Towards robust and efficient text sign reading from a mobile phone

A. B. Cambra, A. C. Murillo
DIIS, Instituto de Investigación en Ingeniería de Aragón
University of Zaragoza, Spain
ana.cambralines@gmail.com, acm@unizar.es

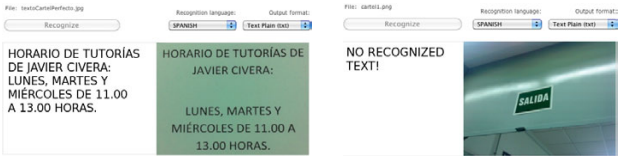
Abstract

Embedded applications on mobile phones are reaching impressive goals thanks to the current powerful smartphones. This work is focused on text recognition applications from mobile phone pictures. Optical Character Recognition (OCR) methods have been developed for a longtime, but they still have poor robustness to process text in general scene images. Our general goal is to study and improve their results, in particular when running locally on a phone. We present a realistic prototype running on iOS, with a light geometry based pre-processing step that helps detecting regions of interest in the image, i.e., likely to contain text-signs. Then, we show how to process and filter these hypothesis to facilitate text recognition by standard OCR methods. This initial version is aimed to rectangular shaped signs to easily take advantage of geometric cues. We demonstrate the performance improvements of including our proposal together with several available OCR libraries. All steps are run locally on the phone in the designed application, which can read or translate the text using additional standard services in the phone.

1. Introduction

There is a broad range of powerful *smartphones* on the market nowadays, and it is getting more and more common to use them for plenty of daily tasks besides standard calling and texting actions: maps, e-mail, agenda, games ... Their common characteristic is that they allow developing and installing new and third party applications that can make use of hardware components, such as GPS localization or integrated camera. As a general goal, this work is aimed to study how easy and feasible is to develop computer vision based applications in these mobile phones, within the particular field of text reading assistance applications.

There are plenty of Optical Character Recognition (OCR) systems, many available as open source libraries. However, they are not usually designed to process general scene images but scanned texts or images zoomed, cropped and centered in the text. We already find several recent commercial applications applying these techniques within mobile devices, making the phone able to "read" text from pictures but with some restrictions for the user. This work presents a proposal towards eliminating these restrictions and improving robustness of this kind of systems, what could increase their fields of application.

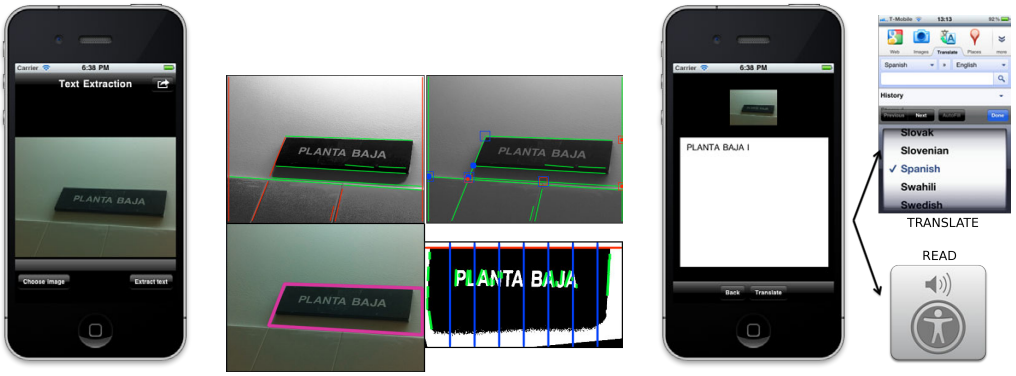


(a) Correct reading (b) Failed reading

Figure 1. (a) Successful recognition results obtained with an *OCR* in a frontal and zoomed view. However, more general scenes with distant text and perspective deformations (b) fail more often.

