

Image-Based Mobile Service: Automatic Text Extraction and Translation

Jérôme Berclaz^a, Nina Bhatti^b, Steven J. Simske^c and John C. Schettino^b

^a CVLab, EPFL, Lausanne, Switzerland

^b HP Labs, Palo Alto, CA, USA

^c HP Labs, Fort Collins, CO, USA

ABSTRACT

We present a new mobile service for the translation of text from images taken by consumer-grade cell-phone cameras. Such capability represents a new paradigm for users where a simple image provides the basis for a service. The ubiquity and ease of use of cell-phone cameras enables acquisition and transmission of images anywhere and at any time a user wishes, delivering rapid and accurate translation over the phone's MMS and SMS facilities. Target text is extracted completely automatically, requiring no bounding box delineation or related user intervention. The service uses localization, binarization, text deskewing, and optical character recognition (OCR) in its analysis. Once the text is translated, an SMS message is sent to the user with the result. Further novelties include that no software installation is required on the handset, any service provider or camera phone can be used, and the entire service is implemented on the server side.

Keywords: Scene text extraction, mobile phone camera, client / server, translation

1. INTRODUCTION

Imagine a traveler dining in a restaurant in a country on the other side of the world and struggling with the menu. "Is it Beef Steak or Lizard Steak?" could be one of the more pressing questions as the traveler reads through the menu. What if this traveler could use her camera phone to get a translation simply by taking a picture of the menu, sending a MMS and promptly receiving an SMS with the answer? In this paper we describe such a system that uses any camera phone and any mobile service provider to accomplish this task quickly and easily. We selected mobile phones because of their ubiquity and unique position in our lives as a companion device.

In just about ten years, mobile phones have evolved from a luxury accessory to an indispensable object. In 2008 there were more than 3 billion mobile subscribers worldwide and some of the more mature markets have over 100% penetration.¹ This trend continues to grow such that over 700 million camera phones were shipped in 2007 and 1.3 billion units are expected to ship in 2012.² An estimated 13% growth rate is expected each year in camera phone shipments. Camera phones are quickly becoming a network connected ubiquitous visual sensor in everyone's pockets.

While technical capabilities of mobile phones have evolved greatly, consumer use has been changing more slowly. Kindberg *et al.*³ report that most people are still using their cell phone camera as they would do with a standard digital camera – that is, just storing pictures – without taking advantage of the other services such as MMS. In this paper we present a first step in creating intelligent useful services that go beyond simple picture taking, to picture interpretation via powerful *cloud services** not constrained by the software and handset platform. These services provide massive computation to deliver unique capabilities of image interpretation. In this context, there is a high potential for new multimedia-based services making use of underexploited cell phone hardware.

Developing a cell phone application that would be usable by almost all customers regardless of the phone brand and carrier is a tremendously difficult task, due to the large combination of different cell phone brands, operating systems, technologies and carriers. To overcome this difficulty, we have chosen to develop an application that relies on a mobile client connected to cloud services and that would therefore be almost independent of the cell phone technology.

More specifically, we introduce an automatic translation application for cell phones, taking advantage of the cell phone cameras and their MMS transmission capability. In a typical scenario, a user takes a picture of a text area with its cell phone camera and sends it as an MMS to an application server. On the server side, the MMS is received and the image is

*For a definition of *cloud computing*, see http://en.wikipedia.org/wiki/Cloud_computing

extracted. Then the text is identified in the image, extracted and sent to an online translation engine. Finally, the translation result is sent back to the user in an SMS. Within a few seconds of its request, the user receives the translation in his text messages.

As displayed on Fig. 1, our application is completely implemented on the server side, giving it access to as much computer power as needed, as well as to almost unlimited resources. In addition, this framework allows our application to run almost independently of the cell phone used to access it. The only requirements are that the phone has an integrated camera and MMS capability, two features present in almost every cell phone today. Moreover, we have opted for a completely unsupervised application, so that no additional software needs to be installed on the client device. Although portable devices have considerably improved their user interface, it is still not very convenient to define an image region on a tiny screen. An unsupervised method thus requires less user interaction, and leads to a smoother process.

