{pass|fail\}$

The algorithm outputs the frame-to-frame motion estimate Δ_{ab} and a status flag indicating if the solution is valid. A covariance matrix is also computed, but not currently used.

Discussion

There are three features of this algorithm that should be noted. First, the algorithm gains much of its speed by exploiting stereo pre-processing (rectification, pre-filtering and correlation). The computation time for visual odometry is typically a small fraction of the time required for stereo processing. Second, the algorithm uses *inlier detection* rather than *outlier rejection* for selecting good feature matches. The clique-based approach described in Step A4 can easily cope with frames containing 90% outliers, which would severely tax RANSAC-style outlier-rejection schemes. Third, the algorithm includes strong validation checks to detect egregious failures. For autonomous vehicle applications, failure is generally acceptable if it is reported as such; unreported failures, on the other hand, can very quickly lead to catastrophic divergence of the integrated pose estimate.

The key tuning parameters in this algorithm are the feature descriptor size (A2), clique inlier threshold (A4), miminum number of features (A6) and re-projection error (A6). None of these parameters shows a high degree of sensitivity, however, and one can easily find a set of values that produces both a high frequency of correctly matched frames and a very low (or zero) frequency of unreported failures. It should be noted that the same set of parameter values is used for all of the experiments described in Sections IV and V. Algorithm run-time is mainly driven by the feature descritor size; experimentally, we have found a window size of 7 to 9 to be a good compromise between speed and reliability.

The algorithm also has some key limitations following directly from the design choices. First, the feature detector and descriptors are not scale invariant, such that large motions along the optical axis will produce relatively few matches. While we could use a scale invariant feature detector such as SIFT [9] or SURF [2], these would significantly reduce our real-time performance; the current corner detector can process a 640x480 image in a few milliseconds, versus tens or hundreds of milliseconds for scale invariant detectors.

