A Guide to Vision-Based Map Building

The Practical Implementation of Vision-Based Sensing

In this article, we describe the practical implementation of vision-based sensing for an outdoor real-time robotics platform. The primary motivation for this work is the Georgia Institute of Technology's team for the DARPA-sponsored Learning Applied to Ground Robots (LAGR) project. The LAGR robot (Figure 1) has as its main sensory input two pairs of color cameras mounted on its head. From this input, the task is to plan a path to a known goal point in GPS coordinates. Additional sensors include an inertial navigation unit (IMU), a Garmin GPS receiver, and a front bump switch. The robot's environment is an unknown outdoor course, approximately 150 m in length, through trees, shrubs, ditches, and grasses of varying density.

We focus our discussion on the grid-based map-building process central to the control of the robot. Map building from stereo information has long been a topic of active research [2], [6], [7], [12]. We hope to present our implementation in such a way that it can be clearly understood and recreated by other practitioners.

The Map Building Process

Figure 2 depicts the map building process of our robot. The steps are the following:

1) A framegrabber captures a pair of images from two calibrated cameras.
2) The images pass through a stereo library, which has knowledge of the relative physical geometry of the cameras as well as their intrinsic properties. Output is a depth map, i.e., three-dimensional (3-D) terrain, in a local coordinate frame.
3) The first coordinate transformation step corrects the depth map for the pitch and roll of the robot, based on the IMU. The second coordinate transformation step converts the depth map from the local frame to global, using the robot's yaw and current global position.
4) A derivative operation is applied to the small terrain map.
5) The new information is incorporated into the robot's global maps, including the terrain and derivative estimates as well as marking those points where the robot has measured a derivative.
6) Finally, a list of points where the derivative has changed value is appended to a stored list.

Grabbing Frames

As small, cheap cameras have grown increasingly common, the software/hardware interfaces needed to grab camera frames has become easier to find and use. Video for Linux (v4l) is the standard library for communicating to an off-the-shelf camera. For our project, we used Point Grey Research's Bumblebee to collect camera images.

Stereo Depth Maps

Producing stereo depth maps (i.e., 3-D terrain maps) from two calibrated cameras has been around for decades [3], [10]. We have used Point Grey's Triclops library as well as Stanford Research Institute International's Small Vision System, while several other vendors provide calibration and stereo software and hardware.

Typically, a stereo depth map is generated by matching small patches in the two images and inferring from the relative geometry of the cameras where the corresponding point in space probably lies. On the LAGR robot, two separate Linux machines (2.0 GHz Pentium-M 1GB RAM) are dedicated to perception processing, allowing us to produce depth maps

BY DAVID WOODEN

1070-9932/06/$20.00©2006 IEEE

reliably at 4 Hz. These maps have pixels of 0.1 m × 0.1 m in physical dimension, which is reasonably precise given that the robot is about 1 m in length, and 0.6 m wide.

Coordinate Transformations

The depth map the stereo process produces needs to be corrected through two transformations to be useful for path planning. First, each point is transformed by a rigid-body rotation for the robot's instantaneous pitch and roll, as estimated by the robot's IMU. Second, the map must be rotated for the robot's yaw and translated from the local frame into the global (i.e. GPS-based) coordinate frame.

The first transformation is a simple matter of applying to each 3-D point a rotation matrix

$$R_y = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \quad (1)$$

for a given pitch angle α. Roll and yaw are transformed similarly.

We describe the two coordinate transformations as separate operations simply for the sake of clarity, though they can easily be combined into a single general transformation matrix. It may be the case, however, that separate processes are responsible for the two transformations and therefore desirable that the transformation details be independently encapsulated within these processes.

Though it has been successful in other applications, we forgo the use of simultaneous localization and mapping (SLAM) [1], [12]. SLAM is more challenging with visual inputs, and the Georgia Tech LAGR team has used the available computing resources for other functionality. For example, we have focused instead on improving GPS filtering, detecting wheel slippage, and implementing visual odometry (see, for example, [8]).

Taking a Derivative

From each stereo depth map, we expect to have a 3-D estimation of terrain. Recognizing that our wheeled robot is of limited size and power, we expect that it cannot traverse terrain that is too steep. Moreover, most obstacles we expect to encounter have abrupt changes in slope (e.g., trees). So, over a transformed depth map, we look for places that have a high derivative so that we can plan around them.