The rest of this paper is articulated as follows: Section 2 explores state-of-the-art related work, Section 3 describes our prototype system in many details and Section 4 discusses our results and explains the advantages of the framework we advocate in this work.

2. RELATED WORK

With the miniaturization of digital cameras and their widespread inclusion in mobile devices, text extraction from scene images or video has received a fair amount of attention recently.

Some of the published work most closely related to our research include Watanabe *et al.*,⁴ which presents an automatic translation tool, targeted at translating Japanese text extracted from a picture. The presented system runs on a PDA equipped with a digital camera. It is not fully automatic, but requires the user to specify the area of the image he wants to translate. Also, some strong assumptions are made: the background is supposed to be more or less uniform, and the text fonts should not vary. The authors later implemented their system on a cell phone,⁵ using Java technology. In this implementation, only the text region selection runs on the cell phone. After this step, the cell phone connects to a server, and sends it the image region to be translated.

A similar approach is developed by Haritaoglu,⁶ in which a scene text extraction and translation system for PDA is introduced. The user takes a picture of a scene containing some text in Chinese and selects the rectangle region he wants to translate. Due to memory and computing power limitations of the PDA, the computation is not performed onboard, but the handheld device connects to a back-end server via GSM modem and sends the picture. The server performs text translation to the user's native language and replaces the original text in the image.

Pilu and Pollard⁷ developed a light-weight system performing text extraction from scene pictures. The focus is given on a fast implementation, such that the algorithm can run entirely on a PDA. The system is demonstrated as a stand-alone application of text extraction with translation, that runs on a handheld device.

Automatic translation is used in a slightly different way in Yang *et al.*^{8,9} The authors introduce an application capable of performing automatic sign translation from video. A first component identifies the text regions in the video and an interface lets the user choose the area he wants to translate. Then another module performs standard text extraction, recognition and translation and feeds the result back to the user. Chen *et al.*¹⁰ refined the application, such that user intervention is no longer needed.

Compared to the approaches described above, our method has a number of advantages. First of all, it is completely implemented on the server side, and relies on widely adopted communication channels. This has the combined benefit of making our system readily deployable on virtually all mobile phones in use today, while requiring absolutely no software installation on the client side. Second, our method is entirely automatic, and requires no user intervention other than taking a picture and sending an MMS. Finally, our system is made of several interoperable modules, which makes it both easy to upgrade by changing one particular module, and simple to deploy in a cloud architecture to obtain high performance and face strong user demand.

3. SYSTEM

As illustrated on Fig. 1, the translation process is initiated by a user sending a picture by MMS (a). This MMS is received by the back-end server, the image extracted (b) and passed to the text extraction module. This latter module attempts to isolate text elements from the rest of the image. It produces a binary image (c) along with the coordinate information of

