PixelLaser: Computing Range from Monocular Texture

N. Lesperance, M. Leece, S. Matsumoto, M. Korbel, K. Lei, and Z. Dodds

Harvey Mudd College

Abstract. The impressive advances in robotic spatial reasoning over the past decade have relied primarily on rich sensory data provided by laser range finders. Relative to cameras, however, lasers are heavy, bulky, power-hungry, and expensive. This work proposes and evaluates an image-segmentation pipeline that produces range scans from ordinary webcameras. Starting with a nearest-neighbor classification of image patches, we investigate the tradeoffs in accuracy, resolution, calibration, and speed that come from estimating range-to-obstacles using only single images. Experiments atop the low-cost iRobot Create platform demonstrate the accessibility and power of this pixel-based alternative to laser scans.

1 Motivation and Context

Robots' spatial reasoning capabilities have matured a great deal over the past decade. Effective localization, mapping, and navigation algorithms have proven themselves in long-term autonomous driving [3,17], tour-guide [2,15], and office-navigation [8] applications. Robots' most robust and widespread examples of spatial reasoning rely upon the ubiquitous laser range finder (LRF), which uses the time-of-flight of projected laser light to directly compute the range to nearby obstacles within the laser range finder's field of view.

Although effective, LRFs have drawbacks that have prevented them from entering the fast-growing field of low-cost commercially viable autonomous robots. As an alternative, monocular vision offers advantages relative to laser scans across several axes: cameras are less power-hungry, less heavy, less bulky, less range-limited, and, perhaps most importantly, less expensive.

Less is more, however, when it comes to computation. Extracting range from pixel intensities requires far more algorithmic and computational effort than extracting range from time-of-flight. Range-from-vision approaches typically use temporal feature correspondence across a monocular image stream to deduce distance from pixels [6]. This body of work is mature, but it is worth noting that these techniques are most successful when significant spatial context is used to support feature matching. Large patches of pixels facilitate accurate and precise correspondence.

1.1 Related Work

In order Recent approaches have boldly asked, "What can we deduce from only those patches, and not the correspondence at all!?" For instance, Hoiem et al.'s photo

G. Bebis et al. (Eds.): ISVC 2010, Part III, LNCS 6455, pp. 151–160, 2010.

pop-out [4] software and Saxena et al.'s Make3d system [12] yield range at each of a single image's pixels. Spatial grouping, e.g., into edges between groundplane and vertical planes, enable the compelling visualizations those groups have produced.

Such work has seen many robot applications. Horswill's Polly [5] pioneered robotic range-from-texture, and many systems have followed. In [4] Hoiem et al. show confidence levels in terrain navigability; Saxena et al. drive an RC car safely and quickly through rough, natural terrain [12]. Similar to the fast color/texture segmentation work of [1], these projects emphasized machine-learning contributions and task-specific image interpretation. This work, in contrast, focuses on the *accuracy* of the range-to-obstacle scans produced by image segmentation. We hypothesize that image-based scans can, in indoor situations, replace laser scans in the localization, mapping, and navigation algorithms popularized over the past decade.

Thus, we follow the efforts of [14] in which Taylor et al. use color segmentation to produce tabletop-scale range scans, though without summative accuracy results. More recently, Plagemann et al. [11] used Gaussian Processes to learn maps from pixel columns to range. They reported ~1 meter precision, sufficient to support off-the-shelf SLAM algorithms under assumptions common to human-scale indoor environments.

1.2 Contributions

Yet the omnicam images and one-dimensional spatial context (pixel radii) of [11] make a tradeoff against accessibility and range accuracy in favor of field-of-view and angular resolution. This work offers a counterpoint to those efforts by (1) using unmodified webcam images, at much lower cost than omnicam images, (2) estimating groundplane segmentation using 2d image context, rather than pixel radii, and (3) classifying patches via nearest-neighbor matching, rather than Gaussian Processes. Section 2 details this pipeline's algorithms and implementation choices. Reporting on several robot test-runs, Section 3 validates the contributions of these design choices:

- Range accuracy comparable to the best previous results [11] using unmodified webcam images -- sufficient to replace laser scans in many algorithms
- Training efficiency that facilitates quick adaptation to new environments without presuming their visual characteristics
- Execution speed supporting real-time obstacle avoidance

We conclude in Section 4 by reflecting on how pixel-based scans may help bridge the accessibility of low-cost commercial robots with the capabilities of today's high-end experimental platforms.

