A vision-based localization algorithm for an indoor navigation app

Oscar Deniz, Julio Paton, Jesus Salido
and Gloria Bueno
VISILAB, School of Industrial Engineering
Universidad de Castilla-La Mancha, Spain
Email: Oscar.Deniz@uclm.es

Janahan Ramanan
E-Twenty Development Inc
Toronto, Ontario
Canada
Email: j.ramanan@etwenty.co

***Abstract*—Current indoor navigation solutions face many problems in hospitals. Heavy electro-magnetic interference from hospital equipment affects the compass on a mobile device. Existing infrastructure may also need to be replaced or modified to provide better signal triangulation indoors. Current solutions are challenging in terms of cost (for both hospitals and users) and disruption of normal operation. In this context, this paper introduces the patent-pending SmartIndoor navigation and wayfinding system. With SmartIndoor, the user only has to point his smartphone to any sign inside the hospital. The recognized text is sent to a server which contains a database of sign texts along with their position inside the building. This paper describes the computer vision task of recognizing texts with a smartphone, arguably the most important module of the whole system. Preliminary results show that the proposed approach can be effective for such smartphone-based indoor navigation system.**

I. Introduction

In todays modern lifestyle we find individuals spending more time indoors. People are used to working in large offices, shopping in giant complexes, and attending classes at large institutional campuses. Yet there is no comprehensive solution to help users locate themselves indoors, in a manner comparable to GPS positioning outdoors. Imagining individuals inside malls, museums, airports, hospitals, institutions, stores, and office buildings there is always a need to know what their current position is in relation to their surroundings. With this capability individuals can determine what the points of interest such as washrooms, ATMs, vending machines, and telephones are located nearest to them on a map. They can also determine what route they should take to reach their destination. In response, larger facilities inevitably display large maps with the You are here icon to help orient users. These are only dispersed sparsely around the venue. The most common technological solution (see recent survey [1]) has been to incorporate external radio signals. This involves mapping the radio signals of the venue/facility and defining zones on the map each having a particular signal profile. The signal profile contains unique radio source tags (identifiers) and their measured average signal strengths. When a users mobile device polls the radio information and finds their data matches one of the zone profiles, the users position is pinned to the location coordinates of the zone. Common radio sources mapped are those of wifi access points, bluetooth beacons, and cell tower signals. Another solution uses unique codes around the facility such as QR codes which contain location information. Again, this requires installing these codes at various points in the facility, which also have to be maintained, and disrupt the look and feel of the environment and might not fit their interior design theme.

As an example scenario, poor wayfinding costs the average hospital $657 per bed annually. Therefore hospitals have long been in search of reliable solutions that can cope with many hallways in large complexes. Current indoor navigation solutions face many problems in hospitals. Heavy electro-magnetic interference from hospital equipment affects the compass on a mobile device. Existing infrastructure may also need to be replaced or modified to provide better signal triangulation indoors. This is challenging in terms of cost (for both hospitals and users) and disruption of normal operation.

This paper introduces the patent-pending SmartIndoor wayfinding system, which provides zero infrastructure wayfinding and real-time turn-by-turn navigation. The app, see Figure 1, works by wayfinding without requiring a compass or any external signals. The user simply sets his start location, sets his destination, and the indoor maps will guide him along the route. To estimate his current location, the user has to point his smartphone to any nearby sign. The recognized text would then be sent to a server that contains a database of texts and their positions within the building. This paper describes the system's core capability, i.e. recognizing texts with smartphones. The paper is organized as follows. Section II describes previous related work in mobile text recognition. Section III describes the system. Experiments are shown in Section IV. Finally, the main conclusions are outlined.

II. Related work

As mentioned above, SmartIndoor relies on the user pointing his smartphone to any nearby sign. The recognized text is sent to a server that contains a database of texts and their positions within the building. The user must receive a first estimate in no more than 2 seconds after taking the picture, so efficiency is key. Text recognition relies on Optical Character Recognition (OCR). OCR is one of the most successful computer vision applications to date [2]. Commercial systems that translate printed text images into computer-readable text

