Autonomous Sign Reading for Semantic Mapping

Carl Case* Bipin Suresh* Adam Coates Andrew Y. Ng

Abstract— **We consider the problem of automatically collecting semantic labels during robotic mapping by extending the mapping system to include text detection and recognition modules. In particular, we describe a system by which a SLAM-generated map of an office environment can be annotated with text labels such as room numbers and the names of office occupants. These labels are acquired automatically from signs posted on walls throughout a building. Deploying such a system using current text recognition systems, however, is difficult since even state-of-the-art systems have difficulty reading text from non-document images. Despite these difficulties we present a series of additions to the typical mapping pipeline that nevertheless allow us to create highly usable results. In fact, we show how our text detection and recognition system, combined with several other ingredients, allows us to generate an annotated map that enables our robot to recognize named locations specified by a user in 84% of cases.**

I. Introduction

Autonomous mapping and navigation is an essential prerequisite for successful service robots. In contexts such as homes and offices, places are often identified by text on signs posted throughout the environment, such as room numbers or the names of people that work in a particular office or cubicle. Unfortunately, robotic mapping systems do not presently capture this information and thus are unable to take instructions from users that refer to these places by their names (e.g., "Room 120" or "John Smith's office"). In this paper, we propose a system that allows a robotic platform to discover these types of names automatically by detecting and reading textual information in signs located around a building. We show that the text-extraction component developed is a valuable addition to the usual mapping pipeline commonly used on mobile robots. In particular, our completed map allows the robot to identify named locations throughout the building with high reliability, allowing it to satisfy requests from a user that refer to these places by name.

The problem of mapping and navigating within an unknown environment has been studied thoroughly in the robotics literature. Robots can now routinely build metric maps of a new environment [1], and navigate from one location to another reliably [2], [3]. Prior work has shown that higher level knowledge can be extracted directly from these highly accurate (usually grid-based) maps, such as the locations of doors and corridors [4], [5] in order to build "semantic" representations of the environment. However, much critical information about a location is included in signs and placards posted on walls, yet existing mapping systems do not fully utilize this source of semantic data. For instance, signs near offices usually include room numbers, lists of office occupants, or the name of a room or hall. Thus, in this work we propose a system that incorporates a text extraction module into the usual mapping pipeline to capture this information and annotate the robot's map, enabling the robot to recognize locations by the names posted near them.

*Denotes equal contributions. The authors are with the Department of Computer Science, Stanford University. {cbcase,bipins,acoates,ang}@cs.stanford.edu

Fig. 1. Reading and Mapping Signs

Our system uses several steps to accomplish the mapping task: (i) it automatically explores a building to collect imagery of all of the walls, (ii) detects and reads characters from these images belonging to signs and placards, and (iii) attaches these annotations to the robot's map. In the first step, we use the map generated by a standard mapping system to traverse the entire building and capture images of walls where signs are likely to be located. We then use a logistic-regression classifier to detect likely text regions, and use an optical character recognition (OCR) system to extract text strings. The strings are then attached to the map based on the location where the text was found.

The most challenging sub-problem in this system is the text detection and character recognition itself (a problem that has warranted much prior work [6], [7], [8], [9], [10]). Unfortunately, while some off-the-shelf components are available, their performance is generally low when applied to non-document images (such as those acquired from our robot). Though initially this would seem to limit their use in robotics, where we need high accuracy for any useful system, we have found that various simple ingredients combined can result in a very usable system. This paper will present a complete end-to-end application including numerous such ingredients that make it possible to deliver high accuracy to the user despite the inherent difficulties in non-document OCR applications. In our final system, the robot can respond successfully in 95 out of 113 possible user queries, far beyond the raw performance of the OCR systems themselves.

We begin by discussing related work, followed by describing the setup for our application and each of the components of our system. Section III gives an overview of the application, while in Section IV-A we describe the process of collecting, and in Sections IV-B through IV-D we describe our text detection and recognition algorithms. Our experimental results are presented in Section V. We assess the performance of our system by presenting results for each algorithmic component in isolation, and then measure the performance of the full pipeline in a simple application carried out in both familiar and novel environments. We demonstrate high accuracy in our associations between labels and map locations.