There are many edge finding algorithms available, including the Laplacian of Gaussian, Prewitt method, Canny method, etc. [9]. We chose the Sobel operator because it is easy to understand and requires only additions and multiplications to implement. The operation is defined by two 3×3 filters, f_x and f_y:

$$f_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad f_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

We apply these two filters to each point in the local terrain map and compute a pseudoderivative

On the LAGR robot, two separate Linux machines are dedicated to perception processing, allowing us to produce depth maps reliably at 4 Hz.

$$d_{ij} = \sqrt{(s_{ij} \cdot f_x)^2 + (s_{ij} \cdot f_y)^2}, \quad (2)$$

where s_{ij} is a 3×3 piece of the terrain map centered at coordinates (i, j).

Note that we only apply this derivative operation to the transformed local depth map, not the overall global terrain map. The reason for this is that we expect the local map to be internally consistent. On the other hand, small errors in the pitch and roll correction across several successive stereo depth maps tends to produce false derivatives. By using just the local terrain map, the resulting derivative map is more robust to IMU errors.

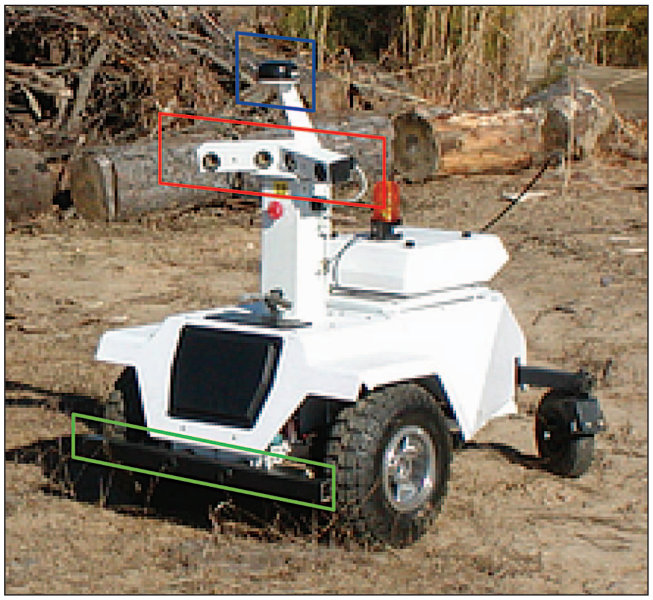

Figure 1. *The LAGR Robot: GPS receiver (blue), stereo cameras (red), and bump switch (green).*

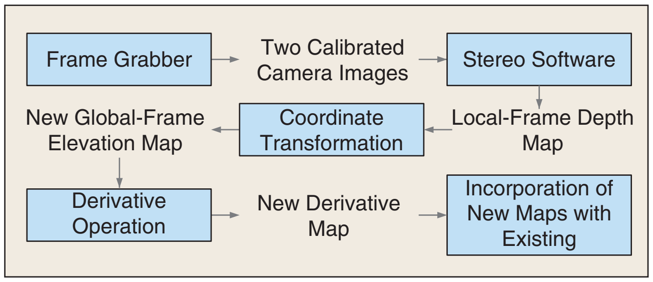
Frame Grabber
Two Calibrated Camera Images
Stereo Software
Local-Frame Depth Map
Coordinate Transformation
New Global-Frame Elevation Map
Derivative Operation
New Derivative Map
Incorporation of New Maps with Existing

Figure 2. *Process overview: from camera images to global maps.*

Our low-level controllers depend on accurate map information, so our map building process needs to run at a high frame rate

In practice we omit the square root operation in (2), noting that we can account for it later when we build a cost map and plan a path. This saves us some computational effort, making the operation very fast. Figure 3 shows an example local terrain and elevation map, corrected for pitch and roll. Identified in the figure are places where good stereo returns were found and where they were absent. Note how the derivative depicted contains high values only for those places where elevation readings were fully present under the 3×3 Sobel filter kernel; the boundary of the elevation map does *not* produce a derivative.