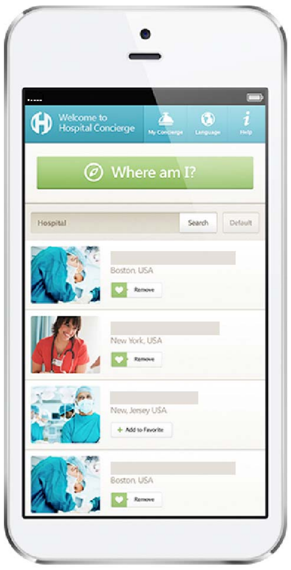


Fig. 1. Hospital Concierge App

have been available for years. More recently, text recognition has become an important task in a variety of applications that differ from the traditional typewriter printed text scenario. Some examples include: recognizing street numbers and signs in street-level imagery, plate recognition, recognition of text within images for CBIR, text reading for the blind and digital-born images, where texts appear in web pages in the form of images. Some authors have called the new scenarios “OCR in the wild” or robust reading, referring to the harder problem of recognizing texts with a variety of fonts, colours, perspectives, etc.

While scanners have been historically the first capture devices, nowadays there are multiple devices which can obtain text images. Ordinary people have into their hands a large number of digital cameras which allow users to capture text images from everywhere. This will be also made more evident with wearable devices. This method provides a desirable flexibility, however, it does not come without a cost. In fact, this flexibility presents new challenges that have to be addressed. Some of these new problems are low resolution, uneven lighting, perspective distortion, non-planar surfaces, complex backgrounds, zooming and focusing, moving objects and the need to use lightweight algorithms [3].

Despite these new difficulties, a few text recognition systems have been already designed for mobile devices. In [4], an Android-based system is described for recognizing the text in aligned nutrition fact tables (NFTs) present on many grocery product packages. The algorithm localizes aligned NFTs via vertical and horizontal projections, and segments the NFTs into single- or multi-line text chunks (OCR was not attempted in this case). In [5] a text sign recognition system for Android mobile devices is described. The system is to be used by visually impaired people and it includes audio feedback via a TTS module. No text localization is performed, text signs are assumed to span most of the image, which is directly used as input to the OCR engine. Surprisingly, the whole system was implemented on a Samsung Galaxy SII smartphone, including a training stage. No processing time results were given by the authors.

Traditional scientific fora for character recognition research has acknowledged this trend towards ”in the wild” reading and new events are being sponsored such as the Robust Reading competition within ICDAR (International Conference on Document Analysis and Recognition), held in 2011 and 2013 [6]. In such competitions, separate tasks are typically configured for text localization and text recognition (for obvious reasons, OCR engines cannot be directly fed with input images, text has to be first located and cropped). However, little effort has been made in the following aspects:

- Computational cost is not typically considered, despite the fact that devices typically used for ”in the wild”' reading have very limited resources. Besides, in such devices feedback to the user must be provided in relatively short times
- Traditional OCR engines are still used (ABBYY, for example, is the baseline OCR used in the ICDAR 2011 and 2013 Robust Reading evaluations), although it is not clear if they can cope so well with the new scenario
- While separating text localization from recognition is appropriate, the tasks are clearly related, for text localization is implicitly performing a text/non-text segmentation that could be leveraged by OCR engines

III. Proposed system

Figure 2 shows an overview of the SmartIndoor system. The map database in the server is filled from CAD drawings of the facility. This paper focuses on the App module which locates and recognizes the text within an input image taken with a smartphone. This module consists of three main parts: text localization, text binarization and text recognition.

For text localization, an algorithm based on [7] was tested. An implementation of that algorithm is, at the time of writing, due to be released as part of OpenCV 3.0. We also considered the Stroke Width Transform (SWT) for text detection, described in [8]. In preliminary experiments, it was observed that the method of [7] was too slow for a feasible mobile implementation. On a iPhone 4s smartphone the method needed approximately 7 seconds per frame (processing input frames at a 480x270 resolution). For these reasons it was soon discarded in favor of SWT.