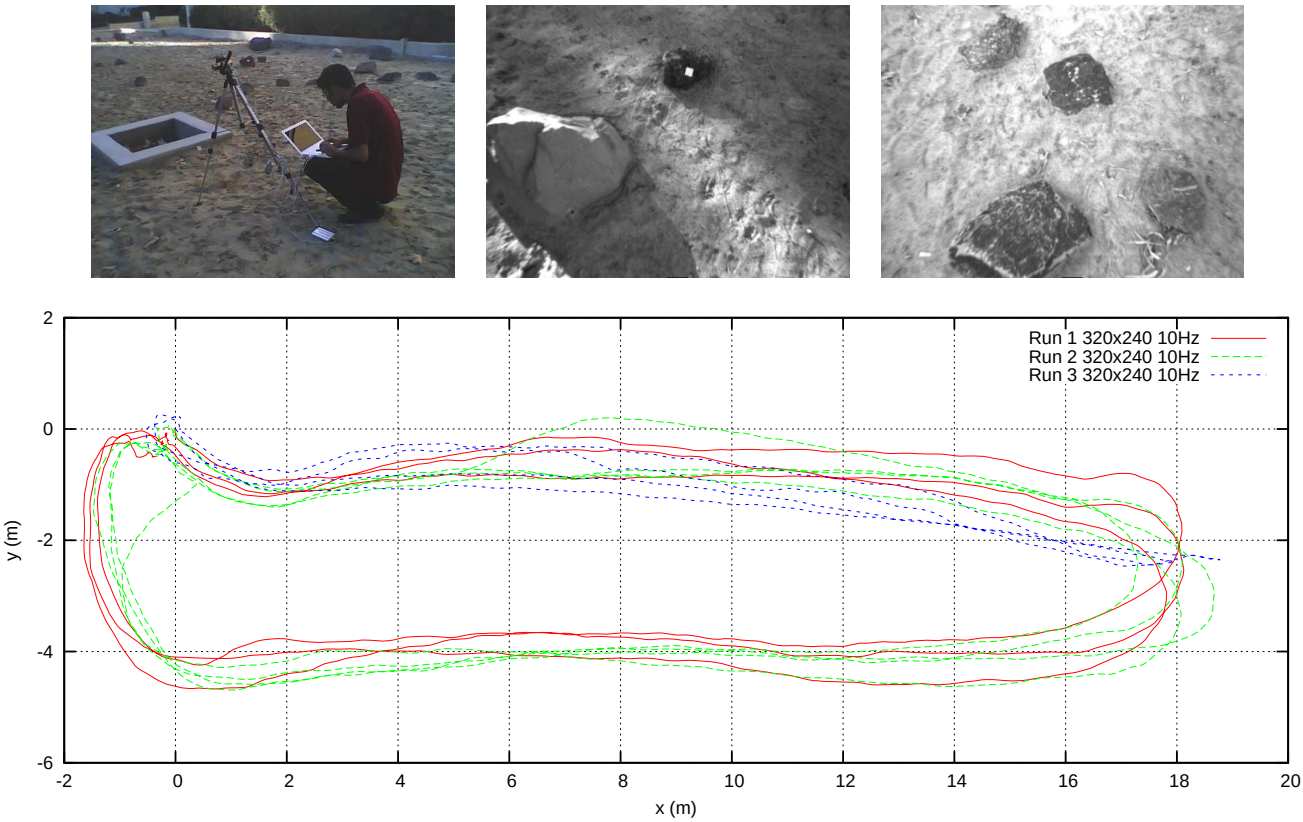

Fig. 1. Top-left: the JPL sandbox, with the BigDog camera mounted on a tripod (the tripod is hand-carried). Top-middle and top-right: typical camera views, showing sand, rocks and leaves. The bright square is a retro-reflector used for ground-truth. Bottom: trajectories estimated by visual odometry (three runs). The first two runs start at $(0, 0)$ and take clockwise laps around the sandbox; the final run travels straight out and back (twice).

			30Hz				15Hz			
Run	Size	Dist.	Frames	Time	2D RMS err.	3D RMS err.	Frames	Time	2D RMS err.	3D RMS err.
1	160x120	156m	4815	14.3ms	0.422m (0.27%)	0.824m (0.53%)	2407	14.4ms	0.597m (0.39%)	0.669m (0.44%)
2	160x120	202m	5875	17.8ms	0.723m (0.36%)	0.934m (0.46%)	2937	17.6ms	0.178m (0.09%)	0.382m (0.19%)
3	160x120	89m	2843	19.7ms	0.435m (0.49%)	0.530m (0.60%)	1421	19.2ms	0.092m (0.11%)	0.095m (0.11%)
1	320x240	154m	4815	16.1ms	0.439m (0.28%)	0.554m (0.36%)	2407	16.1ms	0.196m (0.13%)	0.310m (0.20%)
2	320x240	201m	5875	19.3ms	0.200m (0.10%)	0.461m (0.23%)	2937	19.0ms	0.167m (0.08%)	0.436m (0.22%)
3	320x240	88m	2843	22.7ms	0.073m (0.08%)	0.175m (0.20%)	1421	24.1ms	0.029m (0.03%)	0.030m (0.03%)
1	640x480	154m	4815	22.0ms	0.196m (0.13%)	0.235m (0.15%)	2407	20.9ms	0.158m (0.10%)	0.208m (0.14%)
2	640x480	200m	5875	23.3ms	0.091m (0.05%)	0.362m (0.18%)	2937	23.2ms	0.106m (0.05%)	0.274m (0.14%)
3	640x480	88m	2843	27.9ms	0.055m (0.06%)	0.102m (0.12%)	1421	27.7ms	0.046m (0.05%)	0.047m (0.05%)

TABLE I

BIGDOG SANDBOX RESULTS (HAND-HELD CAMERA).

Second, the motion estimator only compares two frames, and the overall camera trajectory is computed via simple integration of successive Δ's. As we will show in Section IV, this can lead to a paradoxical decrease in accuracy at high frame rates. The alternative is to employ bundle-adjustment (BA) or simultaneous localization and mapping (SLAM), but this is not without cost. Most of the relevant BA/SLAM techniques employ a sliding window of image frames, such that pose estimates are only reported after the frame has departed the sliding window (high latency), or are continuously revised as new measurements arrive (unusual semantics). In contrast, the stereo visual odometry algorithm described here is a real-time, low-latency 'black-box' whose output is directly analogous to wheel odometry or inertial navigation systems.

