PixelLaser: Evaluating monocular range from texture

Michael Leece, Nicole Lesperance, Steve Matsumoto, Max Korbel, Kenny Lei, and Zachary Dodds
Computer Science Department; CS REU in Systems
Harvey Mudd College
Claremont, CA
Contact: dodds@cs.hmc.edu

Abstract — This work presents and evaluates the PixelLaser system, designed to estimate range-to-obstacle scans from single images. Its visual pipeline uses nearest-neighbor texture-matching to segment groundplane (traversable) texture from non-groundplane (obstacle) texture. Using the known pose of a webcamera, the system then transforms these segmentations into range-scans. This paper presents a thorough evaluation of the range accuracy obtained in the resulting scans. In addition, it presents three practical robot applications that use PixelLaser's monocular ranging: (a) a safe-wandering agent, (b) a Monte-Carlo localizer, and (c) a mapping system using CoreSLAM. In each case, algorithms that usually depend on laser scans have succeed with PixelLaser data instead.

Keywords: range estimation, image segmentation, visual control

I. Introduction

The problem of finding range-to-obstacles for autonomous robots is a fundamental one – and one with a beautiful solution: the laser-range finder (LRF). LRFs provide a planar 180° scan of the environment at a resolution of 1° or 0.5°. Yet LRFs are not perfect: they are power-hungry, they are not eye-safe, and perhaps most importantly, they are very expensive: even a middle-of-the-road LRF costs $10,000. Webcameras cost orders of magnitude less, even than cheap LRFs.

With a known height and angle above the ground, one technique for computing range from single images is to determine the contour that divides the groundplane and obstacles: that segmentation is equivalent to the range scan provided by a LRF. This paper pursues exactly this path, proposing and evaluating an algorithmic pipeline that transforms single images into range scans of the sort LRFs provide. This system, named *PixelLaser*, has been implemented and deployed in support of three typical robot tasks: localization, mapping, and safe-wandering. The results demonstrate that for many applications, PixelLaser provides range scans of a quality sufficient to replace a laser – and at a small fraction of the cost.

A. Related work and this work's contributions

Range from single-image-segmentation has seen many robot applications: Horswill's Polly [1] pioneered robotic range-from-texture, and many systems have followed. In [2] Hoiem et al. show confidence levels in terrain navigability; Saxena et al. drive an RC car safely and quickly through rough, natural terrain [3]. Similar to the fast color/texture segmentation work of [4,5], all of these projects emphasized machine-learning contributions and task-specific image interpretation. This work, in contrast, focuses on the *accuracy* of the resulting range-to-obstacle scans. Thus, this work is closest to that of Plagemann et al. [6] whose omnicam images yielded ~1m depth accuracy, sufficient to support off-the-shelf SLAM algorithms under assumptions common to human-scale indoor environments.

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. We demonstrate range accuracy comparable to or better than that of [6], along with a simple training procedure and execution speed that can support real-time obstacle avoidance.

II. Approach

Figure 1 summarizes the algorithmic pipeline taken by the PixelLaser system. To begin, each image is considered in terms of 20x20 pixel subimages. Those small windows are distilled into six components: specifically, their red and green means and variances and two texture statistics, the variance and kurtosis of Laws's sixth texture filter [7]. We will call each of these six-component feature vectors a *patch*.

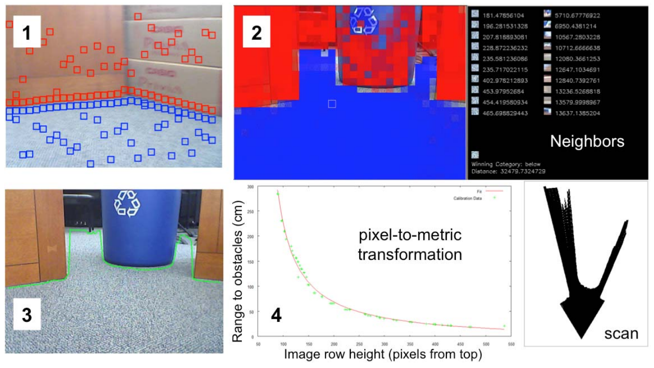
1
2
Neighbors
3
4
pixel-to-metric transformation
Range to obstacles (cm)
Image row height (pixels from top)
scan

