Efficient Off-Road Localization Using Visually Corrected Odometry

Matthew Grimes and Yann LeCun

Abstract— **We describe an efficient, low-cost, low-overhead system for robot localization in complex visual environments. Our system augments wheel odometry with visual orientation tracking to yield localization accuracy comparable with "pure" visual odometry at a fraction of the cost. Such a system is well-suited to consumer-level robots, small form-factor robots, extraterrestrial rovers, and other platforms with limited computational resources. Our system also benefits high-end multi-processor robots by leaving ample processor time on all camera-computer pairs to perform other critical visual tasks, such as obstacle detection. Experimental results are shown for outdoor, off-road loops on the order of 200 meters. Comparisons are made with corresponding results from a state-of-the-art pure visual odometer.**

I. INTRODUCTION

The ability for a robot to localize itself can be critical for successful autonomous operation. While a globally consistent solution to the localization problem must necessarily also perform mapping [1], many applications do not require or benefit from a globally consistent map. Locally consistent approaches such as fixed-time-window SLAM [2] and visual odometry [3], [4] have shown great success in applications such as goal-directed navigation and localization in dynamic environments.

Wheel odometry has been a popular mainstay for robotic localization due to its low overhead and high sampling frequency. Its accuracy however is limited by wheel slip, a source of error that can be challenging to detect and correct without other sensors. Wheel slip can be particularly frequent and destructive in runs over outdoor terrain with loose or uneven ground. Full visual odometry (VO) uses feature tracking to entirely replace ground odometry, but current systems [4] require 100% of the processing time on a high-end CPU. For single-CPU autonomous robots, this cost can be prohibitive. The cost of VO can be a burden even for multi-CPU platforms, as it is often desirable for all camera-computer pairs to be able to perform additional tasks, such as short-range obstacle detection, at high framerates in their respective fields of view.

We have implemented a visual localization system that runs at a fraction of the cost of state-of-the-art VO systems while maintaining comparable accuracy. On our multi-processor, multi-camera system, this allows a single processor-camera pair to handle VO in parallel with other visual tasks, enabling tight coupling between localization, obstacle detection, and control. Our system can be of even more use to robots with limited computational power, such as small robots, consumer-oriented platforms, or extraterrestrial rovers.

We achieve this performance gain by specializing the visual odometer to the task of tracking only one degree of freedom: the robot's bearing. This bearing estimate is combined with a wheel odometer's translation estimate to yield 3-DOF pose estimates with much-improved accuracy over wheel odometry, and comparable accuracy to 6-DOF visual odometers on low-curvature terrain. The efficiency of our system comes from the fact that tracking only the bearing allows us to operate at much lower resolutions than would be acceptable on a 6-DOF odometer. This is because the uncertainty of an object's distance grows rapidly with distance at low resolutions, while the uncertainty of its robot-relative bearing is constant. For example, at our resolution of 160x120 pixels, an uncertainty of ± 0.25 pixels translates to ± 0.1 degrees of yaw, but ± 1.25 meters of distance for an object 6 meters away. Additional speedups are gained by using spherical image projection for more reliable feature tracking, and limiting the feature tracking to windows bounded by wheel odometer output. We show that this hybrid odometry approach achieves much of the benefit of a full visual odometer at a greatly reduced computational cost.

II. RELATED WORK

There has been much work in both wheel odometry and visual odometry (VO), while only limited attention has been paid to the intersection of the two approaches. Schaefer et. al. [5] use wheel odometry to check the output of a full VO system for errors arising from moving objects. Rather than run full visual and wheel odometry systems in parallel, we have focused on how to best exploit wheel odometry to lighten the computational burden of VO. A number of authors have used visual matching to correct IMU or GPS data on aerial platforms [6] [7] [8]. Our system is implemented on a ground rover, where many of the assumptions afforded in the air, such as nearly coplanar features and slow visual flow, do not apply.