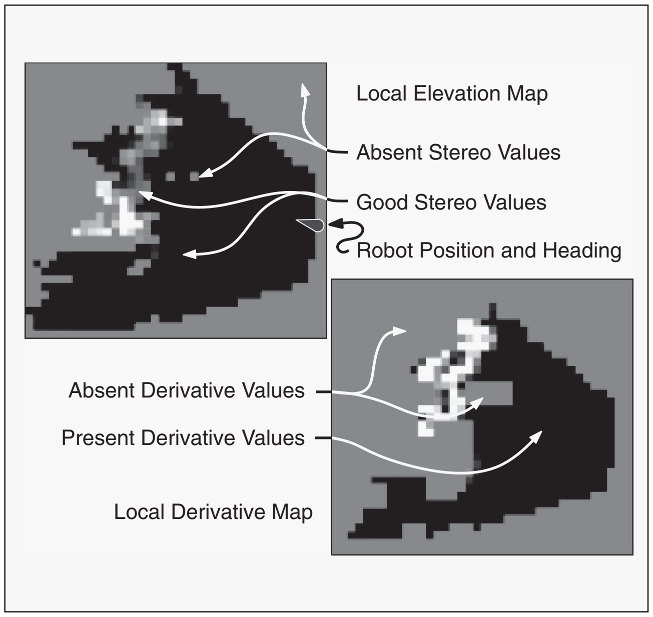

Figure 3. *Example local elevation and derivative maps.*

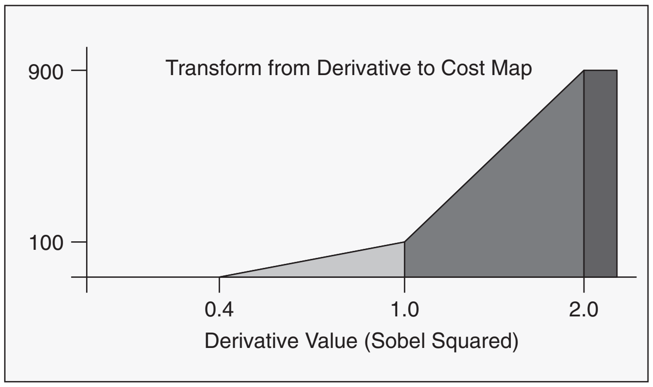

Figure 4. *Piecewise linear transformation of derivative values to cost map values.*

Inasmuch as the Sobel operator requires the local map to have an estimate of all nine pixels under its kernel, holes in the depth map produced by the stereo algorithm may result in an inability to compute a derivative everywhere—there would be holes in our derivative estimate. However, this is of no serious consequence to the overall functioning of the robot; because stereo vision typically operates at more than 2 Hz, it is unlikely that a small number of missed stereo measurements in a single local map will persistently invalidate the derivative map, as the robot drives to the goal.

Incorporating New Measurements

The final step of our map building process is to assign the newly measured terrain and derivative values in the overall global maps. We do not use the terrain map for any purpose in path planning, so we simply overwrite old values with the new and keep it around for display purposes. When putting new derivative measurements into the map, however, we assign the pixel a new value based on a low-pass filter:

$$v_{new} = (1 - \alpha)\ v_{old} + \alpha\ m,$$

where v_{new} is the new value saved in the map, v_{old} is the previous map value, m is the derivative output, and α is a weight $\in [0, 1]$. We set α relatively low, to 0.5, noting that this causes spurious measurements to have little impact and that we receive plenty of terrain updates from the stereo processes.

In addition to terrain and derivative, we maintain one more additional global map, which keeps track of those pixels for which we have computed a derivative. This will be important information in the planning process, when we need to be able to identify terrain that has never been explored.

The Planning Process

We employ a hierarchical control architecture where low-level reactive controllers operate with high frequency and the high-level deliberative functions (like path planning) run slower. This kind of split-level design is not uncommon (see for example [2], [11], [14]). Our low-level controllers depend on accurate map information, so our map building process needs to run at a high frame rate as well—between 15–20 Hz. Path planning is not as time critical, so it runs in a separate thread (operating at around 2 Hz).

For the LAGR project, our planner has taken on two distinct flavors. The first type is an incrementally updating graph structure, as described in [13]. Since this planner is relatively complex, we omit discussion of it here. The second type of planner we have used is simpler: a heuristically improved $A^\star$ grid-based planner [5]. In order to use an $A^\star$ planner, we must construct a cost map, which is fundamentally a transformation of the derivative map into the configuration space of the robot (e.g., accounting for the robot's length and width) and is suitable for grid-based planning.

Our $A^\star$ planner follows a simple three-step process:

1) Grab the set of global-frame map points from the mapping process that have been modified since the last planning cycle.
2) Based on these changed derivative points, update the cost map where needed.
3) Determine a string of global waypoints that the robot should drive to given its current position.

The Cost Map

We transform a derivative measurement into a cost by a heuristically determined piecewise linear transform, shown in Figure 4. This kind of transform is flexible and easy to use when field testing, an important quality since it will likely need tuning, and various aspects of the system upon which the derivative map depends (e.g., density of stereo returns) may be iteratively refined. Recall that we save in the derivative map the output of the Sobel operation squared, which is reflected in the relative slopes of the cost transform.