2 PixelLaser's Algorithmic Pipeline

Figure 1 summarizes the algorithmic pipeline that creates range scans from single images: training, classification, segmentation, and transformation, as detailed below:

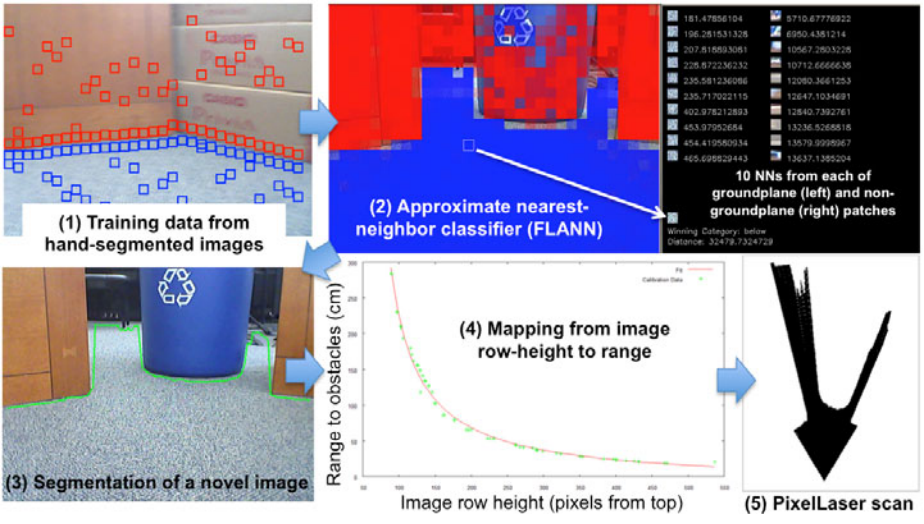

Fig. 1. PixelLaser's pipeline for transforming images into range scans

2.1 Training Data

In order to create a classifier that can distinguish groundplane (*traversable*) from non-groundplane (*untraversable)* image patches, the system requires a segmented set of training images. We obtain these by driving around an environment of interest and then hand-segmenting the results. If another range-finder such as a laser *were* available, automatic segmentation could replace this human-guided step as in [11]. A small number of images, typically 20, suffice for training in a new setting.

The system then decomposes the 640x480 hand-segmented images into 20x20 pixel patches. To emphasize the traversable-to-untraversable terrain transitions, we extract 32 patches immediately above and below the correct segmentation boundary. To help generalize, we select 32 patches further away: the upper left panel of Figure 1 shows the patches chosen from an example image of our laboratory space.

2.2 Feature Extraction and Representation

To represent the 32x4 = 128 patches from each image, we investigated the statistics of the nine 3x3 Laws texture filters [7] and average-color filters in both RGB and HSV space. We investigated the relative classification accuracy of each of these filters both individually and in pairs. The results led us to choose two color-band features, the red and the green means and variances, and two texture statistics, the variance and kurtosis of Laws's sixth filter. Thus, the vector of those six component values represents each 20x20-pixel patch both for classifier training and classification testing.

2.3 Classifier-Building and Classification

We create two approximate nearest-neighbors classifiers from these six-component vectors with the FLANN library [6]. A 20-image training set contains 1280 patches in

each of these classifiers. One classifier holds those patches taken from traversable terrain; the other holds the untraversable terrain, or "obstacles." The system is now prepared to analyze novel images.