Most other work in relative pose tracking has focused either on using non-visual sensors to detect wheel slip, or on employing full visual odometry as a complete replacement to wheel odometry.

A. Wheel odometry correction

Wheel slip can occur in many flavors, from sudden spurts of wheel speed to gently increasing drift. The latter in

This work was supported by the DARPA Learning Applied to Ground Robotics (LAGR) project

M. Grimes is with the Courant Institute of Mathematical Sciences, New York University, 719 Broadway 12th floor, New York, NY 10003, USA mkg@cs.nyu.edu

Y. LeCun is with the Faculty of Courant Institute of Mathematical Sciences, New York University, 719 Broadway 12th floor, New York, NY 10003, USA yann@cs.nyu.edu

particular is difficult to detect by simple cross-checking against motor current or inertial sensor output, requiring more nuanced approaches. Ojeda et. al. [9] correct odometry using parameterized functions of motor current and soil cohesion. However, Maimone et. al. [10] report that such an approach fares poorly unless the soil consistency is nearly homogeneous. Ward and Iagnemma train an SVM on hand-labeled odometric and inertial sensor outputs to detect immobilization [11]. In another paper [12], the same authors employ a model-based approach, inferring robot velocity by fusing the output of a physical model with IMU, GPS, and wheel odometry output. A simpler approach to detecting slip would be to complement odometry with the absolute position measurements provided by GPS. Unfortunately, GPS input can be sporadic and inaccurate in wooded or urban environments [13] [14], and even under optimal conditions refreshes only once per second [15]. On our ground rover platform, we have found that one second is plenty of time for a robot to hallucinate a sharp turn due to wheel slip and react by sharply turning in the opposite direction. When driving alongside entangling vegetation or on narrow forest paths, such sudden "corrective" turns can prove catastrophic to a run. Furthermore, both GPS and odometry exhibit highly non-Gaussian error profiles, as neither GPS "jumps" nor wheel slip errors are well modeled as a random walk around a mean value. Sukkarieh et. al. [13] therefore use a chi-squared gating function to detect and discard blatantly spurious sensor outputs, and feed only vetted sensor readings to a Gaussian model, in their case an EKF sensor fusion algorithm. However, this does not address the problem of GPS's low update frequency, nor a gating function's insensitivity to gradual odometry drift at low speeds. Rather than attempting to selectively detect and correct bad rotation estimates by the wheel odometer, we have opted to replace them entirely with the more reliable rotation estimates of our visual rotation tracker.

B. Full visual odometry

Full visual odometry tracks visual features to make a differential estimate of the robot's full 6-DOF pose. Nistér et. al. [3] tracked Harris corners [16] in real time, discarding spurious feature associations between frames using RANSAC [17]. Konolige et. al. improved upon this in their own real-time system by incorporating bundle adjustment to reduce drift [4]. However, their system occupies all of the processor time on a high-end CPU, requiring additional computers to handle other aspects of autonomous operation such as mapping and planning. The NASA Mars Exploration Vehicle is hit particularly hard by the computational demands of full VO, which can take up to 3 minutes per frame on its 20 MHz processor, leading to an average movement of 10m/hour [10]. By contrast, our system is implemented on the same robotic platform used in [4] where one of its two camera-computer pairs is dedicated to the task of real-time full VO. However, this leaves that camera-computer pair unable to perform other potentially critical tasks on its field of view, such as obstacle detection. For this reason, we have chosen our more lightweight approach.

(a) Camera image (b) Spherical image

Fig. 1. A rectified camera image and its spherical transform. The asymmetry of the spherical image is due to the fact that the camera is pointed off to the left and down, with some axial roll. The robot is pointed straight at the area highlighted by four yellow dots, placed at pitch and yaw values $[0.03, 0.1]$, $[0.03, -0.1]$, $[-0.03, -0.1]$, and $[-0.03, 0.1]$, in radians. After the transformation, these points form an axis-aligned rectangle around the frontal direction. In practice, we transform only the portion of the camera image roughly located around the horizon, highlighted by the green rectangle above.