II. Related Work

There has been a good deal of recent research focused on the question of "semantic mapping." One thread of such work has focused on creating linkages between spatial and geometric information (e.g., laser scans) and object class labels (such as "door" or "wall") so that automated reasoning methods may be applied to those entities. The authors of [11], for instance, use laser scans to detect planar surfaces in the world, then classify those planes as walls, tables, ceilings, etc. using a set of simple constraints. In [12] the authors extend this work by creating hierarchical spatial and topological maps then establishing links between the two.

Other researchers have used various tools from machine learning to identify properties of spaces and recognize higher-level layouts. For example, [13] shows how a mobile robot can learn to classify regions of a 2D map as navigable/non-navigable or as sidewalk/street. Friedman et al. [14] present an algorithm based on conditional random fields and Voronoi diagrams to automatically extract the topological structure of a metric map and label every cell as room, hallway, door, or junction.

A handful of researchers have considered applications of text reading in robotic navigation. For instance, the authors of [15] present an edge-based text detection algorithm applicable to office or home environments. (They suggest it may be useful for robotic navigation but do not demonstrate such an application which would have required the navigation, mapping, and text-recognition components that are integrated into our system.) In [16], an algorithm is presented to detect a set of known (pre-specified) landmarks where some include text such as room numbers, and the authors of [17] provide a model of their environment's hallways and doors so that the robot can search for a given room by first matching the door model and then using it to locate the target text. In contrast to these last two, we provide no prior knowledge of what rooms are present nor any model of the environment beyond very basic assumptions about the general locations of signs. Finally, the authors of [18] describe a robot that can read text from signs of known font, size, and background in a controlled lab environment (with no mapping component). We assume nothing about the text and depend on machine learning to handle novel or unusual environments.

Text detection and optical character recognition are also key components of our application and have warranted a body of research themselves. Early research has used texture segmentation with heuristics to detect text in non-document settings [19]. Other work has applied machine learning as in [20] where a boosted classifier is trained to detect text in street images. [6] shows that generic object detection techniques (in this case, deformable parts models) are also applicable to "word spotting" — finding words from a fixed lexicon in a natural scene. In our application we use a supervised learning method related to that in [20], though other methods might also be applicable.

III. Overview of Application

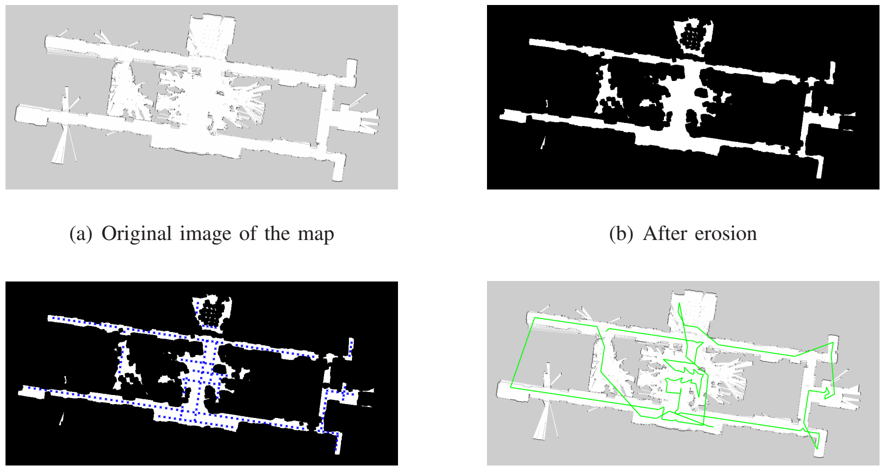
In the remainder of the paper we will develop a text-aware mapping system that fulfills the needs of a simple application: enabling a user to direct the robot's navigation by text queries (naturally, containing names and room numbers referring to places within a building). For instance, when the user types "john smith", the robot will locate the point on the map associated with this name (i.e., the nameplate outside his office.) Thus, our system will need to construct a map then annotate it with text data, including room numbers and any names associated with the rooms and offices.

