

Outdoor Mapping and Navigation using Stereo Vision

Kurt Konolige, Motilal Agrawal, Robert C. Bolles, Cregg Cowan, Martin Fischler,
and Brian Gerkey

Artificial Intelligence Center, SRI International, Menlo Park, CA 94025

1 Introduction

We consider the problem of autonomous navigation in an unstructured outdoor environment. The goal is for a small outdoor robot to come into a new area, learn about and map its environment, and move to a given goal at modest speeds (1 m/s). This problem is especially difficult in outdoor, off-road environments, where tall grass, shadows, deadfall, and other obstacles predominate. Not surprisingly, the biggest challenge is acquiring and using a reliable map of the new area. Although work in outdoor navigation has preferentially used laser rangefinders [13, 2, 6], we use stereo vision as the main sensor. Vision sensors allow us to use more distant objects as landmarks for navigation, and to learn and use color and texture models of the environment, in looking further ahead than is possible with range sensors alone.

In this paper we show how to build a consistent, globally correct map in real time, using a combination of the following vision-based techniques:

- Efficient, precise stereo algorithms. We can perform stereo analysis on 512x384 images in less than 40 ms, enabling a fast system cycle time for real-time obstacle detection and avoidance.
- Visual odometry for fine registration of robot motion and corresponding obstacle maps. We have developed techniques that run at 15 Hz on standard PC hardware, and that provide 4% error over runs of 100 m. Our method can be integrated with information from inertial (IMU) and GPS devices for robustness in difficult lighting or motion situations, and for overall global consistency.
- A fast RANSAC [3] method for finding the ground plane. The ground plane provides a basis for obstacle detection algorithms in challenging outdoor terrain, and produces high-quality obstacle maps for planning.
- Learning color models for finding paths and extended ground planes. We learn models of the ground plane and path-like areas on and off-line, using a combination of geometrical analysis and standard learning techniques.
- Sight-line analysis for longer-range inference. Stereo information on our robot is unreliable past 8m, but it is possible to infer free space by finding “sight lines” to distant objects.

Good map-building is not sufficient for efficient robot motion. We have developed an efficient global planner based on previous gradient techniques [10], as well as a novel local controller that takes into account robot dynamics, and searches a large space of robot motions.

While we have made advances in many of the areas above, it is the integration of the techniques that is the biggest contribution of the research. The validity of our approach is tested in blind experiments, where we submit our code to an independent testing group that runs and validates it on an outdoor robot. In the most recent tests, we finished first out of a group of eight teams.

1.1 System overview

This work was conducted as part of the DARPA Learning Applied to Ground Robotics (LAGR) project. We were provided with two robots (see Figure 1, each with two stereo devices encompassing a 110 degree field of view, with a baseline of 12 cm. The robots are near-sighted: depth information degrades rapidly after 6m. There is also an inertial unit (IMU) with angular drift of several degrees per minute, and a WAAS-enabled GPS. There are 4 Pentium-M 2 GHz computers, one for each stereo device, one for planning and map-making, and one for control of the robot and integration of GPS and IMU readings. In our setup, each stereo computer performs local map making and visual odometry, and sends registered local maps to the planner, where they are integrated into a global map. The planner is responsible for global planning and reactive control, sending commands to the controller.

In the following sections, we first discuss local map creation from visual input, with a separate section on learning color models for paths and traversable regions. Then we examine visual odometry and registration in detail, and show how consistent global maps are created. The next section discusses the global planner and local controller. Finally, we present performance results for several tests in Spring 2006.



Fig. 1. *Left: LAGR robot with two stereo sensors. Right: Typical outdoor scene as a montage from the two left cameras, $\sim 110^\circ$.*

1.2 Related work

There has been an explosion of work in mapping and localization (SLAM), most of it concentrating on indoor environments [7, 11]. Much of the recent research on outdoor navigation has been driven by DARPA projects on mobile vehicles [2]. The

sensor of choice is a laser rangefinder, augmented with monocular or stereo vision. In much of this work, high-accuracy GPS is used to register sensor scans; exceptions are [6, 13]. In contrast, we forego laser rangefinders, and explicitly use image-based registration to build accurate maps. Other approaches to mapping with vision are [18, 19], although they are not oriented towards realtime implementations. Obstacle detection using stereo has also received some attention [18]. There have been a number of recent approaches to visual odometry [15, 16]. Our system is distinguished by realtime implementation and high accuracy using a small baseline in realistic terrain. We discuss other examples of related work in the text.