III. ALGORITHM

In the interest of keeping running costs down, our system tracks only six patches per frame, tracking wide image patches at low resolution. The system samples patches from a region around the horizon, interpreting their horizontal motion from frame to frame as a rotation of the robot. There are two challenges to this approach: one is that a small number of features may be less robust to mismatches. The other is that a robot driving in a straight line will see features drift towards the sides of the image as it drives by ("parallax drift"). Under a naïve implementation, such horizontal motion in the image would be incorrectly interpreted as a rotation. In this section we give a walkthrough of our algorithm, paying particular attention to the solutions to the above problems. The overall algorithm is as follows:

1) Re-map the image to remove feature size distortions due to planar projection.
2) Sample features from a small region of the previous frame selected using wheel odometry.
3) Search for the sampled features in the current frame, again limiting the search area using wheel odometry.
4) Cross-validate the features' motions between frames and discard any outliers.
5) Localize the robot in the current frame by replacing the wheel odometry's rotation estimate with that of the visual odometer, and rotating the translation estimate by the difference in rotations.

A. Re-map image to a spherical projection

Because we track a small number of image patches per frame, care must be taken to minimize the number of mismatched patches. We use wide patches subtending eight degrees of yaw, as larger patches tend to be more distinctive. However, such large features stretch when moved from the center of the image to the edges, where each pixel subtends a smaller solid angle. This distortion can cause mismatches when searching for features that have moved from the center of the image to near an edge, or vice-versa. To remove this

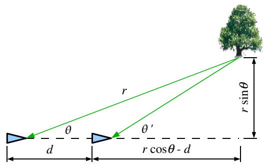


Fig. 2. **Parallax from pure forward motion.** We wish to limit our attention to objects which will not shift under parallax from frame to frame. In a spherical image where each pixel subtends θ_p radians of yaw, this requires that the bearing θ of the object not change more than $\theta_p/2$. Given upper limits on θ and the frame-to-frame translation d, we may solve for a minimum distance r. We avoid parallax by detecting distance using stereo data calculated earlier for obstacle detection, and ignoring all features closer than r.

distortion, we remap the camera image using a "spherical projection", where each pixel row and column covers a fixed amount of vehicle-relative pitch and yaw $[\phi_p, \theta_p]$, respectively (figure 1). We define the mapping from camera image coordinates $[i, j]$ to spherical image coordinates $[k, l]$ as:

$$k(i,j) = (\phi(i,j) - \phi_o)/a \tag{1}$$

$$l(i,j) = (\theta(i,j) - \theta_o)/a \tag{2}$$

Here, a is a chosen ratio of radians per pixel, and $[\phi_o, \theta_o]$ is the pitch and yaw of the view ray corresponding to the upper-left pixel in the spherical image. We choose a as being the radians-per-pixel of the center pixel in the planar image. We perform this mapping using a precalculated coordinate look-up table.

B. *Sampling features*

Even when the robot is moving straight forward, any feature except for those directly in front of the robot will drift towards the side of the image as the robot drives by. If we are to interpret the horizontal motion of features as robot rotation, we must limit our features to those for which this frame-to-frame "parallax drift" is less than half a pixel in the spherical image, and therefore undetectable. As shown in figure 2, this constraint defines a minimum value for a feature's distance r as a function of its bearing θ, the maximum possible travel of the camera between frames d, and the yaw subtended by a pixel in the spherical image θ_p:

$$r_{min} = \frac{d\tan(\theta_p/2 + \theta)}{\cos\theta\tan(\theta_p/2 + \theta) - \sin\theta} \tag{3}$$