IV. APPLICATION 1: BIGDOG

BigDog is a four-legged, dynamically stabilized robot developed by Boston Dynamics. It is a prototype 'pack-mule' intended for use in rugged off-road terrain, and is capable of walking, trotting, climbing slopes and stepping over obstacles. BigDog uses visual odometry for both pose estimation and terrain reconstruction. For pose estimation, visual odometry can be combined with inertial data from the on-board IMU, and with kinematic pose estimates derived from joint encoders in the legs (i.e., leg odometry). For terrain reconstruction, stereo range data and visual odometry pose estimates are fused directly to produce a high-resolution sliding map of the terrain directly in front of and beneath the robot. This allows the robot to plan foot placements

around obstacles with centimeter-level accuracy, even if those obstacles are no longer visible in the camera.

The primary vision sensor on BigDog is a downward-looking stereo camera pair from Point Grey Research (Bumblebee 640x480 monochrome with 40 degree horizontal field-of-view). To evaluate the performance of visual odometry, we collected data from an identical hand-held camera in the JPL sandbox (see Figure 1). The sandbox is a somewhat simplified off-road environment, with sand, rocks and a scattering of fallen leaves. The camera was hand-carried to simulate the motion of the BigDog vehicle, with two human legs substituting for the four legs of BigDog (both produce periodic vertical displacement, albeit at different frequencies). Three trials were conducted, as shown in Figure 1: runs 1 and 2 are clockwise loops around the sandbox (three and four laps, respectively) and run 3 is straight out-and-back (two laps).

Obtaining ground-truth pose estimates for a hand-held camera is somewhat challenging, and we have therefore opted to ground-truth the reconstructed terrain rather than the camera trajectory. Using a TotalStation, we surveyed four key points in the sandbox (i.e., distinctive rocks), then manually identified these rocks in the recorded image sequence. This process establishes pair-wise correspondences between the survey points and the estimated location of those points according to visual odometry and stereo ranging. The two sets of points are related by an unknown 3D rigid-body transformation, which we estimate via least-squares optimization (i.e., minimizing the sum-of-squared point-to-point distances). The results described below represent the residual errors after solving for this transform.

Table I summarizes the results for this data set. The raw data was logged at 640x480 resolution at 30Hz, but we have generated results for multiple camera resolutions (640x480, 320x240 and 160x120) and frame rates (30Hz and 15Hz, the latter simulated by dropping every other frame). The accuracy of visual odometry is indicated by the reconstruction error, defined as the root-mean-squared distance between the estimated and surveyed key points. We distinguish between the 2D error, which includes horizontal offsets only, and the 3D error, which includes horizontal and vertical offsets. Note that there are no visual odometry failures in these results; for all combinations of resolution and frame rate, the algorithm was able to match a sufficient set of features across frames.

The first notable feature of these results in Table I is also the least surprising: in all cases, accuracy improves with increasing image resolution. For example, on run 2, which covers 202m over 5875 frames, the 2D RMS error drops from 44cm at 160x120 to 5cm at 640x480 (processing data at 30Hz). In this environment, where there are good features to track on all scales, increased resolution improves feature localization, which in turn improves the frame-to-frame and integrated motion estimates.

The second notable result is that the accuracy improves as the frame rate *decreases*. This is initially counter-intuitive, until one considers that our visual odometry algorithm is estimating and integrating sequential frame-to-frame motions. Therefore, at higher frame rates, we are also integrating more noise. These results suggest that better accuracy may be obtained with multi-frame or sliding window estimators (e.g., bundle adjustment, visual SLAM and related techniques, as described in Section III). There is, of course, a lower bound on the practical frame rate, since any visual odometry algorithm must have significant image overlap in order to match features. In the case of BigDog, simple calculations based on the expected vehicle rotation rates put that lower bound at around 10 to 15Hz.

Table I lists the processing time for the visual odometry algorithm running on a single core of a 2.4GHz Core 2 Duo processor. These numbers do not include the stereo processing time, which ranges from 6ms at 160x120 to 25ms at 320x240 and 180ms at 640x480. The processing time for visual odometry ranges 14ms at the lowest resolution to 28ms at the highest, and does scale directly with image size. If we consider the steps in the algorithm, we expect the feature detection stage to scale with the number of image pixels, but for all subsequent stages to scale with the number of features. The preferred number of features is one of the parameters to the algorithm, and is roughly constant across resolutions.