Typically, OCR methods present low recognition performance when the text in the image suffers perspective distortion or is not properly aligned, centered, zoomed or illuminated, as can be seen in the example in Fig. 1 [1]. Performance would increase if we could automatically obtain regions of interest containing text and process them to avoid those issues from general scene images. Considering the application should be run on a mobile phone brings important differences with regard to multiple related works, detailed next, on text recognition in unrestricted scenes. Simpler and efficient steps seem a requirement to allow the system to run locally on the phone. We start by restricting this initial work to rectangular text-signs, which allows us to make geometric assumptions that help obtaining an efficient prototype running on the mobile phone. Perspective deformation is corrected in plenty of computer vision applications through well known homography estimation and planar region projection, we present here how to apply these ideas in our particular problem. In order too run a homography based rectification, the basic step in this proposal is the gen-

[1] obtained with one of the many OCR available online: *OnlineOCR*, http://www.onlineocr.net/



(a) (b) (c) (d)

Figure 2. Summary of the proposed process. Acquisition (a), pre-processing steps to get a better view (b) that can be read by an standard OCR (c) and then used for different applications (d).

eration of rectangular region hypothesis in the image. This step is also aimed to automatically detect regions of interest, releasing the user from image acquisition restrictions (such as how the image should be acquired or where the text should be located in the image).

This work shows promising results running a simple approach and presents a fully automated text-sign reader prototype running on iOS. Besides presenting good usability and reasonable execution time, our proposal is shown to improve the performance of standard OCR libraries running on the phone. Several improvements and alternatives still have to be explored to generalize the approach to non-rectangular signs. Figure 2 shows a summary of the process developed. First the user acquires a new image (a), and then different steps based on geometric cues and simple texture analysis are run in a transparent way to the user (b). Finally we obtain the OCR analysis (c) only for the most likely hypothesis. This result is then said aloud or processed by an online dictionary (d). In the following sections, we describe and evaluate the details of our approach and the improvements in the performance when integrating it with different available OCR methods.

1.1. Related Work

There are plenty of successful OCR methods [10] proposed for a long time but we still find recent and novel contributions towards new fields of application, robustness or higher performance. Initially, OCR techniques were thought to process scanned texts from paper format to obtain a digital version of them. Nowadays, there are plenty of applications where it would be useful to recognize the text in general images. For example, in the particular case of reading assistance, we find plenty of proposals in the literature towards visually impaired assistance applications [3, 9, 12, 13]. However we still find few of these ideas integrated on mobile phones, partly due to high computational cost of some of the proposals. In this paper, we analyze which image processing techniques may be suitable for improving the results of this kind of applications running on a mobile phone. Thanks to the explosion of mobile application development we already find interesting computer vision based recent commercial products, such as wordlens[2] that reads texts in an augmented reality application.

As already mentioned, OCR methods typically require specific image acquisition (frontal view, zoom, centered ...) to have a good performance recognizing the text, presenting poor robustness to perspective distortions or more cluttered images, as seen in the initial example in Fig. 1. We can divide related research in two different groups. On one hand, there is a group of works towards better character recognition itself, proposing new OCR techniques. We even find works able to identify highly deformed text [18] as presented in web CAPTCHA systems. On the other hand, there are works that propose how to "pre-process" images of general scenarios to detect which image regions are likely to contain text and how to recognize it. Trying to recognize text in unrestricted images is an active field of research, with specific conferences and competitions dedicated to it. We find approaches proposing to work with image segments, dividing the image into connected components areas after applying different texture filters to the image [16]; or classification approaches at pixel-level, as proposed in [2], using histogram and gradient values on the region around each pixel. Certain works try to detect text along videos, taking advantage of the tracking and temporal information [15], [8]. Other interesting results towards detecting text areas in general scenarios include proposals based on different classification and learning techniques, e.g., using an AdaBoost based classifier [1] or training several SVM classifiers [6], to determine which regions are more likely to contain text. Some recent works propose to move from the use of typical OCR techniques to more general object recognition tech-

[2] http://questvisual.com/

niques adapted for text recognition [14, 11]. The work in [14] proposes, once a character recognition step has been run in the region of interest, to take into account the likelihood of full words for the final recognition results. Authors in [11] present a joint text localization and recognition approach that considers the characters as MSER regions and proposes to train the system with synthetic fonts.

Our work is related to the group of proposals trying to pre-process the images to recognize regions of interest, that can be later processed by standard OCR techniques. Two of the closest related works to our proposal can be find in [4, 17]. Both works use different orientation or perspective correction steps based on geometric constraings, but they do not integrate all steps required in the process nor demonstrate an application feasible in a phone.