With a sufficiently reliable stereo vision system with which to estimate r, one could threshold all features in the image by their distance. However, the threshold r_{min} increases with bearing θ, and stereo depth is unreliable for faraway features at low image resolutions. We therefore opt to limit our sampling to a small range of yaw $-\theta_{max} < \theta < \theta_{max}$ centered on the frontal direction $\theta = 0$. We then substitute θ_{max} for θ in equation 3 to derive a corresponding minimum distance r_{min}. Any feature closer than r_{min} is deemed unfit for use. When an insufficient number of viable landmarks are found in a particular frame, the pose for that frame is estimated using wheel odometry. In practice, this happens relatively rarely outdoors. Even in dense forests such as the one in figure 4(c), the many nearby obstacles (trees) caused our hybrid VO system to defer to wheel odometry on only 7.9% of the frames. We found that this was sufficiently low to maintain good performance on uneven and slippery forest ground. Two exceptions where distal features may be intermittent are areas that are dense with eye-level vegetation, or extremely hilly areas where the terrain regularly rises to eye level within r_{min} meters. The former situation can be mitigated by slowing down to reduce d, and therefore reduce r_{min}, when VO finds itself frequently delegating to wheel odometry. The latter case of severely hilly terrain presents mapping difficulties for any 3-DOF model, though our visual tracker still presents an improvement over wheel odometry for control purposes.

On our system, we estimate distance using the dense stereo image already calculated by another component for the purposes of obstacle detection. If no such calculation is already being done, the approach of [4] may be used, where stereo disparity is calculated for each patch rather than for all pixels. Our use of stereo information is distinguished by its low requirement for depth precision. We need only establish whether a patch is close enough to drift due to parallax between neighboring frames. The lower the resolution, the looser this requirement becomes, as features need to drift farther to cause noticeable pixel shift. This low dependence on depth precision enables our method to operate efficiently on low resolution video. By contrast, pure VO methods that use the depth estimate to measure translation are more sensitive to the high depth uncertainty at low image resolutions.

The choice of θ_{max} can be made based on the robot's expected speed, environment, and camera geometry. On our platform, the cameras point off to the sides, thereby making the frontal direction close to one side of the spherical image (see figure 1(b)). We therefore chose θ_{max} to be the absolute value of the yaw of that side, namely ± 0.09 or ± 0.14 radians depending on the eye. This left 15% to 23% of the horizon image available for sampling. Using these θ_{max} values, along with a d of 0.13 meters (1.3 m/s / 10 Hz), equation 3 yields minimum distances r_{min} of 2.42m and 3.69m.

The region defined by $-\theta_{max} < \theta < \theta_{max}$ is further shrunk on all sides by half the patch dimensions before searching for Harris corners in one of its channels. These corners are used to find suitable points from which to sample RGB patches. The shrinkage is done to ensure that patches centered on these corners are completely contained within $-\theta_{max} < \theta < \theta_{max}$. The horizon images labeled "previous frame" in figure 3 show this region in blue. As each image patch is selected, the Harris corners under that patch are set to zero as a form of non-max suppression. We sample 6 patches measuring 13 x 3 pixels, or 7.6 by 1.75 degrees of solid angle, or 11% by 3% of the field of view. Note that while we sample in a narrow region, the search region is not so constrained. Therefore, θ_{max} does not present a limit on our robot's rotation rate.

(a) Open field (b) Office parking lot

Fig. 3. The robot's view, while running two of the courses in figure 4. "Previous frame" shows the spherically projected horizon image. Harris corners are detected in a region shown in blue, defined either as the frontal direction (figure 3(a)), or as what will become the frontal direction in the current frame, according to wheel odometry (figure 3(b)). Image patches are sampled around the strongest corners. "Current frame" shows their matches in the current frame. "Search windows" shows their search areas, defined to span the patch's position in the previous frame and its position in the current frame as predicted by wheel odometry. Figure 3(b) has wider search windows because the robot is in the middle of a sharp turn. "Patches" shows the isolated patches. The patch framed by yellow dots is a discarded outlier patch. Comparing the yellow rectangle in the previous and current frames, we can see that it has shifted by one pixel relative to the other patches.

