TranslatAR: A Mobile Augmented Reality Translator on the Nokia N900

Technical Report 2010-12

Victor Fragoso* vfragoso@cs.ucsb.edu
Steffen Gauglitz sgauglitz@cs.ucsb.edu
Jim Kleban kleban@ece.ucsb.edu
Shane Zamora char42@gmail.com
Matthew Turk mturk@cs.ucsb.edu

ABSTRACT

Researchers have long been interested in the synergy between portability and computing power but had been limited by unwieldy, uncommonly used devices. The latest generation of mobile phones, i.e. 'smartphones', are equipped with hardware powerful enough to develop novel, interesting applications with allow users to directly interact with the world around them. This paper describes a multimodal, augmented reality translator developed using a smartphone's (Nokia N900) camera and touchscreen combined with OCR (Tesseract) and online translation services (Google Translation API). We describe our methods for tracking, text detection, OCR and translation, and provide results quantifying OCR accuracy on a set of signs collected around the UCSB campus.

1. INTRODUCTION

Have you ever been lost in a foreign country wondering what the sign in front of you indicates? China (when you don't read Chinese) can be particularly bewildering in these situations. A system which can read text characters in the real world and translate them to your native language would be useful while abroad. With the improving imaging, processing, storage, and wireless networking capabilities in today's smartphones, real-time multimodal systems which aid the user in understanding the environment are becoming feasible. This paper presents a system implemented on a Nokia N900 phone which allows the user to simply hold up the phone, and, with a single click, have text translated into his/her language of choice and see the translation through the phone's display ("magic lens" paradigm). This idea is illustrated in Fig. 1.

Our system was prototyped using standard off-the-shelf libraries which support multiple languages. Much of the computer vision to detect, track, and overlay text is accomplished via the OpenCV library. The optical character recognition (OCR) comes from an open source project called Tesseract, and the translation is provided by Google's Translate web service. Our system was developed on the Nokia N900 platform running on Maemo 5 Linux offering commonly used development tools. This makes our application TranslatAR easy to extend and improve by other developers compared to systems on other platforms. This paper will provide an overview of how our augmented reality translation application works. We also quantify the OCR performance to provide a feel for how often the translation can succeed.

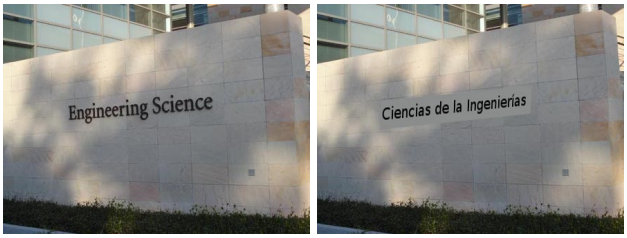

Figure 1: With a single click, TranslatAR detects text area & orientation in a video frame, calls a translation module in the background, and overlays the obtained translation onto the live video.

The remainder of this paper is structured as follows: Section 2 gives an overview of related work. Section 3[1] provides details on the OCR & translation core of our system, Section 4[2] explains the structure of the video processing "frame", including processing of the user's input, live tracking, and video augmentation. Both sections include analysis of specific key aspects. Finally, Section 5 concludes.

2. RELATED WORK

Automatic translation helpers. Various stand alone electronic translation devices are available that provide dictionary and translation functionality, such as the Franklin TGA-

*All authors contributed equally to this project, names appear in alphabetical order.

[1](Jim & Shane)
[2](Victor & Steffen)

470 Global Translator. Language translation applications that provide dictionaries or text string translation capabilities are also available on mobile devices. iTranslate is an iPhone app that provides two-way translation between five languages, and provides text-to-speech functionality for each language. Jibbigo is an iPhone app that provides speech-to-speech translation between Spanish and English. Paul et al. [11] developed a distributed speech-to-speech translation system between Japanese and English. These systems however require the user to type in the text to translate or to (be able to) voice and pronounce the text in question.