The value saved in the cost map is the average of the transformed derivative measurements over a 1.2 m × 1.2 m square region. This averaging smooths some of the variation in the derivative map, while accounting for the physical dimension of the robot. In places where there has been no derivative measurement, we use an “unexplored cost” value. This cost is higher than the cost associated with good terrain, causing the robot to tend towards places in the environment where it has seen flat ground.

In addition, the cost map is stored at a lower resolution (one pixel is 0.3 m × 0.3 m) than the terrain and derivative maps. This is because the high resolution, while important for other purposes the derivative map serves, is not needed for path planning. Reducing its resolution greatly increases the speed of the $A^\star$ planner.

$A^\star$ Planning

Given that the robot may start 150 m from its goal location, straight-ahead $A^\star$ planning would be too slow to be usable for the real-time needs of the LAGR robot [5]. However, this deficiency is mitigated by two algorithmic modifications. First, the planner only performs its search over the square region bounded by where the robot has terrain information. So, the planner plans up to this horizon, and then shoots a straight line the remaining distance to the goal. Second, at each iteration, the planner gives the underlying controllers a sparse set of waypoints all the way to the goal. So, if the planner has a long cycle, the lower-level controllers do not have to wait for a new immediate waypoint; they can operate on the up-to-date map information and try to achieve each in the series of waypoints, until the planner can revise the list.

Figure 5 shows a portion of the global terrain, derivative, and cost maps as the robot is in the midst of test run. The goal location is off the map to the lower left. Note how the global terrain map is cluttered with numerous disparate readings, while the derivative map contains a clean representation of obstacles. This apparent difference is due to the computation of derivative only on local maps, which are expected to be internally consistent, while the global terrain map is not.

Stereo vision has provided us with fairly accurate terrain measurements for our experiments, but it suffers from having limited range

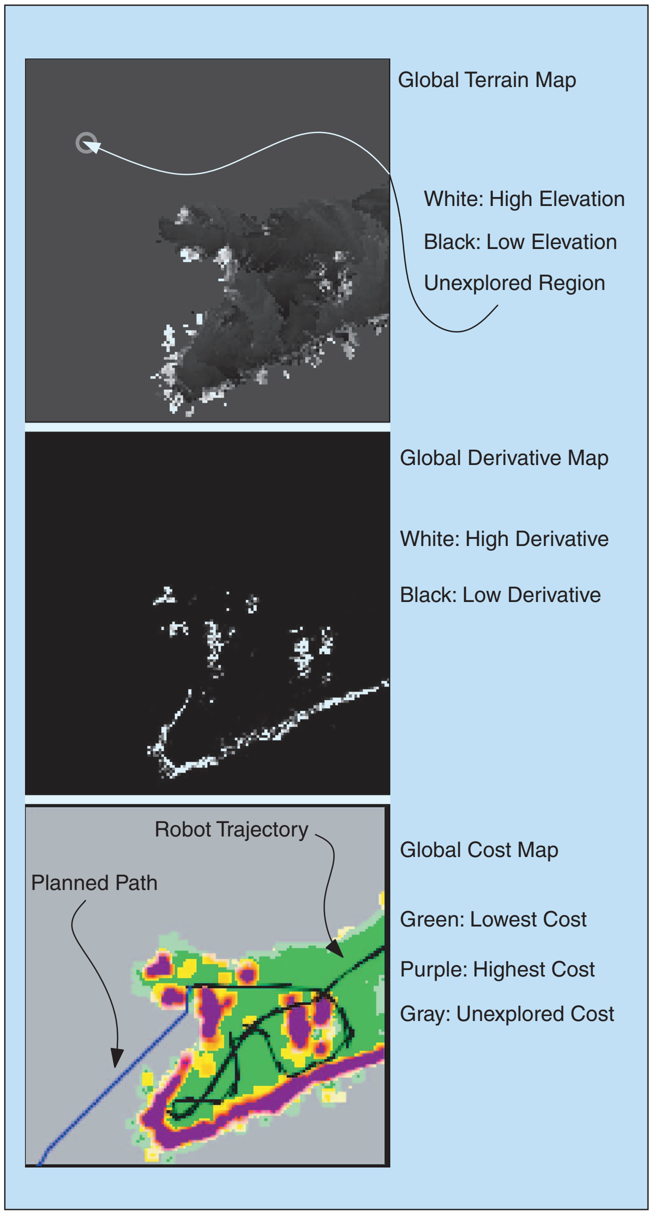

Figure 5. *Snapshot of global terrain, derivative, and cost maps during a run.*

Our team has used color as a valuable signal to indicate likely traversability of terrain outside of stereo range

Additional Sensory Input

Bumps and Slip Detection

Of course, the robot is not strictly limited to stereo information. The robot comes with a bump sensor, which very clearly demarks an intraversable spot in the environment. Similarly, we run a slip detector on IMU/GPS/wheel-amp/wheel-encoder data, to determine points in the environment where the robot loses traction. Also, according to the methodology described in [14], we identify places where the kinematically-informed low-level controllers forbid the robot from entering, despite otherwise clean perception. In all cases, these points are kept in separate lists from the regular terrain and derivative maps and incorporated later in the overall cost map.