C. Searching for patches

In outdoor environments, it is a common occurrence for the robot to rotate less than reported by wheel odometry ("wheel slip"). The opposite case of the robot rotating significantly more than the wheels ("wheel skid") is far less common under vehicle speeds typically operated under by autonomous vehicles [12]. We therefore trust our odometry to set an upper limit on the expected patch motion from frame to frame. Vehicles that do operate at skid-inducing speeds may choose to employ low-resolution whole-image matching, used by Klein et al [18] for efficient high-speed tracking, to provide another prior for the patch locations. When searching for image patches, we limit our search window to a region that encompasses the patch's position in the previous frame and the patch's position in the current frame, as predicted by wheel odometry. This window is inflated on all sides by half the patch dimensions for an added measure of safety. We apply a normalized cross-correlation of the image patch with this search window, and choose the maximum as the best-matching location.

On our platform, the camera used for VO points off to one side, putting the frontal direction near the side of the image. Features sampled from this area are easily scrolled off that frontal side of the screen when the robot turns away from it. We therefore choose the sampling location using wheel odometry. If it indicates a turn away from the frontal side, we sample not from the previous image's frontal region, but from the area that will become the frontal region in the current frame, according to odometry. In order for both these regions to be scrolled off the screen, almost the entire image must be scrolled off-screen in either direction. This is an impossibility on our system, given its maximum turn rate of π radians per second.

D. Cross-validate matches

The change in yaw of an image patch from the previous frame to the current frame is that patch's estimate of the robot's rotation. To detect outliers, we cross-validate by having each patch "vote" for every other patch whose estimate differs by less than θ_p. Patches whose vote tally is more than half the number of patches are deemed reliable, while others are rejected as outliers. The rotation estimate shared by all inliers is taken as the robot's rotation between the previous and current frames. When using a small number of patches, a pair of frames may occasionally present no patches with a sufficient number of votes. On such frames we let the wheel odometer supply the change in pose.

E. Localizing the robot

The robot's current pose is estimated as:

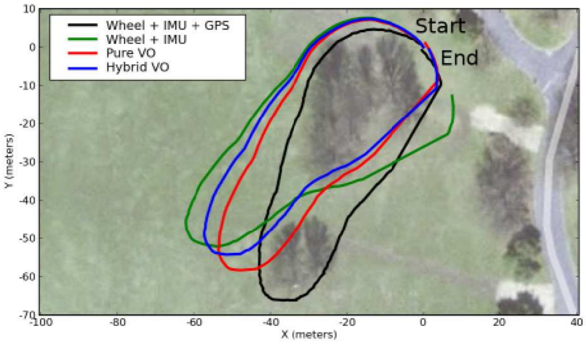
$$p' \quad = \quad p + R_{\frac{\theta_v}{2}} R_{-\theta_w} \Delta p_w \tag{4}$$

Where p' and p are the current and previous pose estimates, Δp_w is the change in pose as reported by wheel odometry, θ_v and θ_w are the change in yaw as reported by visual odometry and wheel odometry, and R_θ is a rotation matrix representing a yaw rotation by θ. The concatenation $R_{\frac{\theta_v}{2}} R_{-\theta_w}$ expresses the fact that we undo the odometry-reported rotation θ_w and replace it with $\frac{\theta_v}{2}$. We use the midpoint method to numerically integrate the pose forward, hence the use of $\frac{\theta_v}{2}$ rather than θ_v.