To classify a new patch, that patch is first represented by its six-component feature vector P. Next, a predetermined quantity of P's (approximate) nearest neighbors are found from the traversable training patches. The average Euclidean distance d_{trav} from P to these N nearest neighbors is computed. Similarly we compute d_{untrav}, the average distance from P to the N nearest neighbors among the untraversable training patches. Finally, each patch receives an overall *traversability score*, the ratio of d_{trav}/d_{untrav}. Patches whose traversability score is lower than a threshold of 0.8 are considered untraversable. Figure 1's top right panel shows an example of the 10 nearest neighbors used for a particular patch and a visualization of all patches: blue when traversable and red when untraversable.

2.4 Segmentation

Rather than classify *every* possible patch in a novel test image, the system uses a bottom-up search strategy. For a particular column in the image, patches are classified upwards from the bottom row until an untraversable patch is encountered. The resulting locations are connected together to provide the image's segmentation, such as shown in Figure 1's bottom left panel. We can tradeoff speed against accuracy by varying the horizontal resolution at which these computations proceed. Using 32 columns across the image requires only 0.7 seconds – sufficient to support real-time obstacle avoidance. Full-resolution processing, e.g., for more accurate map-building, requires less than 10 seconds per image in the current implementation.

2.5 From Segmentation to Range Scans

Our platform presumes a camera whose height and angle are fixed relative to the groundplane. Thus, there is a one-to-one correspondence between image height and obstacle distance. Rather than model all of its parameters explicitly, we use the fundamental form of this relationship to fit an empirical model that maps image row, measured as an offset from the horizon, to obstacle distance. Figure 1's bottom right panels shows that real data fit this model well; the example illustrates that the resulting scans preserve fine environmental details at full horizontal resolution.

So, how well does the system work – and are the resulting scans useful even in algorithmic contexts designed for *laser* scans? Section 3 shows that both of these questions have promising answers.

3 Experiments, Empirical Validation, and Results

Figure 2 shows our iRobot Create platform whose on-board netbook runs Section 2's algorithmic pipeline. Willow Garage's OpenCV computer vision library provides the image acquisition and image processing through its Python bindings [10]. Similarly, the Python interface to the Fast Library for Approximate Nearest Neighbors, FLANN, provides both the offline classifier training and its online use for testing [9]. A collection of custom Python scripts provides the remainder of the pipeline's processing. This section documents their results.

Fig. 2. The Create platform, netbook, and environment in which we tested PixelLaser's pipeline (left). At right are cropped images from the various test datasets.

In order to vary conditions and parameters in a principled manner, this section first focuses its attention on accuracy results from the environment depicted in Figure 2. This is a large, common indoor space in our laboratory building with a textured carpet subjected to varying lighting conditions and a variety of furnishings and objects forming the obstacles in and around the robot.

3.1 Texture Classification Accuracy

After training on 20 images from a run across the room, we measured the accuracy of the resulting image-patch classifier on 4 datasets of varying difficulty: (A) the training set itself, (B) another run taken under conditions similar to the training set – a nearby trajectory at the same time-of-day (and, thus, similar lighting conditions), (C) a non-overlapping visual trajectory from the same room, and (D) a non-overlapping spatial trajectory in a completely different part of the lab space. Figure 3 illustrates the strong performance of our nearest-neighbor classifier. It achieves over 90% recognition, regardless of the dataset or the number of neighbors used. Varying FLANN's accuracy parameter, the percentage of cases in which the approximate NN is the *actual* NN, affected training time but not classification accuracy.