2 Local map construction

The object of the local map algorithms is to determine, from the visual information, which areas are freespace and which are obstacles for the robot: the *local map*. Note that this is not simply a matter of geometric analysis – for example, a log and a row of grass may have similar geometric shapes, but the robot can traverse the grass but not the log.

Figure 2 is an outline of visual processing, from image to local map. There are four basic trajectories. From the stereo disparity, we compute a nominal ground plane, which yields free space near the robot. We also analyze height differences from the ground to find obstacles. Via the technique of sight lines we can infer freespace to more distant points. Finally, from color and path analysis, coupled with the ground plane, we determine paths and traversability of the terrain.

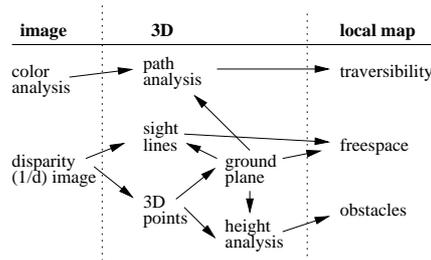


Fig. 2. Visual processing diagram. The paths from visual input to depict the processing flow in constructing the local map.

2.1 Stereo analysis and ground plane extraction

We use a fast stereo algorithm [9] to compute a disparity image at 512x384 resolution (Figure 3, left). In typical outdoor scenes, it is possible to achieve very dense stereo results. The high resolution gives very detailed 3D information for finding the ground plane and obstacles. Each disparity image point $[u, v, d]$ corresponds to a 3D point in the robot's frame.

The most important geometric analysis is finding the ground plane. Although it is possible to detect obstacles using local variation in height [ref], using a ground plane

simplifies processing and yields more stable results. To extract a ground plane, we use a RANSAC technique [3], choosing sets of 3 noncollinear points. Hypothesized planes are ranked by the number of points that are close to the plane. Figure 3 shows an example, with a green overlay indicating the inliers. Points that lie too high above the ground plane, but lower than the robot's height, are labeled as obstacles. This method is extremely simple, but has proven to work well in practice, even when the ground has modest dips and rises; one reason is that it only looks out to 6m around the robot. A more sophisticated analysis would break the ground plane into several segments or model more complex shapes.

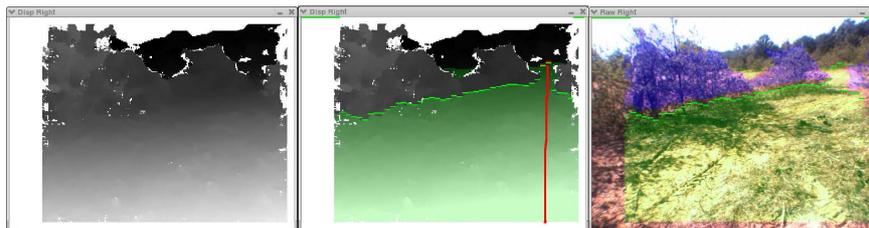


Fig. 3. Left: disparity image from the left view of the robot in Figure 1. Closer pixels are lighter. Middle: extracted ground plane, in green overlay. Limit of ground plane is shown by green bar; sight line has a red bar. Right: Ground plane overlaid on original image, in green. Obstacles are indicated in purple.

2.2 Sight lines

Although we cannot precisely locate obstacles past 6-8m, we can determine if there is freespace, using the following observation. Consider the interpreted image of Figure 3, middle. There is a path that goes around the bushes and extends out a good distance. The ground plane extends over most of this area, and then ends in a distant line of trees. The trees are too far to place with any precision, but we can say that *there is no obstacle along the line of sight to the trees*. Given a conservative estimate for the distance of the trees, we can add freespace up to this estimate. The computation of sight lines is most efficiently accomplished in disparity space, by finding columns of ground plane pixels that lead up to a distant obstacle (red line in Figure 3 middle). Note that the example sight line follows the obvious path out of the bushes.