In SWT, a stroke is defined as a contiguous part of an image that forms a band of a nearly constant width. This operator computes the per pixel width of the most likely stroke containing the pixel being processed. In other words, stroke width consistency is used to extract the pixels inside the body of strokes. After applying the SWT, an image of size equal to the size of the input image is obtained where each element contains the width of the stroke associated with the pixel. Moreover, no assumptions are made about the actual width

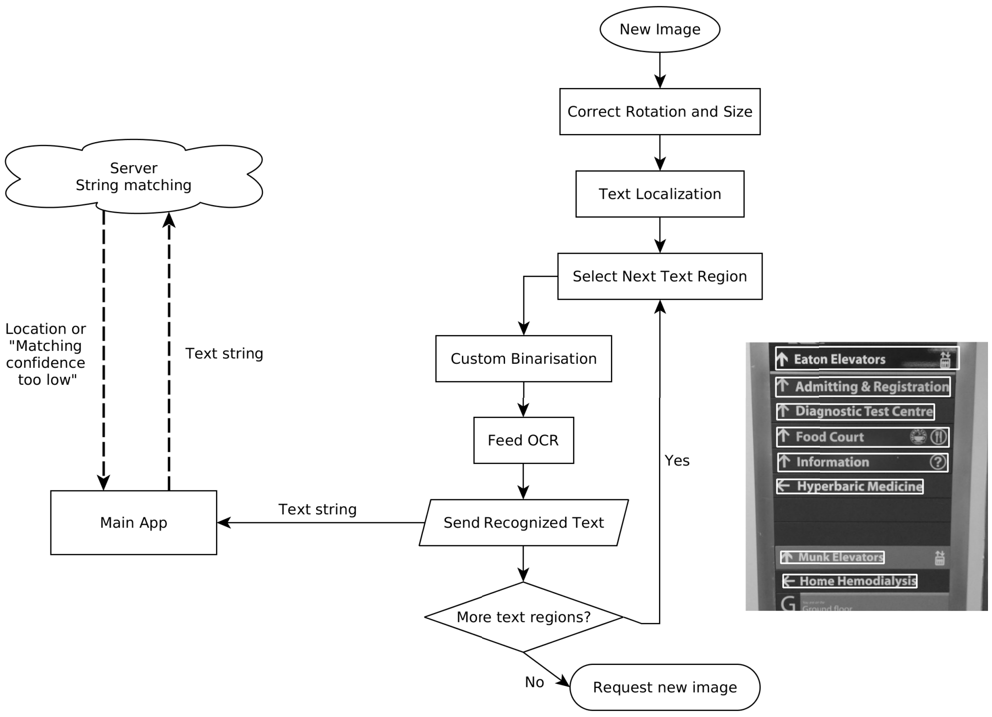


Fig. 2. SmartIndoor localization system

of the stroke. Before applying the SWT operator, the input image is converted to grayscale and edges are extracted using the Canny edge detector. Then the algorithm makes use of gradient directions in order to find all the pixels that are part of the stroke bounded between two edges.

The next step of the algorithm is to group pixels into letter candidates. To achieve this, the connected stroke pixels are grouped into connected components as candidate text characters. Then these candidates are filtered to separate areas which may contain text by using different properties such as the variance of the stroke width, text size and aspect ratio. Once characters are obtained, they are grouped into text lines paying attention to different similarities that text belonging to the same line usually have. For instance they are expected to have similar stroke width, letter width, height and spaces between the letters and words. The SWT method is based on simple properties, therefore it is very suitable for mobile devices. Moreover, it can detect text without restrictions regarding text font and language. It also has some limitations, the local stroke width is only well defined when individual characters are clearly visible.

The implementation of the algorithm used in this system extracts the regions which contain text from the rest of the image. These regions are rectangles that feed the OCR engine.

The behaviour of the algorithm can be modified by changing a number of available parameters. This can be useful to tune the algorithm for using it in different scenarios. For this, a brute force search of the best parameter sets for our dataset was performed. The search is able to produce the best parameter set by comparing the F-Measure obtained in different iterations. In each iteration, a combination of different parameter values is selected and all the images in the dataset are processed by the algorithm. The best parameter set will be the one with the highest F-Measure value.

Some of the parameters are related to the Canny edge detector and others to the maximum and minimum sizes and ratios. Furthermore, the system is prepared to use more than one parameter set, which can help to locate difficult texts. Since in our indoor localization application the user will be pointing the camera to a sign, if no text is found in the first run of the location text algorithm, the system enters into a loop. In each iteration, the image is processed using a different set of parameters until there are no more sets (the first parameter set was the optimal obtained with the utility mentioned above), a text is found or the loop reaches a time limit. This is necessary to keep the application responsive to the user. Specifically, the time limit was set to two seconds.