We make two simplifying assumptions. First, we trust that the environment follows ADA guidelines with respect to room signage[1]. Most notably, signs must be posted five feet above the ground (modulo the size of the sign itself), thus limiting the areas that our robot must search for valid text data. Second, we separate the usual SLAM map-making task from the semantic mapping one; that is, we presuppose the existence of a navigable map for the target environment. This second assumption is made largely for simplicity: since our text modules do not use prior knowledge about the environment, they could reasonably be run simultaneously with typical SLAM systems during map building.

Several components comprise our end-to-end application. The first stage of our system plans a path through the environment's free space that follows the walls.[2] Along this path, the robot stops every few feet to capture photos using a high-resolution camera (often enough to adequately survey the entire length of each wall). The second stage uses a trained text detector to classify image regions as text or non-text, followed by an optical character recognition stage that attempts to read characters from these regions. Finally, using depth data available from the robot, the text outputs

[1]`http://www.access-board.gov/adaag/html/adaag.htm`

[2]This is merely for exploration purposes and does not preclude using the other components within a SLAM system before the map is completed.


(a) Original image of the map (b) After erosion

(c) Way points (d) Final plan

Fig. 2. Path planning

from the recognition module are collated and recorded on the map at their precise location, along with a normal vector representing the direction that the sign is facing (usually outward from a wall). The final map and the attached text data can then be saved for later queries by a user.

We now describe each of these components in detail and evaluate their performance individually. We then move on to demonstrate the performance of the full application, evaluating its accuracy in associating user queries with their corresponding locations on the map.

IV. Methods

A. *Map building, path planning and navigation*

Our system is built on top of the ROS software framework [21]. For our experiments, we manually drive the robot around the office, then build a world map using the GMapping SLAM toolkit [1], as is standard practice. The resulting map is then used to plan a path for surveying the building and collecting images (Figure 2(a)).

The path planning module aims to determine a set of "way points" in the world for the robot to visit as it collects images of the walls. The path must satisfy three goals: (i) maximize coverage of the building, (ii) ensure that the robot is not driven too close to the walls, and (iii) visits the way-points in a reasonably efficient manner.

Given a grid-based map produced by the SLAM toolkit, our system performs the following steps:

1) Binarize the map using a fixed threshold, yielding a binary map of free space.
2) Erode the result by a fixed amount to eliminate positions that are too close to the walls. The added distance ensures that each image covers a reasonable area of the wall, while also making the path safer to follow. (See Figure 2(b).)
3) Apply a Hough Transform [22] to find long straight lines in the eroded binary image (Figure 2(d)). These correspond to sections of wall long enough to incorporate signs, culling out cluttered areas not likely to contain useful information.
4) Create a set of "way points" equally spaced along these lines. These are the locations where the robot will stop to take pictures of the walls. At each point, the robot camera will be oriented orthogonally to the line (and hence orthogonally to nearby the wall).
5) Plan and traverse a path that visits all of the way-points, ordered greedily by distance, to capture images and depth images of the walls. This path is traversed in both directions so that, when necessary, images of the walls on each side of a hallway can be captured.

The result of this procedure is a sequence of images of the walls, along with registered depth data for each. These images are the inputs to the next module, which will scan them for text fragments.

B. *Text detection and recognition*

1) Detection: The output from the mapping module includes a set of images that need to be scanned for text. A significant body of work focuses on detecting text in natural scenes and video-frames. For comprehensive surveys of text-detection, see [8], [23]. In this work, we use a logistic regression classifier that uses a variety of text features, similar to the approaches described in [24] and [20]. For each image, our system computes a set of features known to be associated with the presence of text. The features are computed in 10x10 pixel windows (each 3 pixels apart) across the entire image, yielding a vector of features for each 10x10 patch. These feature vectors are then used as input to the classifier.