OCR & Visual translation. The components of locating, recognizing, and translating text have been the focus of extensive research efforts. Optical character recognition (OCR) [8] has a long history with the primary goal of digitizing scanned documents. While classical OCR systems are capable of recognizing text with high accuracy while maintaining basic structural/layout information, they typically require high signal-to-noise ratio and correct, distortion-free orientation of the text.

Systems for automatic translation of signs have been previously devised by researchers. Yang et al. [17] devised a system for Chinese signs that combined textual OCR with symbol detection. The main contribution of their work consisted of sign extraction. Their prototype, Smart Sight [18], consisted of an unwieldy wearable computer with a video camera, was limited in language flexibility, and did not use the magical lens paradigm. It did, however, provide additional feedback in the form of speech synthesis. Haritaoglu [2] also developed a system for automatic translation of Chinese signs on mobile devices using a PDA with an attached camera. Watanabe et al. [16] developed a system for automatic detection and translation of Japanese text into English using a camera on a mobile phone. While these aforementioned works are most similar to ours, we offer a particularly easy-to-use (single click) and compelling (augmented reality overlay of the result) user interface.

Localizing text. Researchers [6, 3] have been interested in locating and recognizing text in video streams for purposes including information retrieval and license plate identification. Park & Jung [10] developed a system for automatic detection of words in images from mobile devices. Liu et al. [7] developed an edge-based method for text extraction for mobile robots that provides excellent results in extracting text from scenes. Palaiahnakote et al. [9] developed another text detection technique in a video streaming. Both of these approaches however assume that text is roughly horizontal and seen without perspective distortion. In contrast, our approach is robust to significant perspective distortion, not requiring the user to stand perpendicular to the text plane.

Tracking on mobile phones. Visual tracking without visual markers in real-time has succesfully been demonstrated on mobile phones, both with known targets [14, 15] and without [5]. Focusing solely on tracking, above systems are technically more advanced and robust than the tracking used here, but many aspects of our approach are similar to [5, 4].

Threaded bolt algorithm has circle calculations. Go to the threaded bolt algorithm will l than "None" (see item 5 belo algorithm will run.

Figure 2: Images of strings of text captured by the primary camera of the N900 device (left), compared to scanned documents (right) that comprise the majority of documents provided to OCR platforms for text extraction.

3. PART 1: OCR & TRANSLATION

The core components of TranslatAR that provide the novel functionalities we are interested in are the OCR and written language translation components. Our goal was to find the most robust free or open-source packages that could perform these tasks on a mobile device. We chose to employ the Tesseract OCR Engine and Google's Translate web service for these purposes.

Tesseract is an OCR engine developed by HP from 1985 to 1995 until the project was continued and released as open-source by Google in 2005. Tesseract is easily compiled on our target platform and is considered one of the most accurate freely available OCR packages. Google's Translate web service is freely available to any device with an internet connection, requires no platform specific implementation, and provides a "Detect Language" functionality on the text provided for translation. As TranslatAR could be used to translate text from any language into any other language, Google Translate handles all combinations of languages for free. We will describe the details of employing both systems on a mobile device in the following subsections.

3.1 OCR with Tesseract

Tesseract is a command line tool that accepts a TIFF image as input, and returns a text file containing the captured text. It performs none of the document or layout analysis that other OCR packages provide, besides the seperation of horizontal lines of text in the image by newlines. There are also no options by which to request additional functionality – Tesseract is instead a raw OCR engine that focuses primarily on character recognition accuracy. It is open source, compiles easily on any platform, and performs all that is necessary for TranslatAR's purpose.

However, the primary domain that current OCR engines optimize for are scanned documents. Scanned documents are typically 300 or higher dpi images with high contrast and little noise. Our application uses a video feed from the camera of a Nokia N900 which captures at a 640x480 resolution with relatively little contrast and copious amounts of noise. Fig. 2 compares some examples of text captured with a Nokia N900 versus a scanned document.