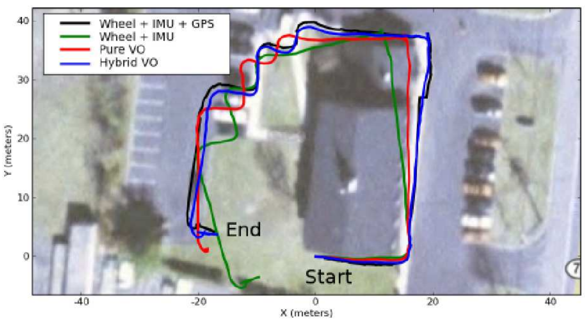
In blending visual and non-visual odometry, we chose not to employ probabilistic model-based sensor fusion techniques, due to the highly non-Gaussian nature of wheel slip in outdoor terrain (see section II-A). However it is simple enough to model the uncertainty of our bearing-only visual odometer, allowing for model-based sensor fusion with other sensors, or with wheel odometry in settings in which it is better behaved. In section III-C, we described how we search for image patches by calculating its normalized cross-correlation within a search area, choosing the peak yaw θ_o as its matching location. We may locally fit a normal distribution $N(\theta_o, \sigma)$ to this peak, finding standard deviation σ by matching the second derivative of $N(\theta_o, \sigma)$ to the numerical second derivative of the cross correlation profile around the peak. The procedure may be repeated for all patches within our consensus set, and their individual uncertainties σ_i may be merged in the standard manner to yield the standard deviation of the bearing estimate: $\sigma_\theta = \left(\sum_i \sigma_{\theta i}^{-1}\right)^{-1}$.

IV. IMPLEMENTATION

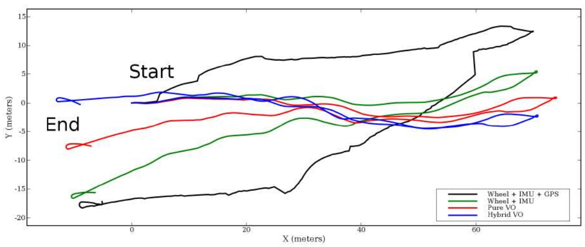
The system described in this paper has been deployed on the DARPA/NREC LAGR platform, an autonomous outdoor rover. The robot runs on two powered wheels and two passive casters, and takes input from wheel encoders, an IMU, and a GPS unit. In addition it takes visual input from



(a) Field Loop: A closed loop around two tree clusters.



(b) Building Lap: A partial loop around a building.



(c) Forest Path: Down a narrow forest path and back.

Fig. 4. Vehicle trajectories, as measured using wheel odometry + IMU, wheel odometry + IMU + GPS, full VO [4], and vision-corrected wheel odometry. Non-GPS trajectories are aligned to the initial orientation estimate given by GPS, which can be noisy. In figure 4(a), the robot's initial pose and final pose are identical. A trajectory's correctness may therefore be evaluated by the size of the opening in the loop. In figure 4(b) the robot follows the sidewalk between the robot's first turn and sixth turn. The sidewalk's shape serves as ground truth during this segment. In figure 4(c), the robot traverses a narrow forest path, then backtracks down the same path. GPS is inaccurate and disruptive in wooded areas such as these.

two stereo camera pairs, pointing slightly to the left and to the right, with fields of view that overlap slightly around the frontal direction. The robot provides three user-accessible computers, one of which runs the VO system described in this paper. We have implemented all software components in Lush, an interpreted language with compilable functions. The visual odometer is entirely compiled, and runs on one of the camera computers. We captured the camera images at a low resolution of 160 x 120 pixels, and used six image patches of 13 x 3 pixels. The Intel Performance Primitives (IPP) library was used for the spherical image transform, Harris corner detection, and normalized cross-correlation. The hybrid VO system runs within the same thread as the short-range stereo-based obstacle detector [19], running at 6 Hz. The processor

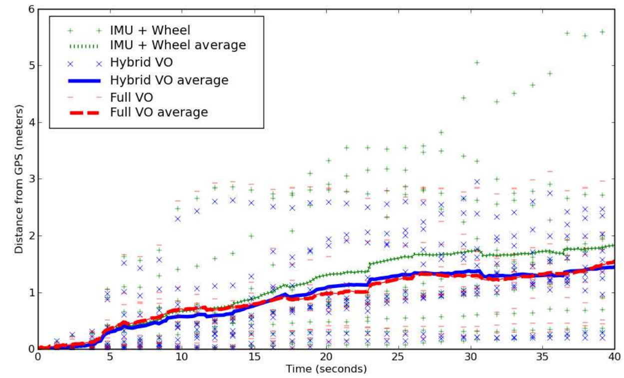


Fig. 5. Drift from GPS position over time. The dots show the distances of estimated trajectories from the GPS position for 10 randomly chosen runs. The lines show the corresponding averages over all runs. The average distance traveled (as measured by GPS) was 30.3 meters.