2.3 Learning color models

We can extend the local map to regions which are beyond the stereo range. Our main learning algorithm is AdaBoost [5]. The input to AdaBoost is a set of labelled training data; AdaBoost calls uses a weak learning method as an iteration step, and focuses on incorrectly classified examples. In our case, the weak learner is taken as a decision stump (decision trees with two leaves) [21].

In offline learning, we focus on learning *paths*, which are characterized by color and geometrical properties. Training samples come from teleoperation, assisted by

an automatic analysis of the route and prospective paths. We sample images at approximately one meter intervals; since our robot is well-localized by visual odometry, we can project the route of the robot on each of the sampled images and calculate the distance of each pixel from the route centerline. Figure 4(b) shows the projected route of the robot in red. The green region corresponds to pixels which are 2 meters or closer to the robot's trajectory. This route corresponds to the robot's traversal on the mulch path shown in Figure 4(a). For each image, we form a gaussian model of the normalized color in several regions close to the robot, and use this model to classify all pixels in the image. If the classified region conforms to a path-like shape, we use it as input to the learning algorithm. Figure 4(c) shows the geometrical path in yellow.

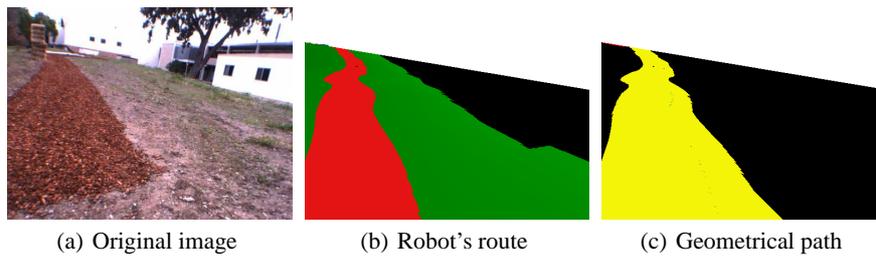


Fig. 4. *Illustration of offline learning.*

Once the pixels belonging to the path are marked in each image, we use two-class AdaBoost to learn the color models for path and non-path pixels. From the model, we construct an RGB lookup table for fast classification, using AdaBoost on each color triplet in the table. During online runs, classification of a pixel is a simple table lookup.

We can also use this algorithm for online learning of more general terrain. We use the current image and the corresponding ground plane/obstacles marked by stereo within close range to classify the pixels beyond stereo range in the same image. This is a standard two class classification problem - the two classes being ground plane and obstacles. We use the AdaBoost algorithm (described before) to learn these two classes and then classify the pixels beyond the stereo range.

2.4 Results

The combined visual processing results in local maps that represent traversability with a high degree of fidelity. Figure 5 shows the results of an autonomous run of about 130m, over a span of 150 seconds. We used offline learning of mulch paths on a test site, then used the learned models on the autonomous run. The first part of the run was along a mulch path under heavy tree cover, with mixed sunlight and deep shadows. Cells categorized as path are shown in yellow; black is freespace. Obstacles are indicated by purple (for absolute certainty), and white-to-gray for decreasing certainty. We did not use sight lines for this run.

The path did not lead directly to the goal, and there were many opportunities for the robot to head cross-country. About two-thirds of the way through the run, no more paths were available, and the robot went through heavy grass and brush to the goal. The robot's pose, as estimated from filtered visual odometry (see the next section), is in green; the filtered GPS path is in yellow. Because of the tree cover, GPS suffered from high variance at times.

A benefit of using VO is that wheel slips and stalls are easily detected, with no false positives. For example, at the end of the run, the robot was caught on a tree branch, spinning its wheels. The filtered GPS, using wheel odometry, moved far off the global pose, while the filtered VO stayed put.