Figure 1. PixelLaser's algorthmic pipeline: (1) training on hand-segmented images builds kd-tree models of obstacle and groundplane textures. (2) A new image is similarly decomposed into squares, each of which is classified. Here, blue indicates a groundplane match; red, an obstacle match. (3) We evaluated three algorithms for computing segmentations from classified patches. (4) With known camera pose, the system converts pixel heights to a range-scan.

In the learning phase, a human correctly segments on the order of 30-60 images taken from our robot wandering the environment of interest. From these correct segmentations, two kd-trees of patches are built: one of only groundplane patches and the other from the environment's obstacles. These two kd-trees represent the learned models of groundplane and obstacle textures, respectively.

With those models in place, the PixelLaser system operates in four-step sequence: (1) the system acquires a new image and computes the patch values from it, (2) using the learned models, each new patch is classified with a single value, the likelihood of its being groundplane or obstacle, (3) a segmentation is computed from the spatial context of those likelihoods, (4) that segmentation is transformed from pixel coordinates into spatial coordinates in the robot's canonical local frame of reference. This fourth step yields a range scan similar to the output of a LRF. We detail steps 2, 3, and 4 next.

A. *Classifying patches using nearest-neighbors (2)*

We hewed closely to [8] in creating our patch-classification system. In order to determine whether a new patch *p* is obstacle or groundplane, the N nearest neighbors to *p* (in a six-dimensional Euclidean sense) are found from each of the groundplane-model and obstacle-model kd-trees. Initial testing was run with our own data structures; our final system leveraged the excellent, free FLANN library [9].

The ratio of the distances between *p* and its two sets of nearest-neighbors yields a likelihood score along a continuum in which a value of 1.0 represents equal likelihood of being ground or obstacle, values greater than one are proportionally more likely to be groundplane, and values less than one are more likely to be obstacle. Figure 2 shows example images and their patch-classification likelihoods from two of our three different environments. In addition, the results section shows that this classification step is quite accurate overall.

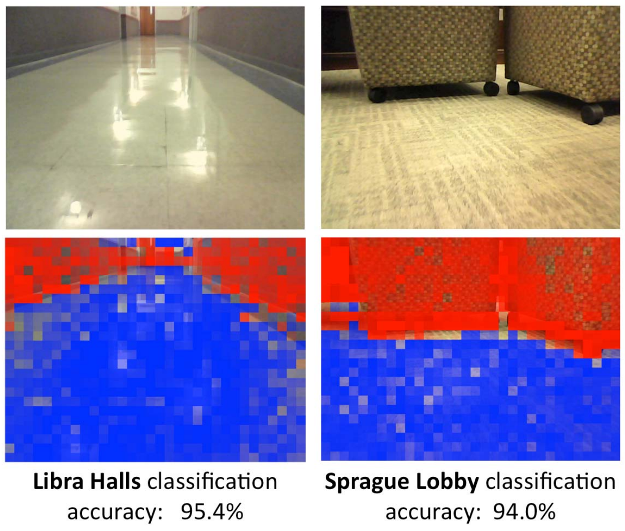
Libra Halls classification accuracy: 95.4%
Sprague Lobby classification accuracy: 94.0%

Figure 2. Non-training images from the "Libra Hallways" and the "Sprague Lobby" environments appear at top. Below are classified 20x20 subwindows: blue saturation represents groundplane likelihood and red, obstacle likelihood.

This work was made possible by the National Science Foundation, awards #0753306 and #0939149, and funds provided by Harvey Mudd College.

B. *Computing a segmentation from classified patches (3)*

The system's nearest-neighbor classification provides, for each image patch, a likelihood that it belongs to the groundplane or the robot's obstacles. The next step in PixelLaser's pipeline transforms an image of these likelihoods into a single segmentation contour separating traversable and untraversable portions of each image.