We use several text features from the text detection literature: local variance, local edge density, and horizontal and

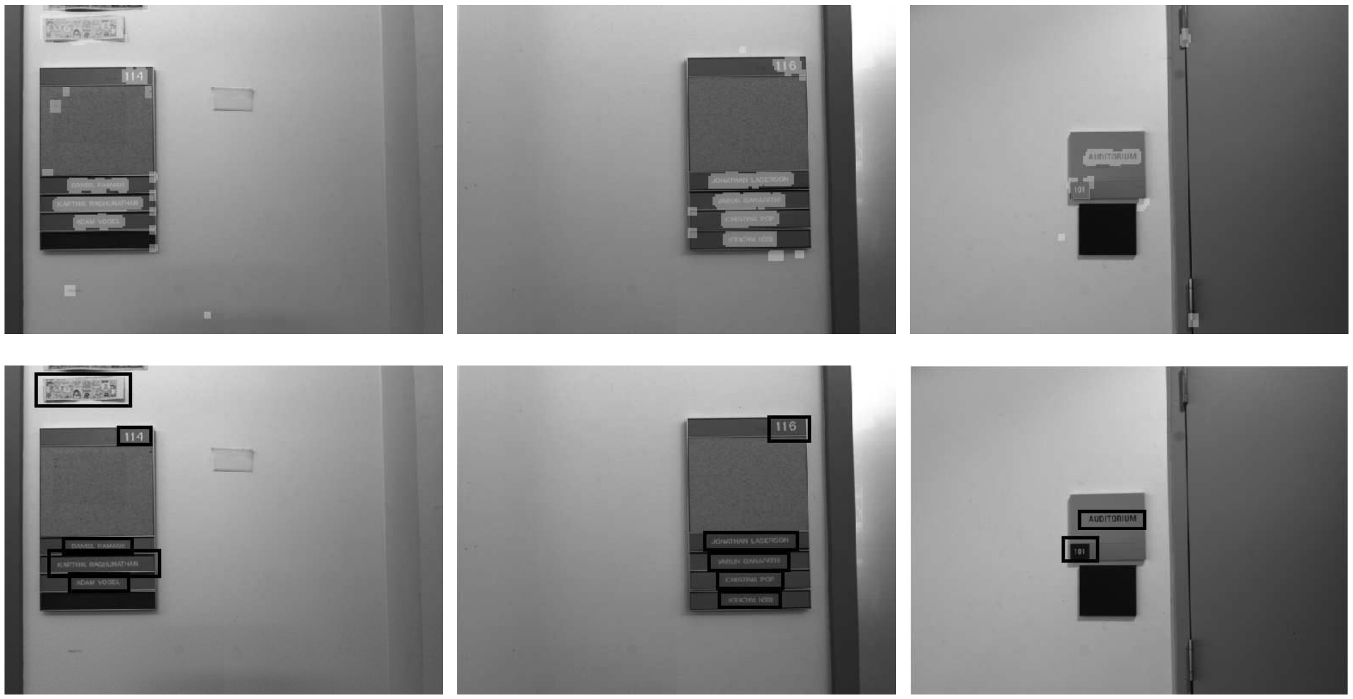


Fig. 3. Text Detection: The first row shows the patches predicted by the logistic regression classifier to contain text. The second row shows the bounding boxes that are submitted to the text-recognition phase.

vertical edge strength.[3] The features provided to our classifier are the max, min, and average value of these features within the 10x10 window being classified. Thus, for each window we have a total of $4*3=12$ features.

Given this set of features and a hand-labeled training set, we train a logistic regression classifier to distinguish between text and non-text image patches. For each 10x10 window, this classifier outputs the probability that the region contains text. Running this classifier across an image and thresholding yields a binary image where the positive-valued pixels represent regions identified as likely to contain text. These pixels are then grouped together into connected components. We reject regions whose areas are less than a predefined threshold ($\frac{1}{2000} * ImageArea$ in our experiments); and regions whose heights are greater than their widths. Finally, we construct tight bounding boxes around the remaining regions. These bounding boxes contain candidate chunks of the image that will be analyzed by the character-recognition module. Figure 3 shows the set of 10x10 patches identified as containing text for an example image.