Text localization and recognition in natural images have conventionally been seen as autonomous tasks executed in a strictly sequential processing chain with little or no information sharing between tasks [9]. In this work we explore the possibility of leveraging the raw SWT images to perform a binarization of the image. SWT images represent text and non-text regions, so this can be leveraged to perform a binarization that could help the OCR. The goal is to obtain a better binarized image than the one computed internally by the OCR engine. The raw SWT images are used as binary masks helping us calculate the mean pixel intensity of text and background in the original grayscale image. Once both values are measured, the middle value is calculated and this value is used to binarize the image. This processing is not performed over the whole image but in each rectangle detected in the text localization step. These partial binary images are then sent to the OCR engine. The results using this binarization step are shown separately in the next section.

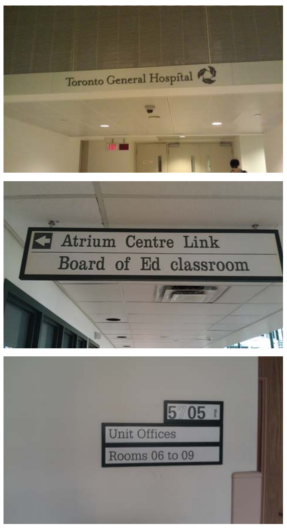
The last step in our STR system is text recognition. For this purpose an OCR engine is integrated in the system. It makes use of text regions extracted in the previous step. The previously located rectangles enable the system to achieve an important improvement: the most important text in the image is sent in the first place to the OCR engine. The rectangles are ordered by size, so the first text region sent out to the OCR will be the largest one. The importance of a text within an image is directly proportional to its size, this assumption fits most scenarios. This step correspond to the "Select Next Text Region" process of the system flowchart in Figure 2. On the other hand, this allows the system to perform text recognition asynchronously. It can send recognized words faster to the application as it does not have to wait until all text areas are processed. This is really important for our target mobile application as it can start its own processing using the most relevant texts, allowing for faster user feedback.

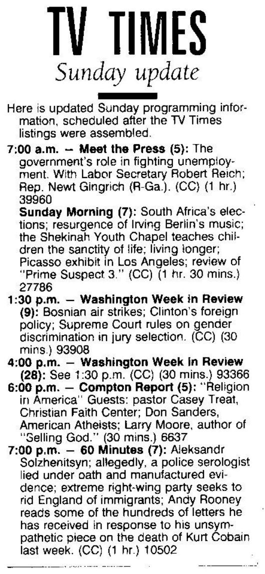
Two OCR engines have been used in order to compare accuracy and performance: Tesseract [10], considered the most accurate open-source OCR engine available (it also supports several platforms although Android and iPhone are not well tested platforms) and ABBYY FineReader, a powerful commercial OCR SDK that integrates ABBYY's state-of-the-art document recognition and conversion software technologies (it also has a mobile version which can be used in iOS and Android). A priori, no specific engine was deemed superior (although ABBY's was expected to perform better), and thus the two were considered in the experiments below.

Apart from the steps above, the input image is resized as this is critical for the execution speed and use in a mobile App. In particular, the image width was set to 800 pixels and the image height varies depending on the image aspect ratio. On the other hand, when the smartphone camera gets a new image, its orientation can be estimated using the smartphone sensors. After that, the image is rotated by the same angle, thus obtaining images with horizontal text strings.

IV. Experiments

This section compares and discusses the results obtained using the system described, implemented on an Android platform. Experiments have been carried out on two different datasets to test various scenarios. The first dataset contains images from a large hospital in Canada. They have been taken by a smartphone to reproduce the conditions in which the application will be used. As ground truth, a text file for each image was manually created containing all the words in the image. In total there are 92 images and 697 words, Figure 3-left shows some examples.




TV TIMES

Sunday update

Here is updated Sunday programming information, scheduled after the TV Times listings were assembled.