We experimented with several segmentation approaches. The naive approach, in which image columns are searched bottom-up until finding an above-threshold jump in obstacle likelihood ran slowly and succeeded only in very constrained environments. Uncommon carpet textures and varying lighting forced us to consider more sophisticated techniques. Another approach, based on seam-carving, used neighboring pixel columns to search in a roughly horizontal path through the image, but it imposed too much lateral constraint and failed in open areas.

The results section summarizes the relative segmentation performance of three of the best algorithms we pursued, all of which improved substantially on the naive and seam-carving approaches:

(1) Multi-resolution segmentation, our baseline approach, sped up segmentation by performing a binary search for groundplane/obstacle transitions within each image column.

(2) Our "Smooth" segmentation approach calculated classification certainties for all of the patches in an image and then applied an averaging filter to suppress unwanted misclassifications. After this smoothing step, segmentation proceeded as before.

(3) "Transition-line" segmentation used a third kd-tree for image-patch classification. This "transition" class modeled the texture of image patches that contain the transition from groundplane to obstacle. This third classification can be learned with little additional overhead from the available training data. After finding transitions as above, the potential transition patches are then compared against the transition patches learned from the training data. The one best matching that transition model is taken to be the correct segmentation.

All three of these techniques used a "snap-to-edge" post-segmentation correction. This step pushes the location of the boundary to the nearest and strongest image edge within a ten-pixel window around the previous segmentation estimate.

C. *Converting from pixel to spatial range-scans (4)*

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. We used 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 3 shows that the empirically-derived transformation from pixel to spatial coordinates fit the expected *inverse-depth* model well. We also note that in our camera's 640-column images spanning a 60° field of view, the angular resolution of 0.1° is about an order of magnitude better than even top-flight LRFs offer. Typically, however, we decimate the columns for speed.

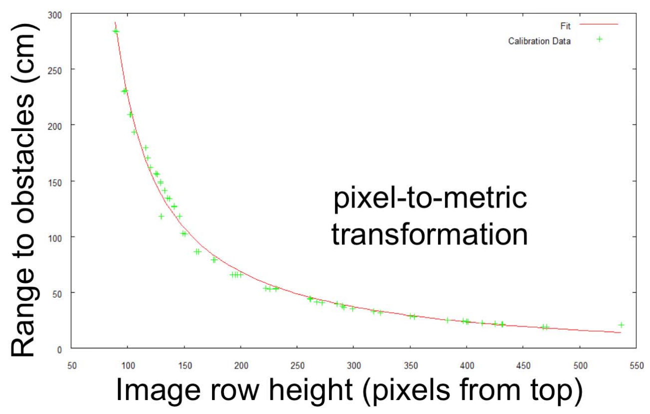

Figure 3. The red curve shows the transformation we used to map the image height of the segmentation into distance-to-obstacles from the camera. It represents the best *1/z* fit to the empirically-gathered data, shown in green.

III. Evaluation

We assessed each stage of the PixelLaser pipeline in order to determine where the bottlenecks in performance lay. Here, we offer the results and reflections on those evaluations.

A. *Evaluation of patch-classification*

Figure 4 demonstrates that the nearest-neighbors-based approach to texture classification (via our *patch* features) works quite well. When a strict threshold of 1.0 is used, on the order of 90-95% of patches are classified correctly. This value is not sensitive to the four different environments tried or the number of nearest neighbors identified from the underlying models of groundplane and obstacle patches. The sample images are taken from the "Sprague Lab" environment; the leftmost image shows one view of iRobot Create-based platform within it.