We tested our text-detection module on images taken by the robot from two different buildings. The images show variations in lighting conditions, font style, background (light-text/dark-background vs. dark-text/light-background), font size, and relative location of text within the image. For training, we hand annotated 23 images, which resulted in 5000 image patches containing text (portions of door-numbers or occupant-names) and 150000 non-text example patches extracted from other parts of the same images. We then tested our classification system against an unseen set of 80000 (similarly labeled) image patches from 13 new images. Our train and test accuracies are listed in Table 1, and the Precision/Recall curve in Figure 4. (For the purposes of evaluating this classifier for our particular scenario, extraneous text in the images that are not parts of signs are counted as negative instances even though at this stage of the pipeline there is no way to distinguish between these classes.)

[3]If I is the image, S is a circular disk filter, and D is an edge filter like the Sobel filter, then local variance $V = S \bullet (I - S \bullet I)^2$; local edge strength $E = S \bullet |D \bullet I|$; and the horizontal and vertical edges are detected using Canny edge detection. We keep only edges shorter than a fixed threshold, as these tend to be associated with text.

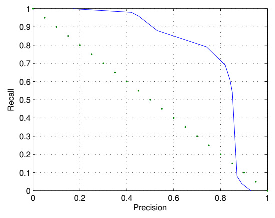


Fig. 4. Precision/Recall curve for text-detection.

TABLE I
PRECISION/RECALL FOR TEXT-DETECTION

	Train	Test
Precision	0.65	0.73
Recall	0.86	0.80

TABLE II

TEXT RECOGNITION ACCURACY

Edit Dist.	Nameplate Data	ICDAR '03
0	66%	59%
1	76%	66%
2	80%	72%

2) Recognition: The text detection module outputs a set of image regions believed to contain textual information. The next step is to extract text strings from these regions. Given the candidate image regions, our approach to reading the text follows that of [20]: we binarize the image and then pass its output to an off-the-shelf OCR engine. In our experiments we use the freely available Tesseract [25] engine.

Given a candidate image region containing text, it is usually possible to separate the text from the background using a simple constant threshold across the image. This is because signs intended for humans are generally monotone text on monotone backgrounds (as opposed to posters and stylized logos). Due to lighting conditions and variation in color choices, however, a fixed binarization threshold for all regions often yields poor results that confuse the OCR engine. Instead, we compute multiple candidate binarizations (nine in our experiments) and run the OCR engine on each. We then choose to keep the resulting text string whose confidence score (output by Tesseract) is highest.

We tested the performance of our text recognition module using two datasets. The first is a dataset of our own construction consisting of 50 images of hand-cropped text from a wide variety of nameplates — a sampling is shown in Figure 5. We additionally test the text recognition on the ICDAR 2003 Robust Word Recognition dataset with 896 of its images[4]. Note that the latter dataset contains script, handwritten, and other non-standard text styles which we make no explicit effort to handle. The results are shown in Table II. We also show how the accuracy changes as the number of errors (measured by edit distance) between the OCR result and the true string are allowed to increase. Thus, zero distance corresponds to the accuracy when requiring an exact match, and distance of one corresponds to the accuracy when allowing one "mistake", and so on. We will see that counting partial matches in this way is valuable in our final application.

C. Map annotation and post-processing

At the end of the recognition phase, we have most of the text data we desire: a set of text strings associated with bounding boxes in images captured from various locations on the map. The final step is to place these text strings accurately on the map by using the depth and position data acquired during the mapping run.