7:00 a.m. – Meet the Press (5): The government's role in fighting unemployment. With Labor Secretary Robert Reich; Rep. Newt Gingrich (R-Ga.). (CC) (1 hr.) 39960

Sunday Morning (7): South Africa's elections; resurgence of Irving Berlin's music; the Shekinah Youth Chapel teaches children the sanctity of life; living longer; Picasso exhibit in Los Angeles; review of "Prime Suspect 3." (CC) (1 hr. 30 mins.) 27786

1:30 p.m. – Washington Week in Review (9): Bosnian air strikes; Clinton's foreign policy; Supreme Court rules on gender discrimination in jury selection. (CC) (30 mins.) 93908

4:00 p.m. – Washington Week in Review (28): See 1:30 p.m. (CC) (30 mins.) 93366

6:00 p.m. – Compton Report (5): "Religion in America" Guests: pastor Casey Treat, Christian Faith Center; Don Sanders, American Atheists; Larry Moore, author of "Selling God." (30 mins.) 6637

7:00 p.m. – 60 Minutes (7): Aleksandr Solzhenitsyn; allegedly, a police serologist lied under oath and manufactured evidence; extreme right-wing party seeks to rid England of immigrants; Andy Rooney reads some of the hundreds of letters he has received in response to his unsympathetic piece on the death of Kurt Cobain last week. (CC) (1 hr.) 10502

Fig. 3. Examples of the two datasets used

The second dataset includes images of printed text, the traditional scenario in which OCR engines have been used. They are contained in a collection used for research in OCR and Information Retrieval [11]. This collection includes both the images and the ground truth, with a total of 6703 words. Figure 3-right shows an example of this dataset.

We first show the time differences between recognizing the rectangle containing all text lines and recognizing the first rectangle, see Figure 4.

Table I shows the recognition results obtained with the first dataset. Each row represents the number of words and characters recognized with both OCR engines (Tesseract and ABBYY) using a different parameter set. Note that these results are calculated without the custom binarization step mentioned above.

The parameter set does not directly affect the behavior of the OCR engine, although it affects the text localization algorithm, which is essential to recognize the text afterwards. The sets were calculated utilizing particular subsets of the image dataset hence the influence of text localization over the OCR engines can be compared in several cases. Moreover, the last row shows the results when using the whole dataset to compute the parameter set whereas the first row correspond to the default parameter set of the algorithm.

Two values have been calculated: number of recognized words and mean Hamming overlap. The former represents the total number of words that the system is able to recognize using a specific parameter set. All word characters have to match in order to be counted. The mean Hamming value counts all the characters from the groundtruth word that are recognized in the same relative position, normalized to the length of the groundtruth word. For instance, for word 'SECTION', OCR outputs such as 'SOCTION' and 'SECTLON' correspond to

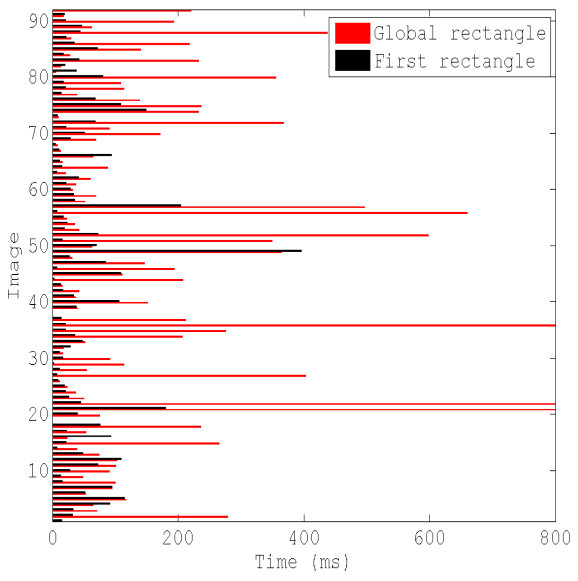


Fig. 4. Time differences between recognizing the rectangle containing all text lines (Global) and recognizing the first rectangle, using the first dataset.

a Hamming overlap of 85.7%, 'BECPLON' to 57.1% and 'NOITCES' to 0%.

	Tesseract		**ABBYY**	
Param. Set	W	H	W	H
1	52.1%	69.1%	40.2%	61.8%
2	4.6%	9.7%	8.5%	32.8%
3	9.9%	14.8%	13.9%	40.0%

TABLE I
RESULTS ON THE FIRST DATASET. W=RECOGNIZED WORDS, H=MEAN HAMMING OVERLAP

From the table it can be observed that Tesseract achieves better results than ABBYY. For instance, using the default parameter set, Tesseract is able to recognize 363 of 697 complete words (52.1%) and 69.1% of the total number of characters, however ABBYY recognizes 280 complete words (40.2%) and 61.8% of characters. As the goal of the system is to recognize as many words and characters as possible, Tesseract was selected as the best option.