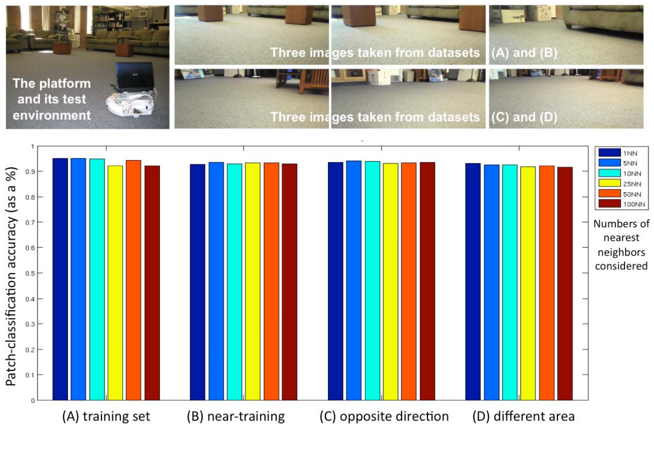

Figure 4. The images at top show the Create-based platform in our "Sprague Lab" environment, along with six images taken from it. Below are the accuracy results for our nearest-neighbors-based classifier, in which 90-95% accuracy is achieved both under near-training and far-from-training (but same overall environment) conditions. In addition, the *number* of nearest neighbors used for the comparisons (color coded) did not make a significant difference.

B. *Evaluation of segmentations and range scans*

Yet the system's nearest-neighbor classification provides, for each image patch, only a likelihood that it belongs to the groundplane. How well does the segmentation step identify the boundary *between* groundplane and obstacles? In order to evaluate this, we hand-segmented a set of images from the robot's testing run (and *not* used as part of the training set). Figure 5 illustrates some example images and the summary of the average pixel errors across the dataset. In addition, the post-transformation errors are listed, in inches. Note that these are arithmetic means: the median error was invariably much closer to 0! The fundamental reason why these errors are so large is that, near the horizon, a single-pixel can represent arbitrarily large distances from the camera. Small regions of mis-segmentation, then, cause the overall average error to be much larger than the typical error.

Even so, the results show that the use of the transition-line segmentation yielded the greatest success across five different environments: The first three only consider data taken from a stationary platform, e.g., a "move-then-think" deliberative approach. In the second set, the data is taken (and analyzed) as the robot moves at 20cm/s. As noted, the specular hallways of the "Libra" dataset posed such a challenge for moving robots that the additional transition classification for image-patches did not help. We suspect that the visual variance in such environments is too great to support a simple texture-based model for the intersection of the floor and wall planes.

Overall, however, these data compare favorably with those reported in [6], in which a ~1m (or ~39 in) average environmental error was reported for indoor scans.

Average pixel error per column (in pixels)

STATIONARY	Sprague Lab	Sprague Lobby	Libra Halls
Multires.	8.86	11.13	14.10
Smooth	3.62	7.99	10.34
Transition Line	2.79	6.08	14.01

STATIONARY	Sprague Lab	Sprague Lobby	Libra Halls
Multires	9.67	16.24	19.75
Smooth	7.44	5.10	22.11
Transition Line	4.12	4.52	14.27

Average distance error per column (in inches)

MOVING	Sprague Lab	Sprague Lobby	Libra Halls
Multires.	27.34	35.10	55.34
Smooth	17.64	33.01	59.18
Transition Line	19.68	36.45	75.28

MOVING	Sprague Lab	Sprague Lobby	Libra Halls
Multires	12.28	16.05	64.52
Smooth	11.08	9.81	30.15
Transition Line	11.01	8.72	22.78

Figure 5. These data summarize the accuracy results, averaged over datasets of 20-30 images with 50-100 image columns considered per image. Of the three segmentation algorithms tried, the "transition-line" approach, which used an additional classification tree in order to model image patches in which the transition from groundplane to obstacles appeared. The accuracy results when the robot was stationary demonstrate better range estimation than in [6]; when moving, they continue to be comparable. It turns out that the key source of error is not the segmentation itself, but wobble of the webcamera, which was the one built-in to the netbook's screen. When the screen bobs up or down, the horizon line shifts and the resulting distances accumulate a large error *all the way across the image*. We believe that a better-stabilized camera would yield errors such as those in the "stationary" category, even in motion.

IV. Applications

PixelLaser's goal is that its resulting range scans can serve as drop-in replacements for laser scans. Here we highlight three deployed applications, ordinarily based on laser data, in which we successfully used PixelLaser scans instead: Monte Carlo Localization [10], mapping via CoreSLAM [11], and safe navigation in the presence of obstacles.