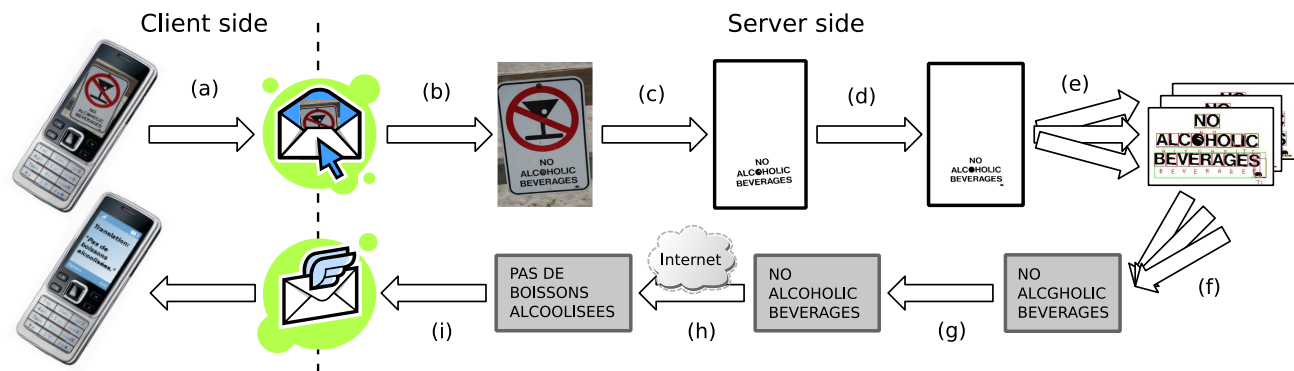


Figure 1. System overview. A user takes a picture of a text and sends it as an MMS (a). The server receives the MMS, extracts the image content (b) and performs text extraction to isolate the text region (c). Skew correction is applied if necessary (d). Then several OCR engines are invoked in parallel (e) and their output is merged (f). Spell checking is then applied (g) to correct potential OCR mistakes. The resulting text is sent to an online translation engine (h), and the final translation sent back to the user in an SMS (i).

every detected character. The next step in the process is image deskewing (d). This system estimates the angle of the detected text and rotates the image accordingly, in order to present as clean an input as possible for the optical character recognition.

The text image deskewed is then passed to several OCR programs (e), each of them producing character guesses. All guesses are merged by an OCR combiner program into a unified text (f). Before being sent for translation, the retrieved text is processed by a spell correction module (g), in order to correct the possible mistakes made by the OCR. Finally, the text is submitted to an online translation tool (h) and the result is sent back by SMS (i) to the user who requested the translation.

In the remaining of this section, we describe each of the components of our system in greater details.

3.1 Image preparation

Currently available OCR algorithms have been developed to deal with paper printed documents. Typically, OCR engines process documents that have been previously scanned by a flatbed scanner, and expect them black and white, relatively clean and well structured.

The problem of text recognition from natural images is much less constrained. Not only is the image in colors and cluttered with non-text objects, some of which might however present text-like textures, but the text itself can take very different shapes, colors, and fonts. Moreover, it is likely to be affected by perspective and lighting effects as well as compression artifacts. To make things even worse, a standard text structure cannot be expected; thus finding text location within the image might represent a challenge by itself.

In these conditions, the direct application of an OCR engine on a scene image containing text would result in an almost certain failure. Therefore, the first task in the server-side pipeline consists of processing the picture to transform it in a way that current OCR engines are able to manage.

3.1.1 Scene Text Extraction

One of the fundamental elements in our pipeline is scene text extraction. Its goal is to identify and isolate text elements in a picture. Scene text extraction is designed to operate on raw images, and produce binary versions, in which all non-text elements have been removed.

Our approach to scene text extraction is illustrated by Fig. 2. After applying a convolution with a Gaussian kernel to the input image (a), in order to get rid of the possible noise and compression artifacts, we perform edge detection using Canny's method¹¹ (b). This result is then thresholded to yield a binary edge image, and the end points of open shapes are merged into closed shapes, based on the gradient of their linking path. At this stage, the binary edge image is segmented into individual glyphs, by performing connectedness analysis (c). The glyphs thus retrieved are then analyzed one by one, and those less likely to represent characters are rejected, according to a set of heuristics based on size, size ratio, and topological constraints (d).