At the desired BigDog frame rate of 10-15Hz, we can run the full stereo plus visual odometry system in real time at a resolution 320x240 (approximately 50ms/frame). This yields an expected 3D reconstruction error of around 40cm over 200m (0.2% of distance traveled).

V. Application 2: LAGR

Visual odometry was an integral component of the JPL effort for the DARPA LAGR (Learning Applied to Ground Robotics) program, and was used for both pose estimation and slip detection. The slip detector compares the vehicle motion as determined by wheel encoders to the motion determined by visual odometry, and measures the degree to which the wheels are spinning in place. This slip signal is an important input into both the planning and learning algorithms: the planner uses slip as a virtual bumper (i.e., if the robot is 'digging in', back up and try a different route), while the learner uses slip as a training signal (so that the robot can avoid similar-looking terrain in the future).

The LAGR vehicle is shown in Figure 2; it is equipped with two stereo camera pairs, wheel encoders and a MEMs IMU. We employ a very simple filter that fuses visual odometry, wheel encoder and IMU data. Most of the time, the filter simply integrates motion estimates from visual odometry, but falls back on encoders and IMU data when the visual odometry reports failure.

Data was collected in partially vegetated terrain in the Arroyo Seco near JPL (see Figure 2). The robot was manually driven around loop course, with four separate runs, each consisting of multiple laps; about 1200m total distance traveled. Ground-truth was collected with a TotalStation, by periodically stopping the robot and measuring the position of a retro-reflector; approximately 6 survey points were collected on each lap. As was the case with the BigDog data

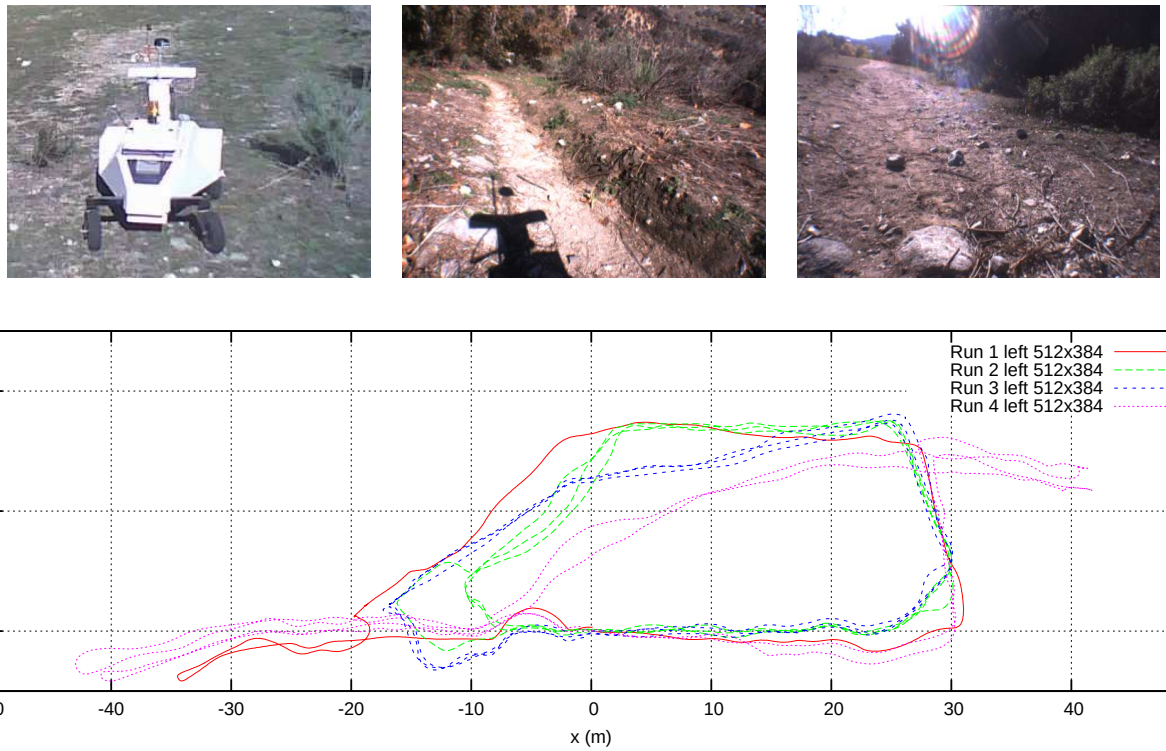

Fig. 2. Top-left: LAGR robot in the Arroyo Seco. Top-right: Sample terrain types, including grass, bushes, rutted and packed earth. Bottom: trajectories estimated by visual odometry (four runs). Each run starts at $(0, 0)$ and takes one or more counter-clockwise laps around the course.