time is also shared by a long-range (5 to 150 meters) obstacle detector [20] running in a separate thread at 1 Hz. The "IMU + wheel" and "GPS + IMU + wheel" trajectories shown in figure 4 were calculated using tuned EKF pose estimators provided with the platform.

V. RESULTS

The hybrid VO system has been tested on various types of outdoor terrain including the area around an office building, an open field, and a narrow path through a forest. For the results presented here, we recorded logs in these settings, and ran both our hybrid VO and the full 6-DOF VO of [4] on them, as a benchmark for accuracy and processor time. Figure 4 shows the pose trajectories of three runs. Predictably, wheel odometry fused with IMU consistently fares the worst. The EKF fusion of wheel odometry, IMU, and GPS does better, except in the forest where the GPS signal can be both sporadic and inaccurate. Even with clear GPS reception, the GPS-aided trajectory features discontinuities due to satellites coming in and out of reception. Full VO and hybrid VO perform comparably in all three runs, except for figure 4(b), where a forward wheel slip was deliberately induced by running the robot up against a curb that was too high to surmount. All but the full visual odometer are fooled into believing that the robot made it over the curb. The plot in 4(c) shows the robot going down a narrow path through a forest. Accurate, high-frequency estimates of the robot bearing are particularly important in such settings, where false rotations due to wheel slip are frequent, and can cause a robot to counter-steer into entangling obstacles on each side. The hybrid VO retraces the path accurately after a quick 180-degree turn.

Figure 5 shows the drift from GPS (taken here to represent ground truth) over time for wheel odometry, hybrid VO, and pure VO. The data is from 10 randomly chosen runs, manually vetted for inaccurate GPS such as that of figure 4(c).

The CPU cost of each of these runs are reported in table I. The runtimes shown are those of a Pentium 4M laptop at

TABLE I

CPU TIME PER FRAME ON A 2 GHZ PENTIUM 4M FOR FULL VO [4] AND HYBRID VO ON THE THREE COURSES SHOWN IN FIGURE 4.

Log file	Full VO	Hybrid VO + range	Hybrid VO
Field Loop	266 ms	11.5 ms	8.0 ms
Building Lap	238 ms	11.0 ms	8.2 ms
Forest Path	153 ms	14.0 ms	9.3 ms

2 GHz, running off of image and sensor data logged by the robot. The robot's CPUs are approximately 2.5 times faster. The per-frame cost and the fraction of frames successfully processed are shown. While our system does not track translations, it does use range information to rule out features that are too close to the robot. As discussed in section III-B, we get this information "for free" by appropriating the stereo image already calculated by our obstacle classification system. If no such stereo image data is available, we can also adopt the approach of [4], where a patch is searched for in the other stereo camera to estimate distance on a per-patch rather than per-pixel basis. In table I, the runtimes for running just the hybrid VO, and for running the hybrid VO with per-patch stereo matching, are shown separately. The full VO runtime is best compared to the latter.

VI. CONCLUSIONS AND FUTURE WORKS

We have presented an efficient hybrid wheel/visual odometer capable of localizing an autonomous robot in unstructured outdoor terrain at 5 to 10 percent of the computational cost of existing VO systems. Hybrid VO has the potential to enable accurate visual localization on platforms for which previous VO systems are prohibitively demanding. We have tested our system in outdoor terrain of varying visual complexity, including open fields with minimal visual features, and forest paths where GPS is error-prone and roots and leaves make wheel slip frequent.