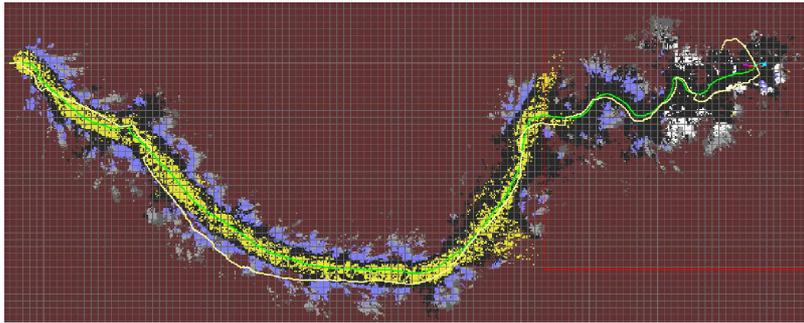


Fig. 5. Reconstruction on a 130m autonomous run. Yellow is recognized path, black is freespace, and purple, gray and white are obstacles.

3 Constructing consistent global maps

In this section we provide solutions to two problems: representing and fusing the information provided by visual analysis, and registering local maps into a consistent global map.

3.1 Map representation

For indoor work, a standard map representation is a 2D *occupancy grid* [14], which gives the probability of each cell in the map being occupied by an obstacle. Alternatives for outdoor environments include 2.5D elevation maps and full 3D voxel maps [8]. These representations can be used to determine allowable kinematic and dynamic paths for an outdoor robot in rough terrain. We choose to keep the simpler 2D occupancy grid, foregoing any complex calculation of the robot's interaction with the terrain. Instead, we abstract the geometrical characteristics of terrain into a set of categories, and fuse information from these categories to create a *cost* of movement.

We use a grid of 20cm x 20cm cells to represent the global map. Each cell has a probability of the belonging to the four categories derived from visual analysis (Section 2): obstacle, ground plane freespace, sight line freespace, and path freespace. Note that these categories are not mutually exclusive, since, for example, a cell under an overhanging branch could have both path and obstacle properties. We are

interested in converting these probabilities into a cost of traversing the cell. If the probabilities were mutually exclusive, we would simply form the cost function as a weighted sum. With non-exclusive categories, we chose a simple prioritization schedule to determine the cost. Obstacles have the highest priority, followed by ground plane, sight lines, and paths. Each category has its own threshold for significance: for example, if the probability of an obstacle is low enough, it will be ignored in favor of one of the other categories. The combination of priorities and thresholds yields a very flexible method for determining costs. Figure 5 shows a color-coded version of computed costs.

3.2 Registration and visual odometry

The LAGR robot is equipped with a GPS that is accurate to within 3 to 10 meters in good situations. GPS information is filtered by the IMU and wheel encoders to produce a more stable position estimate. However, because GPS drifts and jumps over time, it is impossible to differentiate GPS errors from other errors such as wheel slippage, and the result is that local maps cannot be reconstructed accurately. Consider the situation of Figure 6. Here the robot goes through two loops of 10m diameter. There is a long linear feature (a low wall) that is seen as an obstacle at the beginning and end of the loops. Using the filtered GPS pose, the position of the wall shifts almost 2m during the run, and obstacles cover the robot's previous tracks.



Fig. 6. Three stages during a run using GPS filtered pose. Obstacle points are shown in white, freespace in black, and the yellow line is the robot's path. The linear feature is marked by hand in red in all three maps, in its initial pose.

Our solution to the registration problem is to use *visual odometry* (VO) to ensure local consistency in map registration. Over larger regions, filtering VO with GPS information provides the necessary corrections to keep errors from growing without bounds. We describe these techniques in the next two sections.

The LAGR robot presents a challenging situation for visual odometry: wide FOV and short baseline make distance errors large, and a small offset from the ground plane makes it difficult to track points over longer distances. We have developed a robust visual odometry solution that functions well under these conditions. We briefly describe it here; for more details consult [1].

Our visual odometry system uses feature tracks to estimate the relative incremental motion between two frames that are close in time. Corner feature points are detected in the left image of each stereo pair and tracked across consecutive frames. Figure 7 illustrates the detection and tracking of feature points in two images. We use a RANSAC method to find the best set of matching features. As in the ground plane analysis, any three matched features determine a motion hypothesis; we score the hypothesis using the pixel reprojection errors in both the cameras. In a final step, the best motion hypothesis is refined in a nonlinear minimization step.