							VisOdom		Encoders	
Run	Src.	Size	Frames	Dist.	Fail	Time	2D RMS err.	3D RMS err.	2D RMS err.	3D RMS err.
1	left	256x192	2334	166m	10	4.8ms	0.462m (0.28%)	0.484m (0.29%)	2.575m (1.55%)	3.356m (2.02%)
2	left	256x192	3936	335m	5	7.8ms	0.960m (0.29%)	1.501m (0.45%)	7.196m (2.15%)	8.415m (2.51%)
3	left	256x192	3467	360m	13	7.1ms	1.323m (0.37%)	1.742m (0.48%)	12.053m (3.35%)	13.689m (3.81%)
4	left	256x192	4114	406m	39	7.4ms	2.015m (0.50%)	3.038m (0.75%)	16.349m (4.03%)	17.906m (4.41%)
1	right	256x192	2548	166m	1	7.2ms	1.358m (0.82%)	1.654m (0.99%)	2.575m (1.55%)	3.356m (2.02%)
2	right	256x192	4067	335m	0	7.0ms	1.436m (0.43%)	1.559m (0.47%)	7.196m (2.15%)	8.415m (2.51%)
3	right	256x192	3681	360m	5	8.6ms	0.738m (0.21%)	1.088m (0.30%)	12.053m (3.35%)	13.689m (3.81%)
4	right	256x192	4235	406m	11	8.5ms	1.154m (0.28%)	1.642m (0.40%)	16.349m (4.03%)	17.906m (4.41%)
1	left	512x384	2334	166m	0	17.7ms	0.145m (0.09%)	0.434m (0.26%)	2.575m (1.55%)	3.356m (2.02%)
2	left	512x384	3936	335m	3	20.9ms	0.317m (0.09%)	0.758m (0.23%)	7.196m (2.15%)	8.415m (2.51%)
3	left	512x384	3467	360m	5	16.1ms	0.630m (0.18%)	1.013m (0.28%)	12.053m (3.35%)	13.689m (3.81%)
4	left	512x384	4114	406m	13	22.8ms	0.965m (0.24%)	1.364m (0.34%)	16.349m (4.03%)	17.906m (4.41%)
1	right	512x384	2548	166m	1	19.0ms	0.249m (0.15%)	1.485m (0.89%)	2.575m (1.55%)	3.356m (2.02%)
2	right	512x384	4067	335m	1	14.6ms	0.437m (0.13%)	0.736m (0.22%)	7.196m (2.15%)	8.415m (2.51%)
3	right	512x384	3681	360m	0	17.2ms	0.531m (0.15%)	0.601m (0.17%)	12.053m (3.35%)	13.689m (3.81%)
4	right	512x384	4235	406m	8	13.1ms	0.534m (0.13%)	1.033m (0.25%)	16.349m (4.03%)	17.906m (4.41%)

TABLE II
LAGR ARROYO RESULTS

Fig. 3. Two sets of failed frames from from LAGR Arroyo data set. Note the large image motions (left pair) and obscuring vegetation (right pair).

sets, the ground truth trajectory is related to the estimated trajectory by an arbitrary 3D rigid-body transform. Therefore, the results quoted below are the residual errors after solving for this transform.

Results are summarized in Table II. The vehicle has two independent, unsynchronized stereo camera pairs, so we obtain two sets of data (left and right) for each run around the course. Looking at the full resolution results, we see that the 2D error is less than a meter over 400m of travel (between 0.15% and 0.25%). Compare this with the results for wheel odometry plus IMU, where the error over the same distance is 16m (4% of distance traveled). This is despite the fact that this terrain is relatively favorable for wheel odometry, with uneven terrain but no slipping or sinking.