Since this result in which Tesseract fares better was unexpected, we used the second dataset to validate our implementations. Using this dataset, the results are the opposite. As can be seen in Table II, ABBYY performs much better than Tesseract. The recognition rates ABBYY is able to reach are excellent, above 90% complete words. On the contrary, Tesseract offers even worse results than it did in the first dataset, recognizing only 60% of the words. On the other hand, it is important to mention that the absolute results should be better using the application on a smartphone. The reason is that it makes use of smartphone sensors to calculate the rotation at the moment of taking the picture. This increases OCR accuracy significantly.

	Tesseract		**ABBYY**	
Param. Set	W	H	W	H
1	32.6%	63.4%	91.6%	97.0%
2	32.0%	63.8%	92.5%	97.4%
3	32.8%	64.1%	89.0%	96.1%

TABLE II
RESULTS ON THE SECOND DATASET

In Table III the processing times obtained by executing the recognition system with the first dataset are shown. A Samsung Galaxy S II smartphone was used to run the test. Two architectures have been tested with Tesseract that are compatible with most devices nowadays. However, ABBYY was only available for one architecture (ARMv5). The results show that ABBYY is much faster than Tesseract, processing one image in just 1.2 seconds whereas Tesseract takes more than 2 seconds in the best case. However, taking into account the asynchronous processing proposed in this system, both systems will deliver the first word at roughly the same time: 0.95 seconds ABBYY and 1.1 seconds Tesseract.

	Tesseract		**ABBYY**
	ARMv5	**ARMv7**	**ARMv5**
Localization per image	978.55	765.48	977.9
OCR per image	1358.85	1439.62	294.16
OCR per word	336.96	356.99	72.95
Time per image	2303.65	2180.48	1237.99

TABLE III
PROCESSING TIMES USING THE TESSERACT AND ABBYY ENGINES (MS)

If our custom binarization step is performed, the results improve as Table IV shows. They correspond to the system using Tesseract on the first dataset. Taking into account the simple computations involved, the recognition improvements is advantageous.

Set	Rec. Words	Δ	Hamming	Δ
1	54.5%	+4.61%	70%	+1.30%
2	4.3%	-6.52%	9.1%	-6.19%
3	9.9%	0.00%	13.8%	-6.76%

TABLE IV
CONTRASTIVE RESULTS WITH THE CUSTOMIZED BINARIZATION STEP

Note that the absolute recognition rates are still very low. We also tested the string matching capability in order to provide more realistic recognition estimates for the whole system. The already-mentioned Hamming overlap was used to match the outputs of the system with the groundtruth of the first dataset. For each image, the system makes a decision based on the closest matching in the groundtruth texts. Algorithm 1 shows the exact procedure.

Table V shows the recognition results in this case. Note that they are significantly higher, reaching 75%. Still, many errors were caused by frequent words that appeared in a number of different signs. For example, with these two groundtruth text files: [DANGER, ROOM] and [DANGER, ROOM, X-Ray, Section]. When trying to recognize the second sign, there

```
1 for each image in the dataset do
2     Recognize text in the image for each groundtruth text file
      do
3         for each word in the file do
4             Compute Hamming overlap with the output of the
              text recognition (output of line 2)
5         end
6         Compute mean Hamming overlap for this groundtruth
          text file
7     end
8     Select the file the maximum mean Hamming overlap If that
      file coincides with the image, increase hits
9 end
```

Algorithm 1: Algorithm for computing the sign recognition rates.

was a mean Hamming overlap of 100% with the first one and slightly less than 100% with the second one (words DANGER and ROOM were perfectly recognized, but not the other two), which caused an error. To consider this, we modified the mean in line 7 of the algorithm... This modified mean version achieved a recognition rate of 81% (first parameter set).

	Param. Set					
	1	2	3	4	5	6
W	75.0%	23.9%	29.3%	52.2%	72.8%	71.7%

TABLE V
RECOGNITION USING STRING MATCHING (USING TESSERACT).

Also, the SmartIndoor system is designed so that, when matching confidence is low, the user is prompted to confirm or reject the matched sign (wide-field images of the area can be provided to better allow the user to make a selection). If the user chooses to reject, then he/she has to either take another picture or capture another nearby sign. In order to obtain estimated recognition rates with such reject option, we computed the detection-reject curves, see Figure 5. Again, the results show that with only a few rejections recognition rates higher than 90% can be achieved.