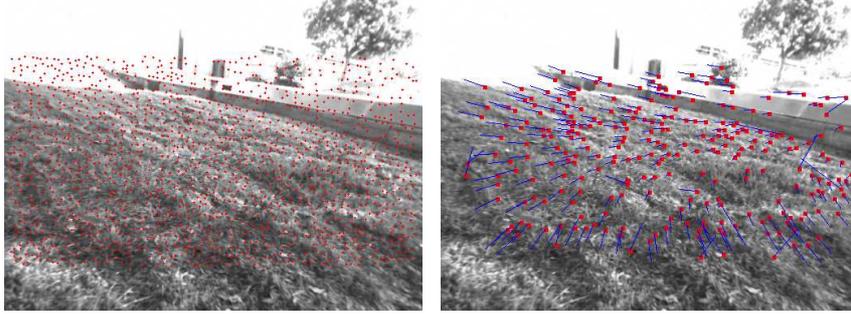


Fig. 7. Example of visual odometry. Left: dense set of detected features. Right: motion of successfully tracked features.

The IMU and the wheel encoders are also used to fill in the relative poses when visual odometry fails. This happens due to sudden lighting changes, fast turns of the robot or lack of good features in the scene (e.g. blank wall).

3.3 Global consistency

Relative motions between consecutive frames are chained together to obtain the absolute pose at each frame. Obviously, this is bound to result in accumulation of errors and drifting. We use GPS to correct the pose of the vehicle through a simple linear filter. Pose information is used when the GPS receiver has at least a 3D position fix, and heading information is used only when the vehicle is travelling 0.5 m/s or faster, to limit the effect of velocity noise from GPS on the heading estimate. In addition, GPS measurements are used only if the robot has travelled a minimum distance from the last GPS measurement. The filter nudges the VO pose towards global consistency, while maintaining local consistency. Over larger loops, of course, the 3 m deviation of the GPS unit means that the map may not be consistent. In this case, other techniques such as wide-baseline image matching [12] would have to be employed.

The quality of the registration from filtered VO, shown in Figure 8, can be compared to the filtered GPS of Figure 6. The low wall, which moved almost 2m over the short loops when using GPS, is much more consistent when VO is employed. And in cases where GPS is blocked or degraded, such as under heavy tree cover in Figure 5, VO still produces maps that are locally consistent. It also allows us to determine wheel slips and stalls with almost no false positives – note the end of the run

Table 1. Loop closure error.

Run Number	1	2	3	4
Distance(meters)	82.4	141.6	55.3	51.0
Method	Percentage Error			
Vehicle Odometry	1.3	11.4	11.0	31.0
Raw Visual Odometry	2.2	4.8	5.0	3.9
Visual Odometry & GPS	2.0	0.3	1.7	0.9

Table 2. Run times for baseline (BL) and SRI.

	Test 12		Test 13	
	BL	SRI	BL	SRI
Run 1	5:25	1:46	5:21	2:28
Run 2	5:34	1:50	5:04	2:12
Run 3	5:18	1:52	4:45	2:12

in Figure 5, where the robot was hung up and the wheels were slipping, and wheel odometry produced a large error.

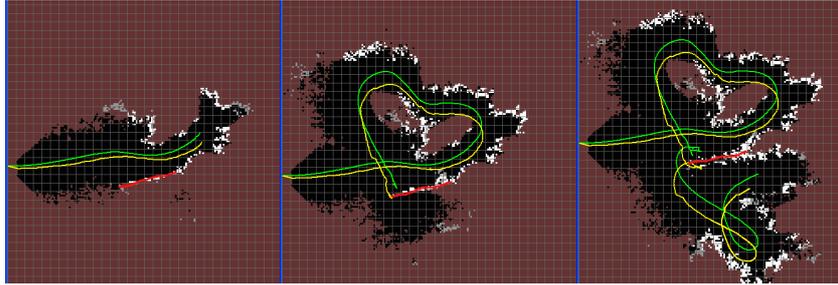


Fig. 8. VO in the same sequence as Figure 6. GPS filtered path in yellow, VO filtered path is in green.