2. Rectangular hypothesis

As described, this work is focused on reading text-signs and we can expect most of them being of rectangular shape. Therefore, we start the process with a search for possible rectangles in the image as regions of interest. Since there are no restrictions on the image to be processed, our approach detects straight segments and cross points (corners) between them all over the image. Out of these corners, possible rectangular hypothesis are generated and the likelihood of each of them containing text is evaluated.

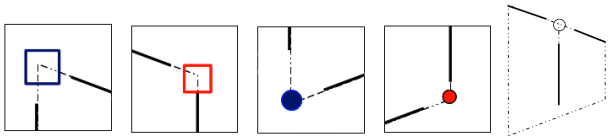
2.1. Straight segments and corner detection

We have implemented a C version[3] of the straight segment detection method proposed in [7]. The obtained segment set is filtered by size, we reject segments below 50 pixels, and by proximity/overlap, we merge segments whose end tips are too close or present big overlap. Then, if we divide the resulting segment set into vertical and others, we can search for cross points between each possible pair taking a segment from each group. To be sure the cross points selected are likely to be part of a rectangular object, we only accept cross points that fall into the image limits and whose distance in the image to the segment tips is below an stablished threshold. According to the relative position of the components of each cross point, we find four different types of corners, as shown in Fig. 3. Part (e) shows an example of particular cases where line segments cross each other in the middle part of one of them, therefore several corners are stored (one per possible type) at that location.

2.2. Rectangular hypothesis generation

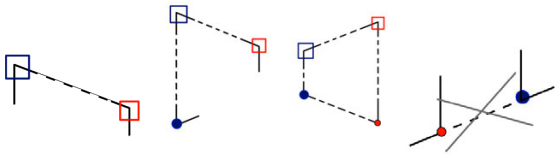
Next step seeks rectangular hypothesis built from the corners found in previous step. First, these corners are grouped with each other trying to get sets made of as many compatible corners as possible. We take into account first a suitable alignment of the line segments generating each corner point; secondly, a distance between the corners, which must be higher than minimum segment length; and finally a correct relative location of the different types of corners (Fig. 4 shows examples fitting or not this criteria).

[3] Available at http://webdiis.unizar.es/~anacris/code/lineDetector.zip



(a) (b) (c) (d) (e)

Figure 3. Types of corners detected: (a) Top-left, (b) Top-right, (c) Bottom-left, (d) Bottom right. (e) Multiple-type corner: TL+TR.



(a) (b) (c) (d)

Figure 4. (a)-(c) Different sets of compatible corners with 2, 3 and 4 corners respectively, and (d) a pair of non compatible corners, due to wrong relative position.

Corners that have not been included in any compatible group are used to instantiate singleton hypothesis. Therefore we have compatible sets made out of one, two, three or four image corners, to allow the system to be robust against occlusions and line segments missed during the extraction process. All of these sets, except the four-corner ones, need a post-processing to estimate the complete shape of the rectangular region. For each of these matched sets, we estimate how the whole region could be. We take into account the segments that composed each corner to generate the missing sides of the rectangular regions. For same cases, as shown in Fig. 5, more than one hypothesis can be generated from a compatible set of corners.

2.3. Rectangular hypothesis evaluation

Besides taking into account the fact that some rectangular region sides and corners may not have been detected, we should deal with the fact that there can be many rectangular shapes in images that do not correspond to text signs. The final OCR step is the most expensive one (the whole processing time dedicated to one hypothesis is split aproximately in 30% of the time of the prefiltering and 70% for the OCR step), therefore we want to avoid overloading the system running it for all hypothesis. We evaluate the likelihood of each of them containing text to allow the system keep working only with a few top (most likely) candidates.