We developed an application that prototyped the usage of Tesseract on the N900. The application would display a video feed onto the device's screen, on which the user would tap to freeze the stream. The user would then draw two lines flush to, above, and below the text to be captured for recognition. The four points of these two lines would be used to

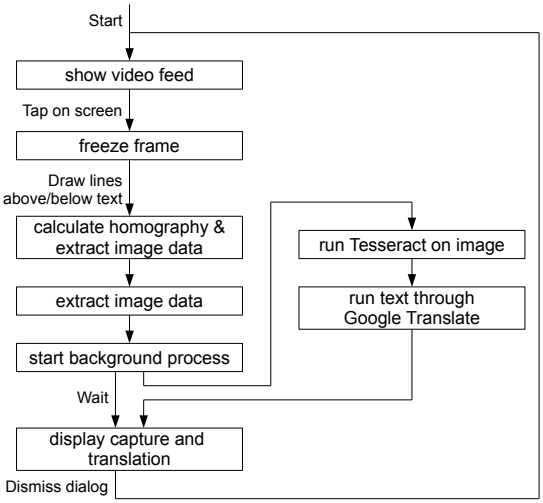

Figure 3: Structure of the OCR/Translation prototype.

Figure 4: Some of the extracted text regions from the test set of 42 signs gathered around the UCSB campus using the Nokia N900 camera.

Dataset	#Chars	Acc	#Words	Acc
News.3B [13]	7524	98.47%	1220	97.51%
Signs, N900	855	87.01%	169	72.78%
Signs, Sony	855	87.60%	169	74.56%

Table 1: OCR rates

compute the enclosed quadrilateral's homography and perform a perspective correction in order to provide better oriented input to Tesseract. Finally, the rectified image data would be saved as a TIFF image and Tesseract would be to used retrieve it's containing text.

Our implementation used OpenCV in order to retrieve a video feed from the N900's video camera. Although the camera on the device can be accessed using GStreamer to gain better control over the video feed, we were not successful in using it to gain any particular advantage to what was more easily provided by OpenCV. We computed the homography transformation with CLAPACK by using the singular value decomposition (SVD) of a composite matrix. GTK+ and the Hildon library developed for Maemo 5 (the N900's operating system) were used to provide a user interface for video playback and touch screen tapping and line dragging. Finally, we pop up a seperate window showing the rectified image, the captured text as provided by Tesseract, as well as the translated text provided by Google Translate. A flow chart of our prototype's functionality can be seen in Fig. 3.

Analysis. Some examples using our prototype can be seen in Fig. 4. In order to determine the effect of using the OCR package on a lower resolution camera and on signs imaged in the real world we collected a test set and measured performance. We used the Nokia N900 and a Sony Cybershot 7MP digital camera to take photos of 42 signs around the UC Santa Barbara campus in order to test the character and word accuracy rates of Tesseract using data captured from the N900 versus a more powerful camera in order to compare results to a more ideal hardware configuration.

Results are shown in the Table 3.1. As a baseline, the first row in the table shows character and word accuracy rates for Tesseract on a standard document text data set, News.3B [13]. The respective rates of 98.47% and 97.51% are fairly high and competitive with the state of art in OCR. The next two rows compare the results we achieve with the N900 camera to the best rate for the Nokia camera found after downsampling the input images to a maximum width 600 pixels. In both cases, the location of the text was manually annotated and warped using a homography as described previously in this section. Character recognition accuracy was measured by determining the number of correct characters returned out of the number possible in the input image. As can be seen from the table, camera quality (resolution, autofocus) does not make much of a difference. The Sony only slightly outperforms the N900 for this set of images containing mostly large fonts. Surprisingly, the Sony performed worse when using the full 3072 x 2304 pixel image as many small noise spots were falsely identified as characters. The resulting rates of 87.01% character rate and 72.78% word rate show that the challenges of imaging real world signs, illumination, perspective, scale, and other photometric variances have a significant impact on performance. Word rate drops quickly with increasing character errors, these rates would not be acceptable for scanned documents.