We are currently working on using bearing-only VO to bootstrap an additional "translation-only VO" step that operates on very nearby patches at the bottom of the image. Such patches have usually been difficult to track, as small features on the ground are often indistinct, whereas larger areas of texture may require a rotation-invariant representation. Using the rotation estimate from bearing-only VO, large ground patches may be rotated in the local ground plane to their expected orientations in the current frame. The distance to such nearby patches may be estimated with stereo vision at a reasonable accuracy even at low resolutions. We expect such a 3-DOF VO system to be far cheaper than existing VO methods, which estimate translation and rotation simultaneously, requiring higher resolutions.

VII. ACKNOWLEDGMENTS

This work was supported by the DARPA LAGR Program (Learning Applied to Ground Robots), under contract HR001105C0038. The authors thank Kurt Konolige and Motilal Agrawal at SRI for use of their visual odometer, and gratefully acknowledge the contributions of their teammates at Net-Scale Technologies and NYU.

REFERENCES

[1] S. Thrun, *Exploring artificial intelligence in the new millennium*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, ch. Robotic mapping: a survey, pp. 1–35.

[2] C. Bibby and I. Reid, "Simultaneous localisation and mapping in dynamic environments (slamide) with reversible data association," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," *Proc. of Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 01, pp. 652–659, 2004.

[4] M. Agrawal and K. Konolige, "Rough terrain visual odometry," *Proceedings of the International Conference on Advanced Robotics (ICAR)*, August 2007.

[5] H. Schäfer, P. Hahnfeld, and K. Berns, "Real-time visual self-localisation in dynamic environments," in *Autonome Mobile Systeme*, 2007.

[6] A. Brown and D. Sullivan, "Inertial navigation electro-optical aiding during gps dropouts," in *Proceedings of the Joint Navigation Conference*, 2002.

[7] E. Andersen and C. Taylor, "Improving mav pose estimation using visual information," in *Proc. of Int'l Conf on Intelligent Robots and Systems (IROS)*. IEEE, 2007, pp. 3745–3750.

[8] M. Veth, J. Raquet, and M. Pachter, "Stochastic constraints for efficient image correspondence search," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 42, no. 3, pp. 973–982, July 2006.

[9] L. Ojeda, D. Cruz, G. Reina, and J. Borenstein, "Current-based slippage detection and odometry correction for mobile robots and planetary rovers," *IEEE Transactions on Robotics and Automation*, vol. 22, no. 2, 2006.

[10] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers: Field reports," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[11] C. C. Ward and K. Iagnemma, "Classification-based wheel slip detection and detector fusion for outdoor mobile robots," in *Proc. of Int'l Conf. on Robotics and Automation (ICRA)*. IEEE, 2007, pp. 2730–2735.

[12] ——, "Model-based wheel slip detection for outdoor mobile robots," in *Proc. of Int'l Conf. on Robotics and Automation (ICRA)*. IEEE, 2007, pp. 2724–2729.

[13] H. D.-W. Salah Sukkarieh, Eduardo Nebot, "A high integrity imu/gps navigation loop for autonomous land vehicle applications," in *IEEE Transactions on Robotics and Automation*, vol. 15, June 1999.

[14] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*, 5th ed. Springer-Verlag, 2001.

[15] *GPS16/17 Series Technical Specifications*, Garmin International, Inc., 2005.

[16] C. Harris and M. Stephens, "A combined corner and edge detector,," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[18] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. 10th European Conference on Computer Vision (ECCV'08)*, Marseille, October 2008, pp. 802–815.

[19] P. Sermanet, R. Hadsell, M. Scoffier, M. Grimes, J. Ben, A. Erkan, C. Crudele, U. Muller, and Y. LeCun, "A multi-range architecture for collision-free off-road robot navigation," *Journal of Field Robotics*, 2009, to appear.

[20] R. Hadsell, P. Sermanet, M. Scoffier, A. Erkan, K. Kavackuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, 2009, to appear.