As basic hypothesis descriptor we compute the gray level histogram, with 16 levels. From prior knowledge we could expect to find two clear local maxima in this histogram, cor-

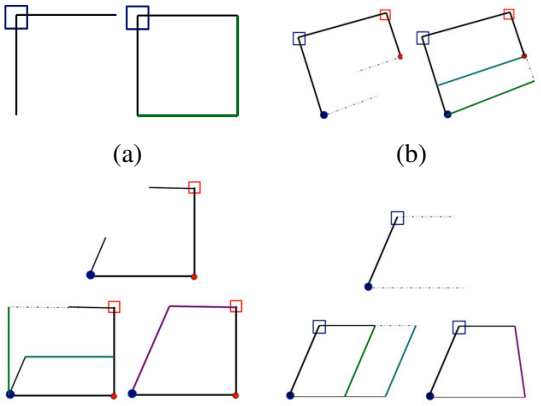


Figure 5. Rectangular hypothesis generated from single corners (a) and from compatible sets of four (b), three (c) and two (d) corners.

responding to the color of the text and background. Moreover, this double-maximum would make that the average and the mode are not too close to each other and that the distribution gets high standard deviation (compared to deviation on homogeneous textures hypothesis). These ideas where confirmed analyzing the values on a reduced set of rectangular hypothesis samples with or without text on them. We also checked on the kurtosis values, that point how disperse/concentrated a distribution is, but the sought range of values for this moment was not clearly segmented for text/non-text.

Then, the likelihood of each hypothesis containing text is obtained using the described histogram cues and a basic geometric cue (r as the ratio between length and width of the rectangular region) as follows:

• We evaluate these expressions for each hypothesis R_i:

Type	expression	weight
Geometry:	$(r \geq 0.5) \wedge (r \leq 10)$	5
Texture:	$(\bar{x} - M_o) \geq 1$	1
	$\sigma \geq 2$	1
	$\exists(Max_1(h) \wedge Max_2(h))$	2

being $\bar{x}$, M_o and σ the average, mode and standard deviation respectively, and h the histogram of the evaluated hypothesis with two local maxima Max_1 and Max_2.

• Each matched criteria increases the number of votes v_i for rectangular hypothesis R_i being a text area. Each criteria votes according to the weight shown in previous step. This way we can control which criteria is more important, for instance depending on how certain we are that it should be always true. These votes are used to estimate a simple likelihood of the hypothesis being of type $Text$, as follows:

$$P_{ini} = P(R_i|Text) = \frac{v_i}{|V|} \quad (1)$$

being $|V|$ the maximum amount of votes that can be obtained.

• Hypothesis with $P_{ini} > 0.75$ are accepted to continue with the rest of the process (except at certain cases with extremely large amount of hypothesis, where only a fixed number of most likely hypothesis are accepted to avoid application overload).

3. Final processing of accepted hypothesis

Once we have selected a set of likely rectangular hypothesis through the geometry cues and the simple likelihood evaluation process, we proceed with a more detailed analysis of the accepted hypothesis, to decide the hypothesis that would be given to the OCR step.

3.1. Frontal projection through homographies

OCR methods are shown to perform better when text is presented in a frontal, zoomed and centered view. Therefore, the key idea in this step is to project the current image to the equivalent that would be seen if the sign would have been seen frontally and centered.

"Virtual" frontal view. We suggest a simple way to estimate how a "virtual" frontal view of the sign will look like, as shown in Fig. 6. The text we are looking for must be contained on a plane in the 3D scene (the sign board) with rectangular shape. Let us suppose ABCD are the corners of the rectangular hypothesis we have instantiated. If it would correspond to an actual rectangular sign, its frontal view should contain parallel and perpendicular edges. Depending on the orientation of sides and angle between hypothesis edges, the process decides which sides to keep as reference and which weight should be applied to the other dimensions. For instance, in the example in Fig. 6, we keep vertical edges as reference because they are closer to their ideal slope. Then, the longest vertical side, $\bar{AC}$ is kept as it is and the horizontal sides are "projected" to be perpendicular to $\bar{AC}$ with an increase on its length inversely proportional to angle α. This angle is always estimated between 0 and 90^o, and we establish a maximum increase of 60% on the side length. Thanks to this guess about how the corners should be projected, we can estimate an homography to automatically project the original view into the virtual one as detailed next.

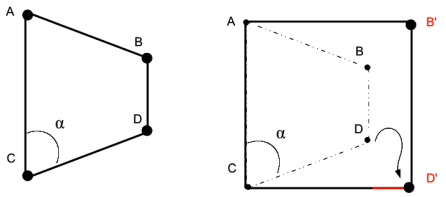


Figure 6. Current view of the rectangular region (left) and goal "virtual" frontal view (right).