Since the robot includes a textured light projector and a stereo camera pair, each captured visual image includes depth

[4]We use the Trial Test Dataset without non-ASCII and very low-resolution (height < 25px) images (removing 214), available at `http://algoval.essex.ac.uk/icdar/Datasets.html`

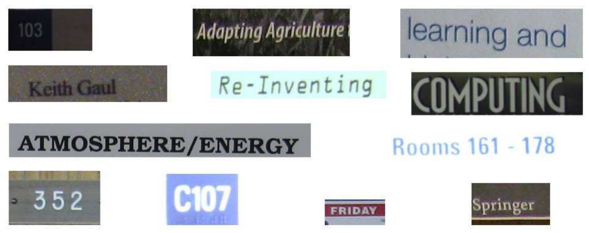


Fig. 5. Sample Text Patches

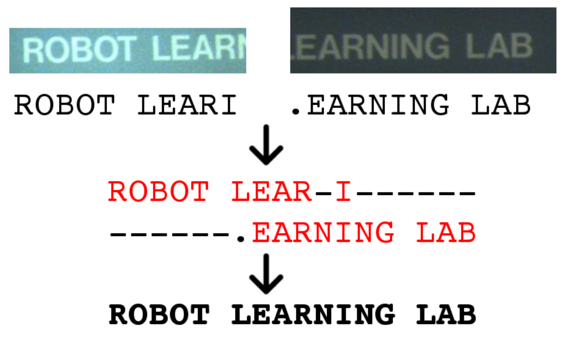


Fig. 6. Two images of the same piece of text, the OCR result of reading the detected text, and the final resulting string created by merging them using a sequence alignment algorithm. See text for details.

data precisely registered with the visual images. Thus, we can straight-forwardly take the center of each text region and find its 3D position relative to the camera (mounted on the robot's head). Using the robot's localization system this point can then be transformed into the map coordinate frame and stored. In addition to recording the location of the text, we also include a surface normal vector computed by fitting a plane to the depth data in a small neighborhood around the text region.

This step is easy to carry out when each string of text is fully contained in exactly one image. However, text inevitably appears in multiple images and can lead to errors in two ways. First and more commonly, a label will be present in two images and will be recognized as the same string both times. Second, it is possible for a name to be split at the border of two sequential images. In this case, we end up with partial overlap between the strings — for example, we may detect "John Sn" and "in Smith". Thus, as a final post-processing stage, we combine these redundant bits of text by matching up parts of strings that occupy the same 3D location.

More specifically, for any string s_i we define its neighbors $\{n_i^{(j)}\}$ to be all detections within a radius r on the map (with $r = 50cm$ in our experiments). We then search for an optimal semi-global alignment of s_i with each $n_i^{(j)}$. This is done using the Needleman-Wunsch sequence alignment algorithm [26], but with no penalties for gaps at the beginnings or ends of the strings.[5] If the alignment score of the best match is above a threshold (in practice, 4 works well), we discard the two partial strings and replace s_i with the optimal alignment of the two strings: We insert/delete characters as suggested by the alignment, and replace mismatched characters with whichever character is furthest from the end of its original string (as these characters tend to be more reliable). This process can be repeated on the remaining strings until no more merges are possible. The result is a reduced set of strings where duplicates and partial strings have been combined to yield an improved set of detections. See Figure 6 for an example of two strings merged together using this process.

D. User Queries

In our final application a robot operator can enter a name or room number and the robot navigates there without further instruction. During this process we must map the user's free-text query ("john smith") to a localized detection. It is possible to require an exact string match, but this requirement is inconvenient for the user who should be able to specify, for example, only a first name as long as it is unique. Furthermore, such a requirement degrades performance, as more intelligent string matching allows the system to recover from minor OCR errors such as one mis-read character. (See Table II.)

Since the user might enter a substring of the target, a global edit distance metric is not appropriate. Instead, we compute a local sequence alignment score between the query string and every detected string using the Smith-Waterman algorithm [27]. The string with the greatest alignment score is used. Note that an exact match necessarily has the maximum possible alignment score, so performance cannot degrade over using exact matches only.

V. Experiments and Results

We demonstrate the performance of the nameplate mapping application on the STanford AI Robot (STAIR) platform. We show that it can successfully detect and read the nameplates present in a novel environment with high accuracy which provides excellent performance for an end user.