3.2 Translation with Google Translate

Google provides a free real-time web-based service which provides the translation necessary for our application. In addition to translating to a specified destination language (which the user can select), the Google Translate API also can detect the source language of the input text. Primarily released in order to aid the translation of webpages, the API responds with JSON to HTTP GET/POST requests. It can translate from to at least 51 languages including Chinese, English, German, French, Russian and Spanish. Unlike traditional translation systems which use rules and dictionary lookups, Google's approach to translation has been to learn it via accumulating data on known translated doc-

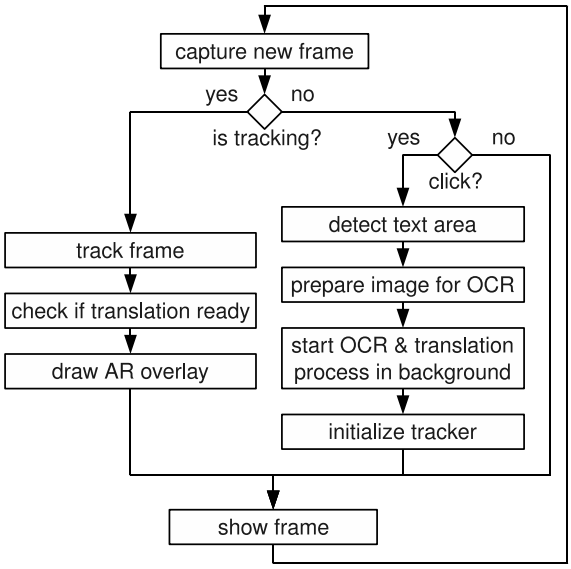

Figure 5: Main structure of the video processing system.

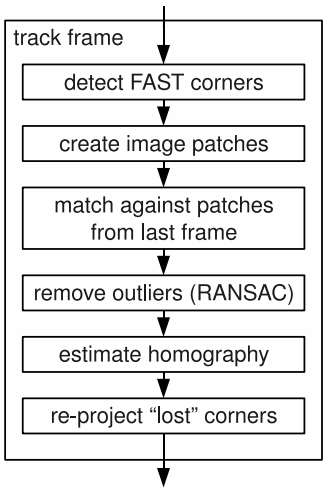

Figure 6: Tracking process.

ument pairs. Integration of the API into TranslatAR is straightforward using the curl C++ library and then parsing the results returned from Google's server. The resulting text is passed the augmenting module. One downfall of the translation API is it cannot currently handle spelling errors, which is a particularly acute problem given poor OCR.

4. PART 2: USER INPUT, TRACKING & VIDEO AUGMENTATION

While part 1 covers the core functionality of TranslatAR, the objective of this part is to increase the ease-of-use and the level of immersion for the translation. Specifically, instead of "freezing" the video stream and requiring the user to outline the text area manually, the user only has to "tap" on the word he/she would like to translate, the area is automatically tracked, and as soon as the translation (as provided by part 1) is ready, it will be overlaid on the video stream in the right colors and orientation.

The main tasks for this part are:

- a framework to process a live video stream from the camera's viewfinder,
- detecting the text area given a single "tap" on the screen,
- real-time tracking of the region of interest over a short period of time,
- overlaying the provided text on the video.

Fig. 5 shows the main structure of the overall system: Initially, the system just grabs and displays frames. Upon the user's "tap", the text area is detected, the OCR & translation process (i.e., part 1) started, and the tracker is initialized. During tracking, a translation (or a placeholder until the translation is available) is drawn as an "augmented reality" overlay on the video. The main components will be described in detail in the following subsections. Section 4.4 presents a short analysis of key aspects.

4.1 Tracking

Fig. 6 shows the components of the tracking process. First, FAST corners [12] are extracted from the video, then matched against the corners from the last frame using a simple 9x9 image patch descriptor. To speed up the matching process (which otherwise is quadratic in the number of keypoints), the matching is contrained to a small circular area around the last position, and descriptors are only computed (i.e. sampled from the image) when needed.