3.4 Results of visual odometry

We have implemented and tested our integrated pose system on several outdoor terrains. Since GPS is accurate to only about 3-4 meters, in order to validate our results, the robot was moved in a closed loop on a typical outdoor environment over 50–100 m, and used the error in start and end poses.

Table 1 compares this error for vehicle odometry (IMU + wheel odometry), visual odometry and the GPS integrated visual odometry for four loops. Except for the first loop, visual odometry outperforms the vehicle odometry, even without GPS filtering, and is comparable to the std of GPS (3m). VO substantially outperformed odometry in 3 of the 4 loops, mainly because of turns and wheel slippage during those runs. This is especially evident in loop 4, where the robot was slipping in mud for a substantial amount of time.

4 Planning and Control

The LAGR robot was provided with a “baseline” system that used implementations of D^* [20] for global planning and DWA [4] for local control. Using this system, we (as well as other teams) had frequent crashes and undesirable motion. The main

causes were the slowness of the planner and the failure of the controller to sufficiently account for the robot's dynamics.

We replaced the baseline system with a global gradient planner and a local trajectory-rollout controller, both described below. This system safely controls the LAGR robot at its maximum speed of 1.3 m/s, with the robot flawlessly negotiating maze-like obstacle courses at an average speed of 1.1 m/s.

4.1 Global planning

For global planning, we re-implemented a gradient planner [10, 17] that computes optimal paths from the goal to the robot, given the cost map. The algorithm has several unique modifications:

- Unlike other implementations, it uses a true Euclidean metric, rather than a Manhattan or diagonal metric.
- The calculation is extremely efficient, using threshold-based queues, rather than a best-first update, which has high overhead.
- It computes the configuration space for a circular robot, and includes safety distances to obstacles.
- Rapid switching of global paths is avoided by including hysteresis - lowering the cost along the path.

Typically we run the global planner within a subregion of the whole map, since the robot is continuously moving towards the goal and encountering new areas. On longer runs, up to 200m, we use an 80m x 80m area; the global planner runs in about 30 ms in this region. The global planner is conservative in assuming the robot to be circular, with a diameter equal to the length of the robot. Also, it does not take into account the nonholomic nature of the robot's motion. Instead, we rely on a local controller to produce feasible driving motions.

4.2 Local control

Given the global cost information produced by the planner, we must decide what local controls to apply to the robot to drive it toward the goal. Algorithms such as DWA compute these controls by first determining a target trajectory in position or velocity space (usually a circular arc or other simple curve), then inverting the robot's dynamics to find the desired velocity commands that will produce that trajectory.

We take the opposite approach: instead of searching the space of feasible *trajectories*, we search the space of feasible *controls*. As is the case with most differentially-driven platforms, the LAGR robot is commanded by a pair $(\dot{x}, \dot{\theta})$ of translational and rotational velocities. The 2-D velocity space is bounded in each dimension by limits that reflect the vehicle's capabilities. Because we are seeking *good*, rather than *optimal*, control, we sample, rather than exhaustively search, this rectangular velocity region. We take a regular sampling (~ 25 in each dimension, ~ 625 total), and for each sample simulate the effect of applying those controls to the robot over a short time horizon ($\sim 2s$).

The simulation predicts the robot's trajectory as a sequence of 5-D $(x, y, \theta, \dot{x}, \dot{\theta})$ states with a discrete integral approximation of the vehicle's dynamics, notably the

acceleration limits. The resulting trajectories, projected into the (x, y) plane, are smooth, continuous, 2-D curves that, depending on the acceleration limits, may not be easily parameterizable (e.g., for the LAGR robot, the trajectories are not circular arcs). Each simulated trajectory t is evaluated by the following weighted cost:

$$C(t) = \alpha \text{Obs} + \beta \text{Gdist} + \gamma \text{Pdist} + \delta \frac{1}{\dot{x}^2}$$