A. Hardware

Our robotic platform is the STAIR platform, which is based on the PR2 robot platform from Willow Garage [28]. A Prosilica GC2450C camera on the head takes the high-resolution (2448x2050) images for text detection and recognition. Also mounted on the head are a textured light projector and two wide-angle cameras (approximately 90° field of view) used to gather depth information.

[5]In our implementation, we use a score of 2 for matching characters, -1 for mismatched/inserted/deleted characters, and -2 for inserted/deleted characters at the ends of strings.

TABLE III
Sign Reading Results — Environment 1

	Room Numbers	Room Names
Env. Total	30	41
Total Detections	30	147
False Positives	5	58
False Negatives	5	2
String-level Accuracy	50%	71%
User-level Accuracy	77%	93%

TABLE IV
Sign Reading Results — Environment 2

	Room Numbers	Room Names
Env. Total	19	23
Total Detections	21	21
False Positives	4	2
False Negatives	2	6
String-level Accuracy	74%	47%
User-level Accuracy	89%	74%

B. Evaluation

To measure the performance of our application, we provide STAIR with a map of a novel environment from which no data was collected for classifier training. We run the entire system described above, the output of which is a set of strings (classified as room number or name), each associated with an (x,y) coordinate on the provided map.

We define two metrics of accuracy to measure performance. Let $\{s_i^*\}_{i=1}^n$ be the n ground-truth text strings (names or numbers) present on nameplates in the given environment and let $\{t_j\}_{j=1}^m$ be the m strings returned by our system. The first metric is string-level accuracy: for each s_i^*, a correct value is a t_j in the correct location[6] that matches s_i^* exactly (edit distance zero). The second metric is user-level accuracy: for each s_i^*, a correct value is a t_j in the correct location such that s_i^* has minimal distance to t_j of all $\{t_j\}$. This definition corresponds to the user's entering each s_i^* in sequence and measuring the robot's navigational accuracy.

To test our system in full we run the entire application in two different novel environments. For each of these environments we report separate results for room numbers and room names. Within these two categories we report the total number of detections, false positives and negatives, and the overall accuracy at both a string-level and user-level as defined above. For room numbers, any detection that is not text or not a room number counts as a false positive; for names, we count only non-text and detections that should have been room numbers. The accuracy numbers are a percentage of all the text in the environment — not only of detected text. The results for the two environments are in Tables III and IV.

We see that while the low-level measures of accuracy are relatively poor, we nevertheless associate a significant number of correct strings with their correct locations on the map. More importantly, from the point of view of a

[6]We define a correct placement as one within 50cm of the coordinates given on a hand-labeled map

user, queries are mapped to the correct locations in the great majority of cases in both buildings (since exact matches are not necessary to get the correct destination). A portion of the map from the first building showing several typical detections and their locations (as recorded by our system) are shown in Figure 7.

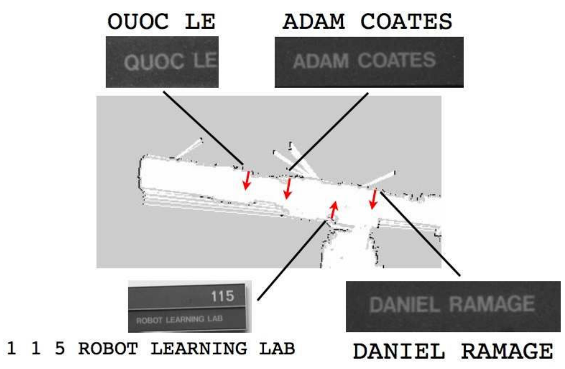


Fig. 7. Portion of map with typical annotations found by our system. The red arrows denote the position and normal vector of the detected signs. The image snippets show the detected text at each location, along with the OCR results.

VI. Conclusion

In this paper we have presented a system that automatically collects semantic labels associated with places on a map by reading them from signs posted on walls in the environment. The output of our system is a map annotated by dozens of accurate text labels suitable for use in applications. Despite the difficulties in detecting and reading text accurately in general, we have shown that our combination of text-handling components is already a valuable addition to the standard mapping pipeline, allowing users to specify locations by name with a reasonable degree of reliability in multiple buildings. That these results are achievable using off-the-shelf OCR software and virtually no assumptions about the language structure or spatial layout of the text is surprising and suggests that even better results may be obtainable in the future through improvements in the basic modules laid out in our work.