Homography estimation An homography, **H**, is a well known projective geometry transform [5], that relates two different planar region projections. We estimate it between the "virtual" image and the original one to obtain a mapping from any pixel **x** that belongs to the sign plane in the original image to the corresponding point **x'** in the other view:

$$\mathbf{x}' = \mathbf{H}\,\mathbf{x}. \tag{2}$$

In general, four corresponding points between the two views of the planar surface are enough to estimate the homography between both views. Therefore, we estimate where the four corners of current hypothesis rectangular region (A, B, C, D) may appear in the virtual frontal view (A', B', C', D'), and estimate the homography with these correspondences. Now we can project the whole hypothesis region according to the estimated **H**. Figure 7 shows two projection examples with images acquired from the mobile phone. We should note that even if the shape we are guessing for virtual view is not accurate, the goal of getting the text in a frontal and homogeneous view (similar size for all characters) is nicely achieved by this simple idea.

Figure 7. Two examples of the projection from an original image (left) to a virtual frontal view (right).

3.2. Re-evaluation of hypothesis likelihood

Finally, we have observed that even though some OCR methods include a binarization step, we can include a more flexible one thanks to the previous steps run in our proposal. Then, a re-evaluation of the likelihood of being text after the projection and binarization steps is run, to try to include the new information that may have been generated there.

Image adaptive binarization. Since we have already cropped the image rectangular region and computed the gray level histogram over the hypothesis region, we analyze its mode to determine a flexible binarization threshold that depends on it.

Figure 8. Final evaluation of segments in the hypothesis area. Long segments still found inside the region (top segment, in red) are used to crop it. Distribution of small segments (in green) inside the 8 region cells (marked with long vertical edges) is used to update the hypothesis likelihood of containing text.

Straigh segments analysis. The final likelihood evaluation of containing text for the remaining hypothesis depends on the distribution of straight segments on the hypothesis region. We extract straight segments on the binarized image, accepting segments of five pixel length or more. Two interesting goals are achieved in this part of the process. First, if long segments (that occupy most of the length or width of the rectangular area) are found, they are used to further crop the view. This way we clean a little more the view that will be provided to the OCR. Secondly, we divide the rectangular region in several cells as shown in Fig. 8 and we re-evaluate the likelihood of the hypothesis being text, P_{final}. We combine the information regarding straight segments distribution, P_{lines}, in the hypothesis with the initial estimated likelihood (1) as shown in (3). Then, we select the most likely hypothesis, maximum P_{final}, together with hypothesis with a likelihood within 80% of that maximum.

$$P_{final} = P_{ini} * P_{lines} \tag{3}$$

P_{lines} is computed from the count of cells containing small segments: $P_{lines} = \frac{C_i}{|C|}$, where C_i is the number of cells that contain at least one straight segment and $|C|$ is the number of cells (8 in our experiments). Note that this re-evaluation is conservative, only decreasing the final probabilities of hypothesis being text, but it does not directly reject any, unless there are no straight segments detected. If this happens, we are quite certain that there are no sharp characters or that the binarization step went wrong and all the region in the hypothesis became black or white, losing the details. Besides, in order to present a prototype running on the mobile phone, we need to keep acceptable execution times. As mentioned before the OCR step is the most expensive one and this filter helps processing with it as few hypothesis as possible. The experiments show some examples of the benefits of using this final filter.

3.3. Integration with the OCR step

The hypothesis accepted at this point will finally be processed by an OCR to recognize their text. We have evaluated three of the most used OCR methods available as open

source to be able to integrate them in our prototype: Ocrad[4], Tesseract[5] and Gocr[6]. To choose which one was more suitable for our prototype, we performed some tests to evaluate their results with several images with text signs on them. Results and analysis of these tests are summarized in Table 1, where each column corresponds to a different OCR library. First row shows the average percentage of characters appearing in an image that are correctly identified by the approach; Second and third row summarize two additional criteria, besides performance, of interest for building the prototype: possibility of using different trained model depending on the language to be used and ease to integrate with the iOS prototype, respectively. According to this brief analysis, we decided to use the best compromise option, tesseract, to implement the final prototype described and tested in the following section.

Table 1. Evaluation of three open source OCR libraries.