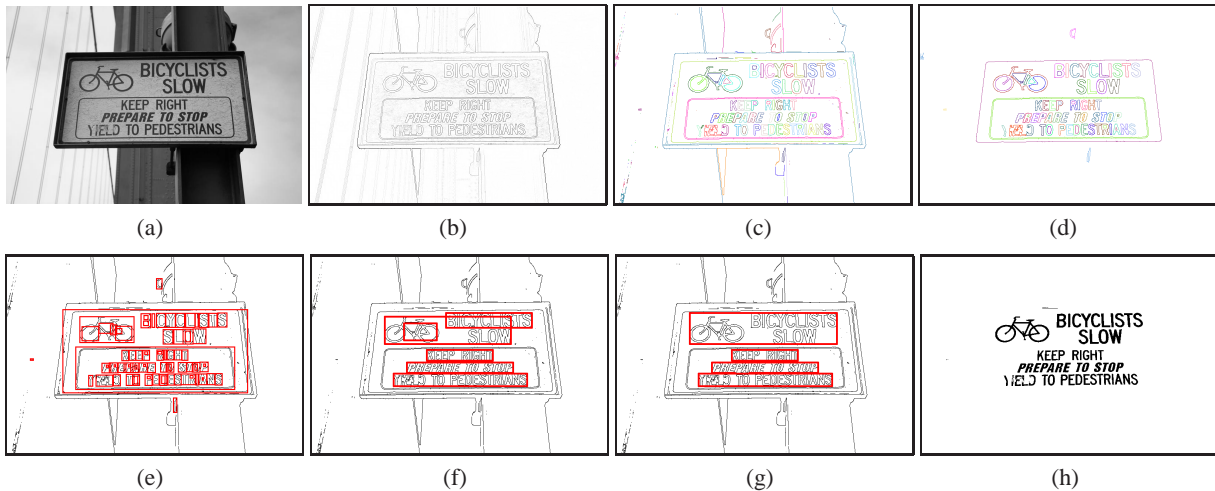


Figure 2. Text extraction steps. The input image is converted to grayscale (a). We perform edge detection (b) and threshold the result. Glyphs are then segmented (c) and filtered with a set of heuristics designed to identify character-like shapes (d). Bounding boxes are determined for the resulting individual glyphs (e). Those are then clustered into lines (f), which are later merged (g) based on their topology. Finally, the selected area of the image is converted to binary (h). Images (c) to (g) are best viewed in color.

For every glyph accepted so far, a bounding box is determined (e). The boxes are then clustered into words, and the words grouped into lines (f), discarding isolated boxes that do not fit into the structure along the way. We end up with a set of bounding boxes covering the detected lines of text in the image (g). The final image is then generated by extracting the content of the original image in the areas covered by the line bounding boxes (h), and for each one of them, finding the appropriate support, in order to deal with text in inverse video. Typical text extraction results as well as failure cases are shown in Fig. 3.



Figure 3. Text extraction results. The first row shows examples of successful text extraction. Note the correct extraction, even though the text is lighter than its background. The second row illustrates two typical extraction failure cases: small drawings can be mixed with characters (left) and Asian characters are not handled well by our system, which is targeted at western writings (right).

3.1.2 Image Skew Correction

Even transformed by scene text extraction into binary version with only text elements, the images may not yet be in an appropriate format for OCR processing. At this stage, images might still be plagued by rotation or perspective effects, to which most traditional OCR engines are very sensitive. Additional post-processing is thus needed to expect satisfying OCR

results. In this work, we concentrate on rotation correction, and do not address the problem of perspective deformation. This shall be considered in future work, in order to make the system more robust to any kind of deformation.

Here, we base our reasoning on the fact that a large majority of texts written in western countries are aligned horizontally. Therefore, OCR engines expect characters aligned to the document in which they appear. However, user pictures, especially those taken with a cell phone camera, are rarely perfectly level. Our goal is thus to determine whether the glyphs found by the text extractor form an angle with the horizon, and, if they do, to compute this angle α . We can later rotate the image by the inverse angle $-\alpha$, in order to align the text horizontally. Note that we do not apply the rotation directly on the binary images, as this would alter their quality significantly. Instead, if the rotation is needed, we perform it on the original input image and re-apply text extraction.