A. *Monte Carlo Localization*

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 [10], Figure 6 shows the initial snapshot in the localization and three subsequent 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. Our sensor model, on the other hand, needed relatively small room for error because of the scans' fidelity.

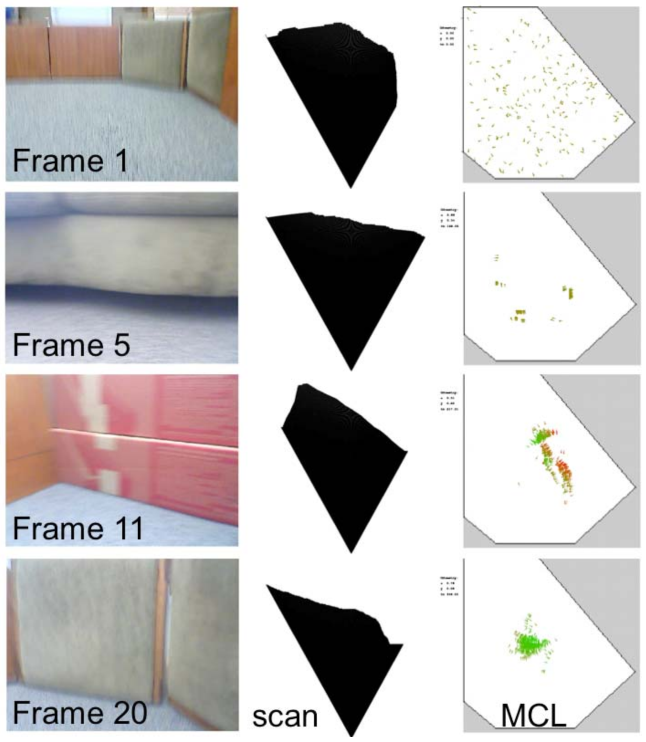

Figure 6. Four snapshots from a Monte-Carlo localization routine run using the scans from the PixelLaser pipeline. These come from an enclosed, mapped "playpen" that was part of the Sprague Lab environment. In each row the left image represents the robot's current view (showing the effects of motion blur), the middle image represents the resulting scan, and the right-hand image shows the population of pose hypotheses, i.e., MCL's particle filter, as updated after the Bayesian update based on that scan. Typically, MCL is run on robot vehicles using laser-based range scans.

B. *Mapping: CoreSLAM*

To test these scans' ability to support off-the-shelf mapping algorithms designed for use with laser scans, we used CoreSLAM [11]. Three playpen images and their scans appear in Figure 7, along with the resulting map that CoreSLAM produced. As CoreSLAM's 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 laser-based maps published in [11].

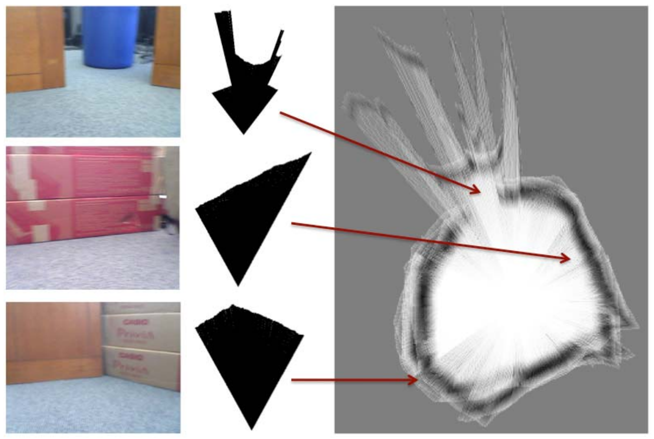

Figure 7. Similar to the MCL example, we used PixelLaser's scans as a drop-in replacement to the CoreSLAM algorithm for mapping environments. CoreSLAM requires odometric accuracy greater than the Create's in order to resolve all of the ambiguities, but its result (at right) demonstrates that PixelLaser's scans can offer a workable alternative to laser-based sensing.