	Ocrad	Tesseract	Gocr
correct recognition	84 %	77%	57 %
different languages		+	
easy integration	+	++	+

4. Prototype developed and evaluation

We have designed and implemented an iOS application to demonstrate our proposal in the mobile phone. The prototype have been tested with around 200 different images, with different resolutions, acquired from a mobile phone. Our test images are divided in two types, frontal and oblique views, that are usually easier and more difficult to recognize respectively for an standard OCR. Some of the test images are shown in Fig. 12. Our proposal evaluation has been twofold: on one hand, checking the usability of the application; on the other hand, evaluating the performance and improvements observed with the different steps proposed.

4.1. Prototype design and implementation on iOS

The application designed runs three basic stages. First, the user has to choose an image from the *iPhone Photo library* or acquire a new image with the device camera. Second, the proposed processing is applied to the chosen image. This step is transparent for the user, who must just wait until it is finished. Once the text is recognized, besides showing it on the screen, we have implemented two useful possible final applications: one option sends the text to an online translator; the other one just reads the resulting text using the *voiceover* option at iOS. An example of the application running at the different explained stages is shown in Fig. 9. More details on the different steps running on different test cases can be found in the video provided as additional material to this paper (recorded while running the process in the iOS simulator platform).

(a) (b) (c)

Figure 9. Screenshots of an execution of the proposal. Image acquired (a), result after automatic processing (b) and final use of the answer, translation from spanish to english in this case (c).

4.2. Evaluation of hypothesis likelihood estimation

First, we present a brief analysis of the proposed hypothesis likelihood evaluation for containing text. We checked the classification of every generated hypothesis (447) in a few images, with a likelihood acceptance threshold of 75%. A summary of these results is presented in Table 2. True Negatives (TN) in this case are hypothesis that did not contain a text sign and were properly rejected, while True Positives (TP) are hypothesis accepted that did actually contain a text-sign. First column of results shows performance before projecting the hypothesis to frontal virtual views. At that point, critical issue is to avoid false negatives, and we can appreciate that very few of the rejected hypothesis did contain text. Final results shown in the second column correspond to the final results of our proposed process, just before running the OCR step. We can appreciate how the projection and simple line segment analysis proposed in section 3 further improve the results. Accuracy is higher, i.e., we achieve better classification of the hypothesis. Now more hypothesis without text are rejected so the accuracy of the text class recognition increases to 0.90 from 0.63. This last re-evaluation step, subsection 3.2, significantly reduced the amount of hypothesis accepted to be processed in the last OCR step (the percentage of processed hypothesis was decreased from 60% to 18 %). Besides, we observed that none of the hypothesis rejected after this step would be

Table 2. Hypothesis classification accuracy.

		Before projection	After projection and line filtering
No text	$\frac{TN}{TN+FN}$	0.92	
Text	$\frac{TP}{TP+FP}$	0.63	0.90

[4]http://www.gnu.org/software/ocrad/ocrad.html
[5]http://code.google.com/p/tesseract-ocr/
[6]http://jocr.sourceforge.net/

properly read with the OCR step. This re-evaluation just helped to re-rank likely hypothesis. Figure 10 shows an example where the initial texture analysis incorrectly gave a high P_{ini} of being text, but the low amount of segments found decreased its final likelihood to the half.

Figure 10. Re-evaluation step provides better likelihood estimation in cases where simple histogram analysis fails. This example got high P_{ini} but low P_{lines}, properly decreasing the final likelihood of being text because there are very few straight segments (green).

4.3. Evaluation of final recognition performance

In order to demonstrate that the proposed process helps in the aimed applications, we carried out a first detailed evaluation using 10 different images of each type. These tests consisted of calculating the percentage of characters appearing in the image which were correctly recognized using different open and commercial OCR or running the whole process proposed in our prototype before the OCR. Besides the chosen open-source library to integrate in our prototype (tesseract) we also evaluate the improvements obtained when using the OCR available in Google Docs web application and the commercial OCR ABBYY (available for trial version). To run these experiments, we saved the processed hypothesis images that were provided to the OCR step during the execution of our proposal, to provide exactly the same ones to the other OCR methods. Table 4.3 shows the average μ and standard deviation σ in the percentage of image characters correctly recognized for different configurations. First two columns show results with our proposal run together with the OCR and last two columns were obtained just running the OCR. Different rows present results using easier tests (frontal views) compared to harder cases (oblique views). We can see the pre-processing proposed significantly improves character recognition results for most cases. As expected, it especially helps with oblique views thanks to the re-projection to virtual frontal views.