After this, RANSAC is used to sort out outliers and finally, a homography between the last and the current frame is estimated, minimizing the squared error of all "inlier" matching pairs. Note that the estimating a homography implies that the area to be tracked is planar, which is a fair assumption considering that we are interested in a fairly constrained area containing text. Finally, all "lost" interest points are reprojected into the new frame so that the number of tracked points remains the same – this is computationally more efficient than searching for new keypoints to replace the lost ones.

Note that both detector and descriptor are the same that were used in Klein & Murray's seminal work "Parallel Tracking and Mapping" [4], where the authors opt for many very efficient to compute features rather than few expensive ones. For our work, FAST was chosen as it faster than other popular interest point detectors by at least one order of magnitude [1]. The (compared to other detectors) rather low repeatability for noisy video [1] is acceptable, as we assume a cooperative user and only require tracking over a short period of time and small to medium changes in viewpoint. For the same reasons, image patches are the most suitable descriptor, as they require very little time to compute and invariance to in-plane rotation or robustness for large baseline matching is not required.

4.2 Text Detection

Fig. 7 illustrates the main steps an overall of the text detection process. The text detection module takes the current frame and the point (x_p, y_p) provided by the user's "tap". First, both horizontal (I_x) and vertical (I_y) image gradients are computed using a first-order Sobel-Scharr kernel.

An approximate text bounding box is determined as follows: Starting from (x_p, y_p), the vertical extent (i.e., text height) is determined by expanding a vertical strip until the maximum vertical gradient encountered along its upper and lower edge fall below a predefined threshold (indicated by the red lines in Fig. 7(a) –denote by y_u and y_l, respectively). Then the horizontal extent is computed with the same method, using the approximated text height to determine both the height of the horizontal strip and the width of the gap that is required to stop the expansion. Resizing the gap with the text prevents that the algorithm stops between two letters while being more invariant to different text sizes. The approximated bounding box is indicated by the red lines in Fig. 7(b).

Then, a constrained and modified "Hough Transform" is used to determine the exact position and orientation of the text: Considering all lines that cut the segment $\overline{(x_p, y_u), (x_p, y_l)}$ (bright red line in Fig. 7(c)) and within an angle of $0° \pm 15°$ (quantized with 1-pixel and $1°$ steps, respectively), select the line segment that maximizes

$$\sum I_x(x, y) - I_y(x, y)$$

where the sum includes all pixels on the line (bilinearly interpolated) between the left and right limit. That is, select the line that is best supported by horizontal edges while cutting few vertical edges. Fig. 7(c) illustrates the resulting line segments for this example (one each for lower and upper half). Finally, the lower line is shifted down until it goes through (x_p, y_l), to include potential descenders, and vice versa for the upper line. The final area is outlined in red in Fig. 7(d).

Note that this algorithm is robust to significant perspective distortion, not forcing the user to stand perpendicularly in front of the text, but not to in-plane rotation, assuming that the user will hold the phone approximately horizontal.

4.3 Video Augmentation

The last step of TranslatAR is to display the translation by augmenting the video stream over the region of interest. This process consits of two parts: extraction of foreground and background color and rendering of the graphical overlay.

Foreground-Background Color extraction. The extraction of these colors is executed before calling the OCR, taking as input the rectified image clip containing the text (i.e. the same image that the OCR system receives). The background color is extracted by sampling a horizontal line in the upper bound and taking the average as estimation, assuming that the background will be roughly uniform.

Given that the text must be clearly readable, we may assume that the foreground color has a strong contrast with the background. Taking this into account, the foreground color estimation is done by scanning pixels starting from a point in the center, and accepting a sample as estimation of the foreground color if its intensity is sufficiently different from the background. This technique was chosen over more accurate estimators (such as k-means clustering) due to its very low processing time requirements.