Beyond Stereo Vision

Stereo vision has provided us with fairly accurate terrain measurements for our experiments. However, it suffers from having limited range; in our case, the depth maps extend only as far as 6 m from the robot. Yet, the camera images clearly contain much more information about where the robot should try to drive and where it should try to avoid. Our team has used color as a valuable signal to indicate likely traversability of terrain outside of stereo range (see [4]). The stereo map is used to provide a learning signal for how traversable a color value probably is.

Conclusions

We have described a simple vision-based mapping system for mobile robot navigation in an unknown outdoor environment, consistent with the efforts of Georgia Tech's LAGR team. This approach, which has been robust to different environments and responsive to time-critical constraints, is a good example of robotics in practice.

Acknowledgments

The author would like to acknowledge the invaluable advice of Magnus Egerstedt. Additionally, the efforts of Matt Powers, Douglas MacKenzie, and Tucker Balch, along with the entire Georgia Tech LAGR team, are deeply appreciated and without which this work could not have been accomplished.

Keywords

Stereo vision, map building, mobile robotics, unstructured environments.

References

[1] M. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Tran. Robot. Automat.*, vol. 17, no. 3, pp. 229–241, 2001.

[2] M. Hebert, A. Stentz, and C. Thorpe, "Mobility planning for autonomous navigation of multiple robots in unstructured environments," in *Proc. IEEE Int. Symp. Intell. Control,* 1998, pp. 652–657.

[3] A. Izaguirre, P. Pu, and J. Summers, "A new development in camera calibration: calibrating a pair of mobile cameras," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1984, pp. 74–79.

[4] D. Kim, J. Sun, S.M. Oh, J. Rehg, and A. Bobick, "Traversability classification using unsupervised on-line visual learning for outdoor robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.,* to be published.

[5] S.M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge Univ. Press, 2006 [Online]. Available: http://msl.cs.uiuc.edu/planning/.

[6] J. Leonard, H. Durrant-Whyte, and J.J. Cox, "Dynamic map building for autonomous mobile robot," in *Proc. IEEE Int. Workshop Intell. Robots Syst.*, vol. 1, 1990, pp 89–96.

[7] D. Murray and J.J. Little, "Using real-time stereo vision for mobile robot navigation," *Auton. Robots,* vol. 8, no. 2, pp. 161–171, 2000.

[8] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, vol. 1, 2004, pp. I–652–659.

[9] R.W. R.C. Gonzalez, *Digital Image Processing.* Reading, MA: Addison-Wesley, 1992.

[10] L. Robert, M. Buffa, and M. Hebert, "Weakly-calibrated stereo perception for rover navigation," in *Proc. ICCV,* 1995, pp. 46–51.

[11] J.K. Rosenblatt, "DAMN: A distributed architecture for mobile navigation," *J. Experimental Theoretical Artificial Intell.*, vol. 9, no. 2–3, pp. 339–360, Apr.–Sep. 1997. Stanford, CA, 1995.

[12] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2000, pp. 321–328.

[13] D. Wooden and M. Egerstedt, "Oriented visibility graphs: Low-complexity planning in real-time environments," in *IEEE Conf. Robot. Automat.*, to be published.

[14] D. Wooden, M. Powers, D. MacKenzie, T. Balch, and M. Egerstedt, "SCAM: Layered hybrid control with feedback between layers,"*Robot.: Sci. Syst.*, to be published.

David Wooden is pursuing his Ph.D. at the Georgia Institute of Technology, where he received his M.S. (2005) and B.S. (2003), all in electrical engineering. He has worked for the control, automation, and systems integration firm, Engineered Software Products, Lilburn, Georgia, and his current research interests are in path planning, especially real-time graph-based methods as well as in behavior-based control.

Address for Correspondence: David Wooden, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA. Phone: +1 404 385 4077. E-mail: wooden@ece.gatech.edu.