We analyzed in more detail the results using the OCR integrated with our prototype. This second set of tests, consisted of 50 images of each type, frontal and oblique views. We count how many "correct readings" we obtain, considering a good reading when the text in the sign has been recognized clearly enough for a person to understand the meaning. Although this is a subjective criteria, we typically observed in our tests that over 80% of characters recognized can provide a good idea of the meaning. Besides, in most of that cases the "grammar correction" features from text editors are able to suggest the correct word. Figure 11 summarizes the results of this experiment, with clear improvements when incorporating our preprocessing steps.

Table 3. OCR performance with and without our proposal (Pre).

	Pre + Tesseract		Tesseract	
	μ	σ	μ	σ
Frontal	74,89 %	32,73%	36,76 %	38,98 %
Oblique	47,45 %	45,46%	16,44 %	31,34 %
	Pre + GoogleDocs		GoogleDocs	
	μ	σ	μ	σ
Frontal	53,12 %	49,96%	47,44 %	45,42 %
Oblique	30,00 %	48,30%	4,21 %	13,31 %
	Pre + ABBYY		ABBYY	
	μ	σ	μ	σ
Frontal	76,53 %	34,42%	77,72 %	41,12 %
Oblique	54,29 %	49,85%	19,00 %	35,17 %

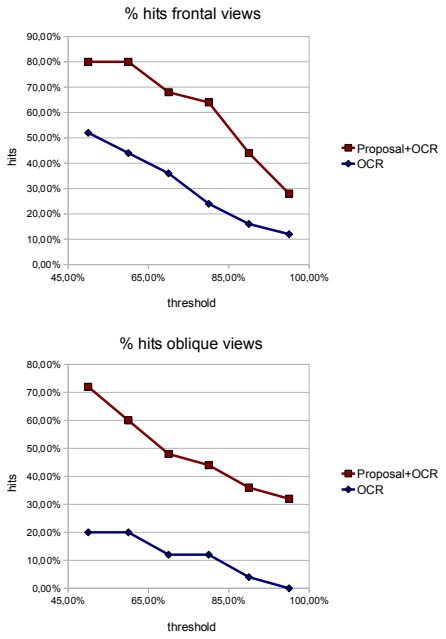
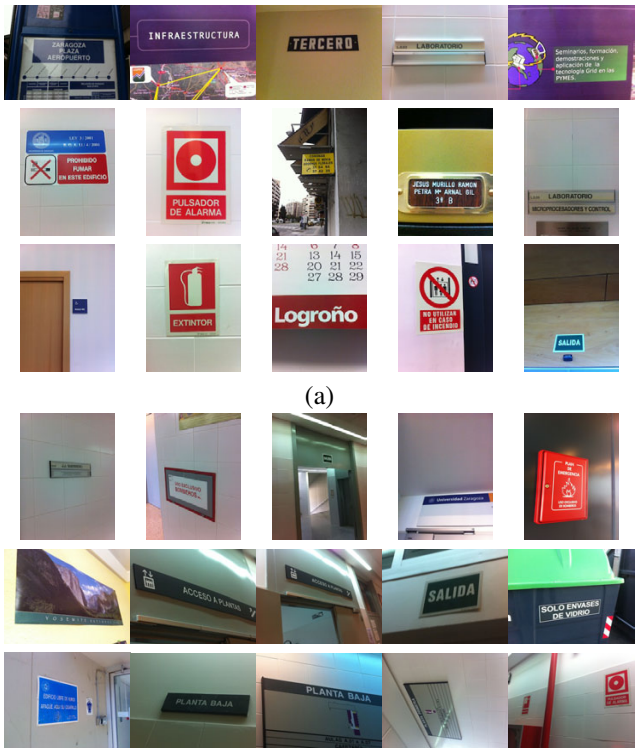


Figure 11. Percentage of tests considered as correct readings (hits), using different acceptance threshold on the % of recognized characters for frontal and oblique views.