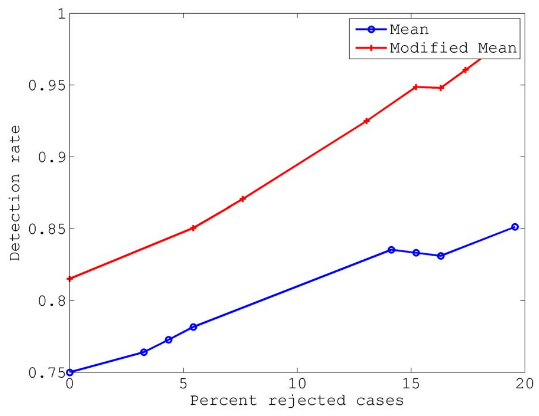


Fig. 5. Reject curve using the first parameter set.

V. Conclusions

This paper has introduced the SmartIndoor wayfinding system. An implementation for the core smartphone-based text recognition capability has been shown that is both robust and efficient. When combined with a reject option it allows the system to be of practical use. Overall, SmartIndoor is a cost effective indoor navigation system with zero infrastructure requirements. The SmartIndoor system, developed as part of the UTEST 2013 University of Toronto Incubator program, is currently in beta deployment status throughout a number of hospitals in Canada and USA. Beta tests with hospital clients will help provide the best experience for visitors, patients, and staff. Eventually, the system can be expanded in a number of ways. First, other properties and tags can also be tagged with location entries in the database; color (certain departments have color codes), images (certain zones in the facility have logos, symbols that are used as identifiers), environment (certain zones in the facility have unique designs on the walls or floors which can be used to distinguish them from other zones). Second, a secure web portal will be provided for hospital staff to update maps and other hospital data. This is synchronized with the app on users devices.

Acknowledgment

The authors are supported by project TIN2011-24367 from the Spanish Ministry for Economy and Competitiveness.

References

[1] N. Fallah, I. Apostolopoulos, K. Bekris, and E. Folmer, "Indoor human navigation systems: A survey," *Interacting with Computers*, vol. 25, no. 1, pp. 21–33, 2013.
[2] K. Jung, K. In Kim, and A. K. Jain, "Text information extraction in images and video: a survey," *Pattern Recognition*, vol. 37, no. 5, pp. 977–997, May 2004.
[3] J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: a survey," *Int. Journal of Document Analysis and Recognition (IJDAR)*, vol. 7, no. 2-3, pp. 84–104, Jul. 2005.
[4] V. Kulyukin, A. Kutiyanawala, T. Zaman, and S. Clyde, "Vision-Based Localization and Text Chunking of Nutrition Fact Tables on Android Smartphones," in *Int. Conf. on Image Processing, Computer Vision and Pattern Recognition*, 2013, pp. 314–320.
[5] O. Foong, S. Sulaiman, and K. Ling, "Text signage recognition in Android mobile devices," *Journal of Computer Science*, vol. 9, no. 12, pp. 1793–1802, 2013.
[6] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez i Bigorda, S. Robles Mestre, J. Mas, D. Fernandez Mota, J. Almazan Almazan, and L.-P. de las Heras, "ICDAR 2013 Robust Reading Competition," in *12th Int. Conf. on Document Analysis and Recognition (ICDAR)*, Aug 2013, pp. 1484–1493.
[7] L. Gomez and D. Karatzas, "Multi-script text extraction from natural scenes," in *12th Int. Conf. on Document Analysis and Recognition (ICDAR)*, Aug 2013, pp. 467–471.
[8] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform." in *CVPR*. IEEE, 2010, pp. 2963–2970.
[9] N. Roubtsova, R. Wijnhoven, and P. de With, "Integrated text detection and recognition in natural images," vol. 8295, 2012, pp. 829 507–829 507–21. [Online]. Available: http://dx.doi.org/10.1117/12.906761
[10] R. Smith, "An Overview of the Tesseract OCR Engine," in *Procs. of the Ninth Int. Conf. on Document Analysis and Recognition - Volume 02*, ser. ICDAR '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 629–633.
[11] K. Taghva, T. Nartker, J. Borsack, and A. Condit, "UNLV-ISRI Document Collection for Research in OCR and Information Retrieval," in *Proc. SPIE 2000 Intl. Symp. on Electronic Imaging Science and Technology*, 2000.