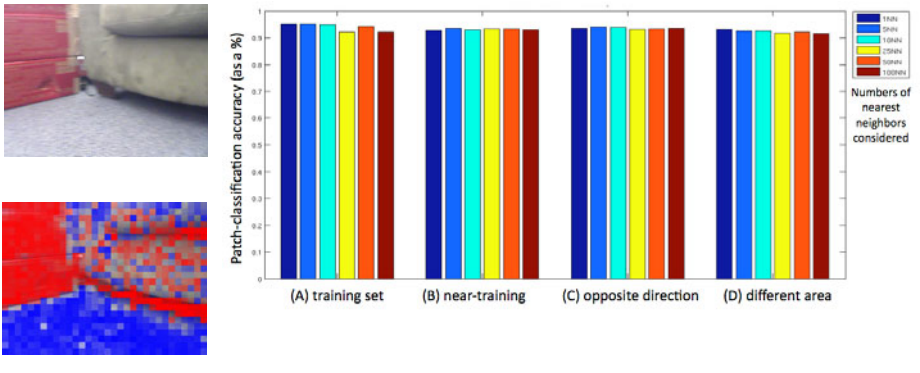

Fig. 3. The accuracy of the FLANN classifier in determining whether image patches are groundplane or not. At left is a test image and the resulting classification of its patches: blue indicates groundplane patches and red indicates those considered untraversable. At right are the classification accuracies across four datasets and five quantities of nearest neighbors. Even the smattering of misclassified patches at left does not push the accuracy below 90%.

3.2 Segmentation Accuracy

Yet classification accuracy is only the first step of the PixelLaser pipeline. Starting from classifications such as those in Figure 3, a bottom-up search seeks the correct segmentation between traversable and untraversable terrain. Figure 4 shows three such results, where the green contour represents the boundary computed purely from search through patch-classifications; the red contour has "snapped" the green one to the strongest intensity edge within a few pixels. This latter correction works well in many environments.

Figure 5 shows two histograms of errors across 10 images. The left-hand histogram holds the pixel offsets between hand-segmented and automatically segmented images from test set B, representing the conditions under which we imagine the system most often deployed. The average absolute-value pixel error is 4.24 pixels per image column or "ray," with the median error far lower. Although [11] does not cite its errors in pixels, we believe that error measurements of pixels-per-ray *before* conversion to distance is the most useful metric for judging and comparing approaches involving image segmentation.

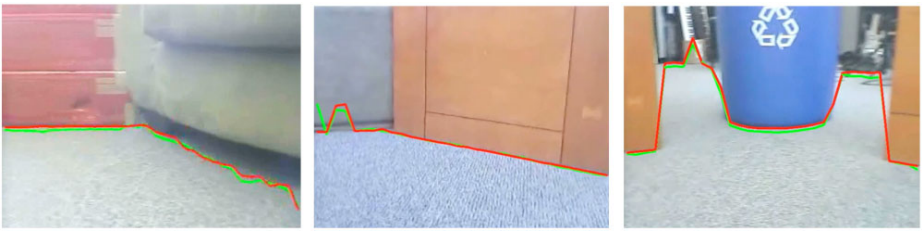

Fig. 4. Three examples of groundplane segmentations resulting from the classifications shown in Figure 3. Image patches at a horizontal resolution of 20 pixels provide the raw segmentation shown in green; the red line has "snapped" to nearby strong edges, when present.

3.3 Scan Accuracy

Pixels-per-ray, after all, avoids the critical dependence between range accuracy and the size of the environment being sensed. A pixel close to the bottom of Figure 2's images might represent only a millimeter of distance along the robot's groundplane. Near the horizon, however, that same one-pixel image displacement can represent an arbitrarily large metric displacement in the robot's environment.

To quantify these range errors, we transformed the pixel errors from left of Figure 5 to obtain the histogram of range errors that appears to its right. The average absolute error across test set B amounts to 91.2 cm. Note that this result is comparable to the approximately 100 cm average errors reported in [11]. The outliers account for almost all of this, as is indicated by the median range error of only 4.1 cm!

3.4 Applications to Spatial-Reasoning Tasks

Although we have designed this test set specifically to have images with long indoor distances to obstacles, ~20 feet, we note that this comparison is not truly