5. Conclusions and Future Work

We have presented in detail an approach to read text-signs in images acquired from a mobile phone in general scenes. The whole process is described, evaluated and demonstrated in a functional prototype working on iOS. Many image processing and classification techniques are too costly to run on the mobile phone, so we try to kept this in mind to design our prototype steps. The key ingredients of our proposal are a simple and efficient analysis of the image based on geometric cues. This allows that two additional basic texture analysis steps are enough to filter the



(a)

(b)

Figure 12. Some of the test images used to evaluate the proposal, both from frontal (a) and oblique (b) views.

best regions of interest. Currently our approach is aimed to rectangular text signs, to facilitate some geometric assumptions needed for the homography based image rectification step. We study how to present the selected regions of interest in an optimal way to be processed by an standard OCR and demonstrate the improvements of including our pre-processing before running several standard OCR techniques. Future work improvements can be done towards a more general and more efficient approach. For instance, the approach could be extended to other geometric shapes, finding alternative ways of estimating the homography for rectification. Regarding efficiency, some steps could be optimized by for example detecting area overlap between hypothesis to save some repeated computations.

Acknowledgments

This work has been supported by projects DPI2009-14664-C02-0 and DPI2009-08126.

References

[1] X. Chen and A. Yuille. Detecting and reading text in natural scenes. In *Computer Vision and Pattern Recognition*, pages II–366 – II–373 Vol.2, 2004. 2

[2] P. Clark and M. Mirmehdi. Combining statistical measures to find image text regions. In *Proc. of International Conference on Pattern Recognition*, pages 450–453, September 2000. 2

[3] N. Ezaki, M. Bulacu, and L. Schomaker. Text detection from natural scene images: towards a system for visually impaired persons. In *Int. Conf. on Pattern Recognition*, volume 2, pages 683 – 686 Vol. 2, aug. 2004. 2

[4] S. Ferreira, V. Garin, and B. Gosselini. A text detection technique applied in the framework of a mobile camera-based application. *Proc. of Camera-based Document Analysis and Recognition, Seoul*, 2005. 3

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 5

[6] M. C. J. Fabrizio and B. Marcotegui. Text extraction from street level images. *City Models, Roads and Traffic (CMRT)*, pages 199–204, 2009. 2

[7] J. Košecká and W. Zhang. Video compass. *ECCV '02 Proceedings of the 7th European Conference on Computer Vision*, pages 476–490, 2002. 3

[8] H. Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. *Image Processing, IEEE Transactions on*, 9(1):147 –156, jan 2000. 2

[9] C. Mancas-Thillou, S. Ferreira, J. Demeyer, C. Minetti, and B. Gosselin. A multifunctional reading assistant for the visually impaired. In *EURASIP Journal on Image and Video Processing*, 2007. 2

[10] S. Mori, C. Suen, and K. Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80(7):1029 –1058, jul 1992. 2

[11] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Proc. of the 10th Asian Conf. on Computer Vision*, volume IV, pages 2067–2078, November 2010. 3

[12] M. Tanaka and H. Goto. Text-tracking wearable camera system for visually-impaired people. In *ICPR*, pages 1–4, 2008. 2

[13] Y. Tian, C. Yi, and A. Arditi. Improving computer vision-based indoor wayfinding for blind persons with context information. In *Proc. of Int. Conf. on Computers helping people with special needs*, pages 255–262, 2010. 2

[14] K. Wang and S. Belongie. Word spotting in the wild. In *Proc. of the European Conference on Computer Vision*, pages 591–604, September 2010. 3

[15] C. Wolf and J.-M. Jolion. Extraction and recognition of artificial text in multimedia documents. *Pattern Anal. Appl.*, 6:309–326, February 2003. 2

[16] V. Wu, R. Manmatha, and E. M. Riseman. Finding text in images. In *Proc. of the Int. Conf. on Digital libraries*, DL '97, pages 3–12, New York, NY, USA, 1997. ACM. 2

[17] Q. Ye, J. Jiao, J. Huang, and H. Yu. Text detection and restoration in natural scene images. *Journal of Visual Communication and Image Representation*, 18(6):504 – 513, 2007. 3

[18] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai. Attacks and design of image recognition captchas. *In Proc. of Conference on Computer and Communications Security*, 2010. 2