Component	Time
capture frame	24 ms
frame preprocessing	34 ms
detect keypoints	4 ms
descriptor & matching process	24 ms
RANSAC & homography estimation	22 ms
reprojection of "lost" keypoints	1 ms
draw AR overlay	24 ms
total time per frame	263 ms

Table 2: Examplary execution time for the main steps of the processing pipeline. Frame preprocessing includes color conversion (from the camera's native YUV to RGB for the display, and to grayscale for the tracking) and downsampling (from 640x480 to 320x240). Time for descriptor creation & matching cannot be broken down further, as descriptors are only created when needed and thus are intertwined with the matching, same for RANSAC & homography estimation.

Overlay process. The overlay process is the final routine in the main system pipeline. The translated text is rendered in a separated image using the background and foreground color by OpenCV routines. A warping transformation is then applied in order to fit the requested area to overlay and finally a mask operation is applied to "merge" the current frame and the rectified translation. By doing this, we overlay the translation onto the current frame.

4.4 Analysis

Time. Table 2 shows some representative timings for the video processing & tracking pipeline. So far, no special hardware acceleration has been employed, however, we went to several iterations of optimizing the code and removing bottlenecks, most notably, moving memory allocation into startup were feasible, making sure that no redundant or superfluous computation occurs (such as information that is needed at multiple different components, or descriptors that are actually never matched), and a per-row-access look-up table for keypoints (see [4]).

Table 2 shows that there is no single component which consumes a majority of the computation time, and hence it is difficult to achieve a significant speed-up with a single measure. However, several measures would be reasonable to implement:

- Compared to the difficult task of real-time tracking, the time to capture & preprocess the frame seems very high. As obvious from the N900's built-in camera application, much higher framerates are feasible here. Es-

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

Figure 7: Text detection operating on the image shown in Fig. 1. First, the vertical extent of the text is determined (a), then – using the assumed text height – the horizontal extent (b). A constrained "Hough Transform" is used to determine the exact baseline and orientation (c), and finally, the area is expanded to account for ascenders and descenders (d).

Result	# of images	Percentage
very good	22	56.4%
acceptable	8	20.5%
bad	9	23.1%
total	39	100%

Table 3: Text detection accuracy on 39 signs (same dataset as used in Section 3). "Acceptable" performance is when the detected area is approximately correctly oriented, but larger than necessary. On these images, the OCR is still likely to work correctly, but the AR overlay will look "off".

pecially color conversion is perfectly parallizable and should be moved e.g. to the GPU.

- Likewise, drawing the text augmentation (which on the CPU has to be implemented as rendering of the text, warping the text, and merging the text onto the video frame) should be implemented with the help of OpenGL shaders. We started implementing a respective solution, but were not able to finish it in time due to some peculiarities in the off-screen initialization process of OpenGL ES 2.0.

Tracking Performance. Admittedly, our application is a reasonably simple case for tracking compared to other tracking applications [5, 15]: the target is known to be well-textured (as it contains text) and planar, we require tracking only over short periods of time, and do not need the capability of "expanding" tracking areas or recovery. Moreover, we can assume that the user is cooperative and focuses on a single object.

However, given this (and with the cutback of low framerate, see above), the tracking works quite well: it is capable of "following" the text long enough to retrieve and read the translation despite normal amounts of jitter and supports changes of viewpoint and "zooming" to inspect the result.

Text Detection Performance. We evaluated our text detection algorithm using the dataset of signs collected on UCSB's campus that was used in part 1. Table 3 lists the obtained results, Fig. 8 shows several examples of both good and bad detection results. Major problems are especially non-uniform background (see Fig. 8 bottom left) and unusual fonts, text proportions, spacing, or sizes (see Fig. 8 bottom right).