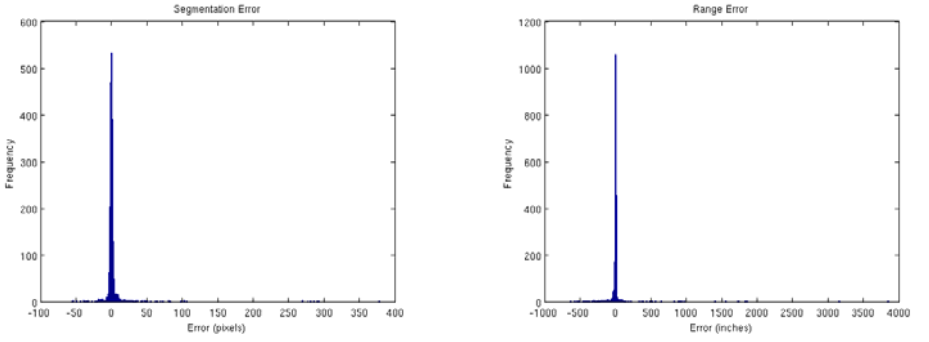

Fig. 5. At left is a histogram of the segmentation errors for 2107 column-segmentations across a group of 28 images from dataset B. The corresponding range errors appear at right. The average absolute errors are 4.24 pixels and 91.2 cm, respectively. Note that the outliers are quite heavy, especially in the range errors at right: there, the *median* range error is only 4.1 cm!

commensurate with [11]: in that work, the authors sacrifice pixels for field of view by using omnicam images. We believe that many applications can benefit from the improved accuracy and simplicity of unmodified webcam images, and so we have also tested whether these scans, with their 60° field-of-view, still support localization, mapping, and navigation tasks as do the full-circle scans of [11] or the 180° scans of LRFs.

To test localization, we mapped a small "playpen" and seeded it with a particle filter of 100 randomly selected poses. Using the Monte Carlo Localization algorithm from [16], Figure 6 shows the initial snapshot in the localization and three more instants as the particle filter converges around the correct pose after 20 scans are considered. Here, the motion model introduces considerable error to reflect the uncertainty in the iRobot Create's odometry.

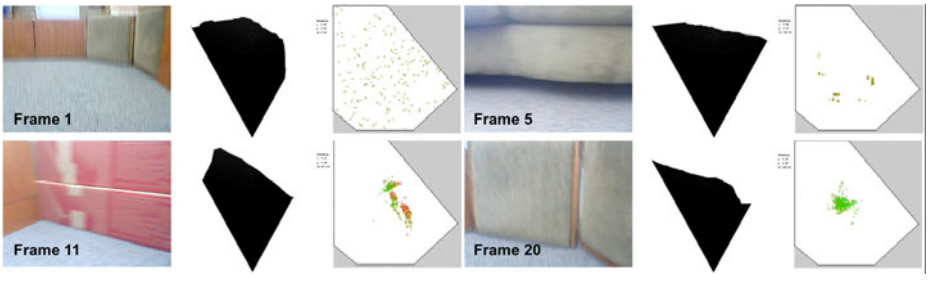

Fig. 6. Four frames from a localization run using MCL within a known map. The ambiguities apparent after 5 frames are handled appropriately by the particle filter: several clusters face different possible walls. Particles' colors indicate the likelihood of matching the current scan: bright green are the likeliest; red are the least likely. The filter converges to a single, correct cluster by the final frame. Better noise models would produce even tighter convergence.

To test these scans' ability to support off-the-shelf mapping algorithms designed for use with laser scans, we used CoreSLAM [13], obtained from *OpenSLAM.org*. Three playpen images and their scans appear in Figure 7, along with the resulting map produced by CoreSLAM. As the authors point out, CoreSLAM uses only a single best-guess estimate of each scan's pose. This leads to the slippage seen in the map: there are two separate recycling bins mapped in the bird's-eye view although only one was present in the environment. Even so, we are heartened that this map is of the same qualitative accuracy as those published by CoreSLAM's authors.