References

[1] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb 2007.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance." *IEEE Robotics and Automation*, vol. 4, no. 1, 1997.

[3] B. P. Gerkey and K. Konolige, "Planning and control in unstructured terrain," in *ICRA Workshop on Path Planning on Costmaps*, 2008.

[4] D. Anguelov, D. Koller, E. Parker, and S. Thrun, "Detecting and modeling doors with mobile robots," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 3777–3784.

[5] W. Shi and J. Samarabandu, "Investigating the performance of corridor and door detection algorithms in different environments," in *International Conference on Information and Automation*, 2006.

[6] K. Wang and S. Belongie, "Word spotting in the wild," in *European Conference on Computer Vision (ECCV)*, Heraklion, Crete, Sept. 2010. [Online]. Available: http://vision.ucsd.edu/ kai/grocr/

[7] B. Epshtein, E. Ofek, and Y.Wexler, "Detecting text in natural scenes with stroke width transform," in *Computer Vision and Pattern Recognition*, 2010.

[8] J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: a survey," *International Journal on Document Analysis and Recognition*, vol. 7, no. 2, pp. 84–104, July 2005.

[9] V. Vanhoucke and S. B. Gokturk, "Reading text in consumer digital photographs," in *SPIE*, 2007.

[10] T. Sato, T. Kanade, E. Hughes, and M. Smith, "Video OCR for digital news archives," in *IEEE Workshop on Content-Based Access of Image and Video Databases(CAIVD'98)*, January 1998, pp. 52 – 60.

[11] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915–926, 2008.

[12] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernandez-Madrigal, and J. Gonzalez, "Multi-hierarchical semantic maps for mobile robotics," in *International Conference on Intelligent Robots and Systems (IROS)*, Aug. 2005, pp. 2278–2283.

[13] D. Wolf and G. Sukhatme, "Semantic mapping using mobile robots," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 245–258, Apr. 2008.

[14] S. Friedman, H. Pasula, and D. Fox, "Voronoi random fields: extracting the topological structure of indoor environments via place labeling," in *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, 2007, pp. 2109–2114.

[15] X. Liu and J. Samarabandu, "An edge-based text region extraction algorithm for indoor mobile robot navigation," in *Mechatronics and Automation, 2005 IEEE International Conference*, vol. 2, Jul. 2005, pp. 701–706.

[16] M. Mata, J. M. Armingol, A. D. L. Escalera, and M. A. Salichs, "A visual landmark recognition system for topological navigation of mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[17] M. Tomono and S. Yuta, "Mobile robot navigation in indoor environments using object and character recognition," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2000, pp. 313–320.

[18] D. Létourneau, F. Michaud, and J.-M. Valin, "Autonomous mobile robot that can read," *EURASIP J. Appl. Signal Process.*, vol. 2004, pp. 2650–2662, January 2004.

[19] V. Wu, R. Manmatha, and E. M. Riseman, Sr., "Textfinder: An automatic system to detect and recognize text in images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 11, pp. 1224–1229, 1999.

[20] X. Chen and A. Yuille, "Detecting and reading text in natural scenes," in *Computer Vision and Pattern Recognition*, vol. 2, June 2004, pp. 366–373.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[22] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[23] K. Jung, K. I. Kim, and A. K. Jain, "Text information extraction in images and video: a survey," *Pattern Recognition*, vol. 37, no. 5, pp. 977–997, 2004.

[24] R. Lienhart, "Video ocr: A survey and practitioner's guide," *In Video Mining, Kluwer Academic Publisher*, pp. 155–184, 2003.

[25] R. Smith, "An overview of the tesseract OCR engine," in *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 629–633.

[26] S. Needleman and C. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[27] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[28] K. Wyrobek, E. Berger, H. der Loos, and J. Salisbury, "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 2165–2170.