5. CONCLUSIONS

We have presented a prototype implementation of a single-click augmented reality translator developed on the Nokia N900 smartphone. The application is capable of overlaying an automatically translated text over a region of interest which is extracted and tracked in the video stream of the camera. While accuracy improves if the user manually annotates the region where the text lies in the scene, this would require interrupting the tracking harming the augmentation. We rapidly developed our prototype within a period of six weeks using many open source tools including OpenCV, Tesseract, and the Google Translate API. While we were able to address all previously set goals and hence consider our prototype a success, there are many issues that require careful engineering before translatAR will be useful for real-world use. Most notably,

- lack of control over the viewfinder's focus currently limits translatAR to relatively large fonts,
- the framerate has to be significantly improved, the most obvious way is to make use of hardware acceleration in many of the image processing & rendering tasks,
- "real world" imaging problems such as illumination variance, glare, and dirt are found to cause significant problems for both our text detection and the Tesseract OCR module. Integration of a spell checker before translation would absorbe some of the OCR errors.

6. ACKNOWLEDGMENTS

The authors would like to thank Matthew Turk and Nokia for providing the hardware used for this work, and Daniel Vaquero and Natasha Gelfand at the Nokia Research Center Palo Alto for their help with various implementation aspects on the N900 (most notably the camera drivers).

7. REFERENCES

[1] S. Gauglitz and T. Höllerer. In-depth evaluation of popular interest point detectors on video streams. Technical Report 2009-08, Department of Computer Science, UC Santa Barbara, May 2009.

[2] I. Haritaoglu. Scene text extraction and translation for handheld devices. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–408–II–413 vol.2, 2001.

[3] A. Jain and B. Yu. Automatic text location in images and video frames. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1497–1499 vol.2, Aug 1998.

Figure 8: Examples of good (top row) and bad (bottom row) text detection. Remarkably, our algorithm was able to very accurately detect the word in the rather sloppily hand-written sign (top right). The failure cases are due to very non-uniform background (left); incorrect exclusion of ascenders (second left); failure to detect horizontal borders as the margin between text and sign is very narrow (second right); letters are very large, so that the expansion algorithm stops *inside* one of the letters (right).

[4] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM Intl. Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[5] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 83–86, Oct. 2009.

[6] R. Lienhart. Automatic text recognition for video indexing. In *MULTIMEDIA '96: Proceedings of the fourth ACM international conference on Multimedia*, pages 11–20, New York, NY, USA, 1996. ACM.

[7] 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*, volume 2, pages 701–706 Vol. 2, July-1 Aug. 2005.

[8] S. Mori, H. Nishida, and H. Yamada. *Optical Character Recognition*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[9] T. Q. P. Palaiahnakote Shivakumara and C. L. Tan. A gradient difference based technique for video text detection. *International Conference on Document Analysis and Recognition*, 1:156–160, 2009.

[10] A. Park and K. Jung. Automatic word detection system for document image using mobile devices. In *HCI (2)*, pages 438–444, 2007.

[11] M. Paul, H. Okuma, H. Yamamoto, E. Sumita, S. Matsuda, T. Shimizu, and S. Nakamura. Multilingual mobile-phone translation services for world travelers. In *Coling 2008: Companion volume: Demonstrations*, pages 165–168, Manchester, UK, August 2008. Coling 2008 Organizing Committee.

[12] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. 2006 European Conf. on Computer Vision (ECCV'06)*, volume 1, pages 430–443, May 2006.

[13] R. Smith. An overview of the tesseract ocr engine. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 629–633, Washington, DC, USA, 2007. IEEE Computer Society.

[14] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proc. 7th IEEE and ACM Intl. Symposium on Mixed and Augmented Reality (ISMAR'08)*, Cambridge, UK, Sept. 15–18 2008.

[15] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 57–64, Oct. 2009.

[16] Y. Watanabe, K. Sono, K. Yokomizo, and Y. Okada. Translation camera on mobile phone. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 2, pages II–177–80 vol.2, July 2003.

[17] J. Yang, J. Gao, Y. Zhang, and A. Waibel. Towards automatic sign translation. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–6, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[18] J. Yang, W. Yang, M. Denecke, and A. Waibel. Smart sight: A tourist assistant system. In *ISWC '99: Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, page 73, Washington, DC, USA, 1999. IEEE Computer Society.