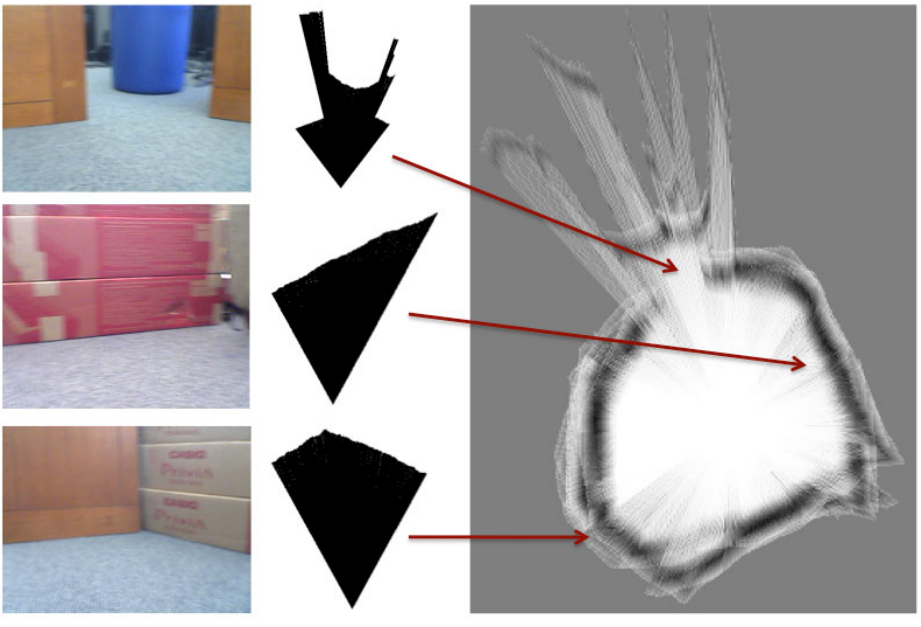

Fig. 7. At left are three images from a partially enclosed area. The robot circled the space twice, collecting images every second. Afterwards, the scans (middle) were created and integrated into the single map shown at right using the off-the-shelf CoreSLAM algorithm

Finally, to test whether these *PixelLaser* scans can support autonomous navigation, we trained a classifier on the completely different environment of the Westin Atlanta Hotel during 2010's AAAI conference. Figure 8 shows three of the training images and several snapshots from its extended autonomous run. Guided by no sensors other than the camera running the PixelLaser pipeline, the Create wandered for twenty minutes around the conference's exhibition hall. The only obstacles the system could not handle through that time were the very thin legs of the easels and chairs. It turned out that the segmentation's horizontal resolution sometimes missed them entirely.

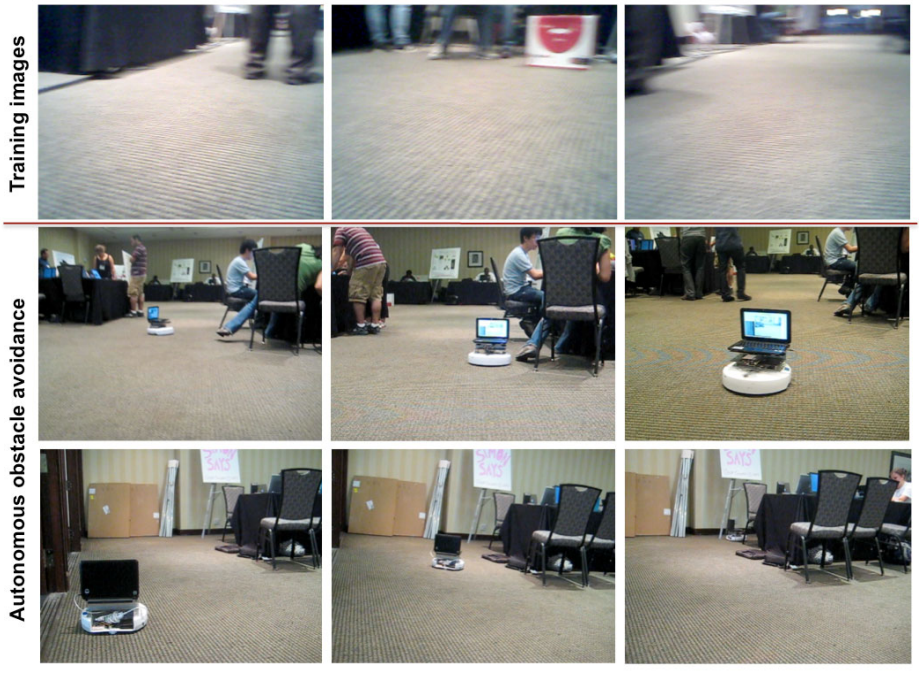

Fig. 8. Training images from a completely distinct environment (the downtown Atlanta Westin during AAAI 2010) appear at top; below are snapshots from an extended autonomous run in which the robot used no sensing other than the PixelLaser scans to avoid obstacles