This data set was collected on a sunny day, and the robot is often driving into its own shadow. Self-shadows have traditionally been a problem for visual odometry algorithms (particularly on MER); the algorithm tends to lock-onto strong features at the shadow boundary, and incorrectly report that the vehicle is stationary. However, with clique-based inlier detection, feature points on the shadow boundary are disregarded (not part of the maximum clique) and self-shadows are therefore ignored.

Unlike the BigDog sandbox results reported in the previous section, visual odometry reports a number of failures on this course, some of which are shown in Figure 3. There are two principle failure modes. First, there are some long gaps in the logs (each lasting a second or more) when the logging disks where unable to sustain the frame rate. Not all of these gaps are successfully bridged by the algorithm, particularly when the image overlap is small or the scale change is large. While this is more of a system failure than a visual odometry failure, it is worth noting that the latter reports a failure rather than returning an incorrect motion estimate. The second failure mode occurs when the robot is close to vegetation at camera-height. The dense stereo methods we employ fail on nearby vegetation, leaving the visual odometry with few pixels to work with. Once again, this is a failure of stereo rather than of visual odometry per se, but it does highlight a limitation of our approach: visual odometry will only work in environments where stereo works. This includes rocky, sandy environments (such as Mars), but excludes certain vegetated environments on Earth (e.g., pushing through tall grass).

VI. Discussion and Conclusion

In this paper, we have presented results obtained through "pure" (or nearly pure) visual odometry, demonstrating a high degree of reliability and an accuracy of better than 0.25% over 400m of travel.

While it may be possible to improve on this result using multi-frame estimators (e.g., sliding window bundle adjustment), real-world accuracy is limited by other factors. These include missed frames and poor camera calibration. When visual odometry fails, we must rely on other, less accurate, sensors to bridge the gap. In our LAGR experiments, for example, we have found that a single missed frame, if poorly timed, can introduce more error than a thousand correctly matched frames. It is also possible that poor camera calibration may introduce systematic biases into the motion estimate. One response to this, which we have not employed in this paper, is to tune the camera calibration using visual odometry; i.e., tweak the calibration until the estimated trajectory fits the ground-truth trajectory. This approach must be used with caution, however, since other factors (such as terrain type and feature distribution) may also contribute to long term biases.

Having presented the capabilities and limitations of pure visual odometry, it should be noted that even the most minimal robot is likely have some form of proprioceptive sensing (including encoders and/or IMU). The stereo visual odometry algorithm described in this paper is therefore intended to augment rather than replace these sensors, and to work with higher-level pose estimators that fuse data from multiple sources.

References

[1] M. Agrawal and K. Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive GPS. In *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 1063–1068, 2006.

[2] H. Bay, T. Tuytelaars, and L.V. Gool. SURF: speeded-up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.

[3] M. Buehler, R. Playter, and M. Raibert. Robots step outside. In *International Symposium of Adaptive Motion of Animals and Machines (AMAM)*, Ilmenau, Germany, Sept 2005.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[5] S. B. Goldberg, M. W. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *Proceedings of the 2002 IEEE Aerospace Conference*, volume 5, pages 2024–2036, 2002.

[6] C. J. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conferences*, pages 147– 151, 1988.

[7] H. Hirschmüller, P.R. Innocent, and J.M. Garibaldi. Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics. In *Control, Automation, Robotics and Vision (ICARCV'02)*, pages 1099–1104, 2002.

[8] A. E. Johnson, S. B. Goldbert, Y. Cheng, and L. H. Matthies. Robust and efficient stereo feature tracking for visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, U.S.A., 2008.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[10] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the Mars Exploration Rovers. *Journal of Field Robotics, Special Issue on Space Robotics*, 24(3):169–186, March 2007.

[11] L. Matthies. *Dynamic Stereo Vision*. PhD thesis, Dept. of Computer Science, Carnegia Mellow University, 1989. CMU-CS-89-195.

[12] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, Jan 2006.

[13] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, 2006.

[14] G. Sibley, L. Matthies, and G. Sukhatme. Bias reduction and filter convergence for long range stereo. In *Robotics Research*, volume 28 of *Springer Tracts in Advanced Robotics*, pages 285–294. 2007.

[15] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision (ICCV)*, pages 389–394, 98.

[16] R. Zabih and J. Woodfill. Non-parametric local transforms for computer visual correspondence. In *European Conference on Computer Vision (ECCV)*, pages 151–158, 1994.