C. *Safe navigation and obstacle-avoidance*

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 in that span were the very thin legs of the easels and chairs, and even that problem was a deficit of the navigation routines, not the sensing.

V. 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. PixelLaser'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.

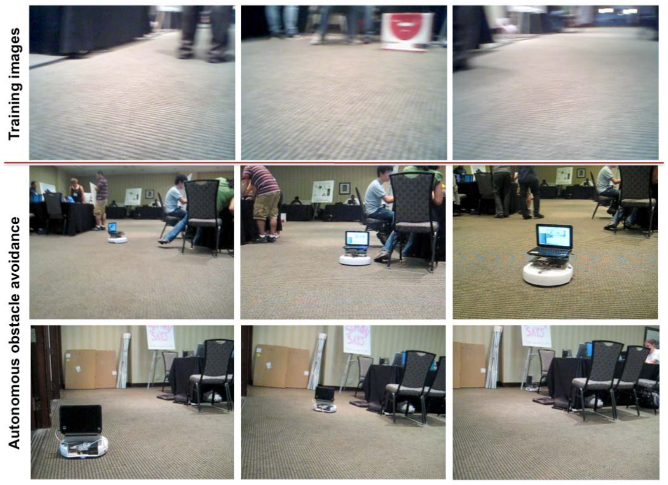

Figure 8. A summary of PixelLaser's support for autonomous safe-wandering. At AAAI 2010 in Atlanta, the system was quickly re-trained on the exhibition hall's images, three of which appear in the top row, above. Once trained, a simple algorithm stopped the robot when it came too close to an obstacle, and then chose a turning direction based on which side of its webcamera's field of view had more available freespace. That camera, along with the PixelLaser processing, was the only sensor used. Ultimately, the Create became stuck in an easel's legs not from a perceptual failure but because the navigation could not find a path through the many thin obstacles.

Acknowledgments

The authors acknowledge and thank the National Science Foundation (REU #0753306 and CPATH #0939149) and Harvey Mudd College for their support and facilities.

References

[1] Horswill, I. Analysis of adaptation and environment. *Artificial Intelligence*, 73 (1995) 1-30.

[2] Hoiem, D., Efros, A. A., and Hebert, M. Recovering surface layout from an image. *International Journal of Computer Vision*, Vol. 75, No. 1 (Oct. 2007) 151-172.

[3] Saxena, A., Chung, S. H., and Ng, A. 3-D depth reconstruction from a single still image *International Journal of Computer Vision* Vol. 76, No. 1 (2008) 53-69.

[4] Blas, R., Agrawal, M., Sundaresan, A., and Konolige, K. Fast color/texture segmentation for outdoor robots. Proceedings, *IEEE IROS*. Nice, France (September 2008) 4078-4085.

[5] Taylor, T., Geva, S., Boles, W. W. Monocular vision as range sensor. Proceedings, CIMCA 2004. July 12-14, Gold Coast, Australia. (2004) 566-575.

[6] Plagemann, C., Enres, F., Hess, J., Stachniss, C., and Burgard, W. Monocular range sensing: a non-parametric learning approach. IEEE ICRA'08. IEEE Press (2008) 929-934.

[7] Laws, K. Rapid texture identification. Proceedings, *SPIE Vol. 238 Image Processing for Missile Guidance*. (1980) 376-380.

[8] Boiman, O., Shechtman, E., and Irani, M. In defense of nearest-neighbor based image classification. Proceedings, IEEE CVPR. Anchorage, AK (June 2008). IEEE Press (2008) 1-8.

[9] Muja, M. and Lowe, D. G.: Fast approximate nearest neighbors with automatic algorithm configuration. Proceedings, VISAPP'09 (2009).

[10] Thrun, S., Burgard, W., Fox, D., *Probabilistic Robotics*. MIT Press (2005) Cambridge, MA.

[11] Steux, B., El Hamzaoui, O. CoreSLAM: a SLAM algorithm in less than 200 lines of C code (submitted to ICARCV 2010). Available at *www.openslam.org/coreslam.html*.