4 Verdict and Perspective

Despite not having the sensors to take advantage of the past decade's advances in spatial reasoning, commodity robots have become a part of modern life. This work's examples of localization, mapping, and navigation are proofs-of-concept – certainly they offer broad opportunities for possible improvements. Yet even as we refine these applications, their common foundation – that of extracting range scans from image segmentations – has proven to be an accurate, flexible, and *inexpensive* approach for supporting reasoning about a robot's local surroundings. We look forward to the next generation of commercial platforms that, at no greater cost, will add such spatial reasoning to their repertoire of capabilities.

References

1. Blas, R., Agrawal, M., Sundaresan, A., Konolige, K.: Fast color/texture segmentation for outdoor robots. In: Proceedings, IEEE IROS, Nice, France, pp. 4078–4085 (September 2008)
2. Buhmann, J., Burgard, W., Cremers, A.B., Fox, D., Hofmann, T., Schneider, F., Strikos, J., Thrun, S.: The Mobile Robot Rhino. AI Magazine 16(2), 31–38 (Summer 1995)

3. Fletcher, F., Teller, S., Olson, E., Moore, D., Kuwata, Y., How, J., Leonard, J., Miller, I., Campbell, M., Huttenlocher, D., Nathan, A., Kline, F.R.: The MIT - Cornell Collision and Why it Happened. Journal of Field Robotics 25(10), 775–807 (2008)
4. Hoiem, D., Efros, A.A., Hebert, M.: Recovering Surface Layout from an Image. International Journal of Computer Vision 75(1), 151–172 (2007)
5. Horswill, I.: Analysis of Adaptation and Environment. Artificial Intelligence 73, 1–30 (1995)
6. Kanade, T., Kanade, B.Y., Morris, D.D.: Factorization methods for structure from motion. Phil. Trans. of the Royal Society of London, Series A 356, 1153–1173 (2001)
7. Laws, K.: Rapid texture identification. In: Proceedings, SPIE. Image Processing for Missile Guidance, vol. 238, pp. 376–380 (1980)
8. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The Office Marathon: Robust Navigation in an Indoor Office Environment. In: IEEE ICRA 2010, pp. 300–307 (2010)
9. Muja, M., Lowe, D.G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: Proceedings, VISAPP 2009 (2009)
10. OpenCV's, homepage `http://opencv.willowgarage.com/wiki/` (accessed 07/17/2010)
11. Plagemann, C., Enres, F., Hess, J., Stachniss, C., Burgard, W.: Monocular Range Sensing: A non-parametric learning approach. In: IEEE ICRA 2008, pp. 929–934. IEEE Press, Los Alamitos (2008)
12. Saxena, A., Chung, S.H., Ng, A.: 3-D Depth Reconstruction from a Single Still Image. International Journal of Computer Vision 76(1), 53–69 (2008)
13. Steux, B., El Hamzaoui, O.: CoreSLAM: a SLAM Algorithm in less than 200 lines of C code. In: Submission ICARCV 2010 (2010), `http://www.openslam.org/coreslam.html`
14. Taylor, T., Geva, S., Boles, W.W.: Monocular Vision as Range Sensor. In: Proceedings, CIMCA, Gold Coast, Australia, July 12-14, pp. 566–575 (2004)
15. Thrun, S., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., Schulz, D.: MINERVA: A second-generation museum tour-guide robot. In: Proceedings, IEEE ICRA 1999, pp. 1999–2005. IEEE Press, Los Alamitos (1999)
16. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
17. Urmson, C., Baker, C., Dolan, J., Rybski, P., Salesky, B., Whittaker, W.L., Ferguson, D., Darms, M.: Autonomous Driving in Traffic: Boss and the Urban Challenge. AI Magazine 30(2), 17–29 (2009)