where Obs is the sum of grid cell costs through which the trajectory takes the robot; $Gdist$ and $Pdist$ are the estimated shortest distances from the endpoint of the trajectory to the goal and the optimal path, respectively; and \dot{x} is the translational component of the velocity command that produces the trajectory. This cost calculation is very fast because the computation of cell costs and distances was done by the planner, allowing the controller to access them via table lookup. We choose the trajectory t for which $C(t)$ is minimized, which leads our controller to prefer trajectories that: (a) remain far from obstacles, (b) go toward the goal, (c) remain near the optimal path, and (d) drive fast. Trajectories that bring any part of the robot into collision with a lethal obstacle are discarded as illegal.

5 Performance

For the LAGR program, the independent testing group ran monthly blind demos of the perception and control software developed by eight teams and compared their performance to a baseline system. The last three demos (11, 12, and 13) were considered tests of performance from the first 18 months of the program. The SRI Team was first in the last two tests (Test 12 and 13), after being last in Test 11. Most of the problems in Test 11 were caused by using the baseline planner and controller, and we switched to the one described in the paper for Tests 12 and 13. Figure 2 shows the times for the runs in these tests. We achieved the short run times in Tests 12 and 13 through a combination of precise map building, high-speed path planning, and careful selection of robot control parameters. Our average speed was over 1.1 m/s, while the robot top speed was limited to 1.3 m/s. Map building relied on VO to provide good localization, ground-plane analysis to help detect obstacles, and sight lines to identify distant regions that are likely to be navigable.

6 Conclusion

We have demonstrated a complete autonomous system for off-road navigation in unstructured environments, using stereo vision as the main sensor. The system is very robust - we can typically give it a goal position several hundred meters away, and expect it to get there. But there are hazards that are not dealt with by the methods discussed in this paper: water and ditches are two robot-killers. Finally, we would like to use visual landmarks to augment GPS for global consistency, because it would give finer adjustment in the robot's position, which is critical for following routes that have already been found.

References

1. M. Agrawal and K. Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *Intl. Conf. of Pattern Recognition (ICPR)*, 2006. To appear.
2. P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin. Terrain perception for DEMO III. In *Proc. of the IEEE Intelligent Vehicles Symp.*, 2000.
3. M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. ACM.*, 24:381–395, 1981.
4. D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.
5. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Computer and System Sciences*, 55(1):119–139, 1997.
6. J. Guivant, E. Nebot, and S. Baiker. High accuracy navigation using laser range sensors in outdoor applications. In *ICRA*, 2000.
7. J. S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *CIRA*, 1999.
8. K. Iagnemma, F. Genot, and S. Dubowsky. Rapid physics-based rough-terrain rover planning with sensor and control uncertainty. In *ICRA*, 1999.
9. K. Konolige. Small vision systems: hardware and implementation. In *Intl. Symp. on Robotics Research*, pages 111–116, 1997.
10. K. Konolige. A gradient method for realtime robot control. In *IROS*, 2000.
11. J. J. Leonard and P. Newman. Consistent, convergent, and constant-time slam. In *IJCAI*, 2003.
12. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. of Computer Vision*, 60(2):91–110, 2004.
13. M. Montemerlo and S. Thrun. Large-scale robotic 3-d mapping of urban structures. In *ISER*, 2004.
14. H. Moravec and A. Elfes. High resolution maps for wide angles sonar. In *ICRA*, 1985.
15. D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *CVPR*, 2004.
16. C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone. Robust stereo ego-motion for long distance navigation. In *CVPR*, 2000.
17. R. Philippsen and R. Siegwart. An interpolated dynamic navigation function. In *ICRA*, 2005.
18. A. Rankin, A. Huertas, and L. Matthies. Evaluation of stereo vision obstacle detection algorithms for off-road autonomous navigation. In *AUVSI Symp. on Unmanned Systems*, 2005.
19. D. J. Spero and R. A. Jarvis. 3D vision for large-scale outdoor environments. In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, 2002.
20. A. Stentz. Optimal and efficient path planning for partially-known environments. In *ICRA*, volume 4, pages 3310–3317, 1994.
21. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.