Assuming that, in most cases, characters are aligned horizontally into words, we compute the angle between any couple of glyphs identified by the text extractor and store the result into a histogram. Because of the horizontal alignment, the number of angles between couples of glyphs located on the same line should be significantly higher than other various angles between couples on different lines. Thus, we expect to observe a well defined peak in the histogram bin corresponding to the tilt angle. As shown by results of Fig. 4, this is the case in most images.

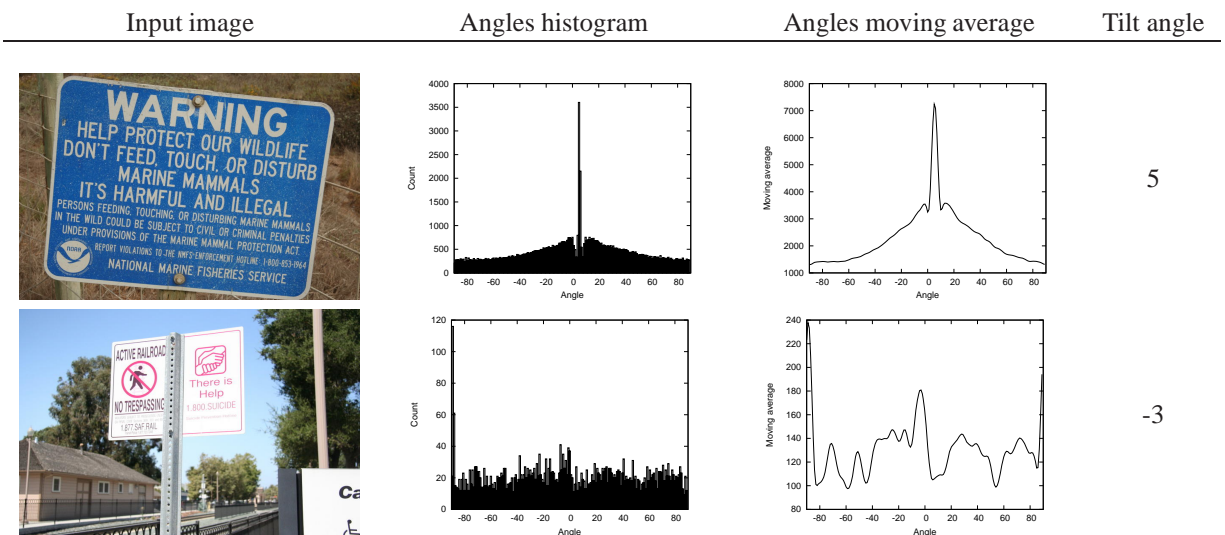


Figure 4. Text deskewing results. On the first row, the large number of characters makes it fairly easy to determine the tilt angle. The histogram in the second row is noisy, due to the limited number of characters and to some false positives from text extraction. In this case, the corresponding moving average becomes very useful to identify the tilt angle.

In images with a small number of characters, or on those where text extractor produced a significant amount of false positives, the histogram might be particularly noisy, making it difficult to determine the tilt angle of the text. For those cases, we additionally compute a moving average on the histogram values, with the effect of reducing noise and thus enhancing the results. The usefulness of this method is illustrated on the second row of Fig. 4, in which the moving average made it possible to infer the horizontal angle, which would have been impossible using the histogram only.

If the angle found this way is higher in absolute value than a given threshold (2 degrees, for our prototype), the original image is rotated by the inverse angle value and text extraction is repeated on this new image.

3.2 Optical Character Recognition

Optical Character Recognition has been a very active area of research for several decades and as a result many different engines have been implemented, some of which are available in the public domain. For this reason, we chose to rely on an already tested and solid implementation.

Although OCR on scanned documents is generally considered as a solved problem, our input images, even corrected, are still not close to the ideal binary image coming from a flatbed scanner. Thus, we cannot expect as good OCR results on our images as they would be on scanned documents. In addition, the cloud services architecture lends itself easily to

configuration of web based services. Thus, we can rely on multiple OCR services to generate a stronger combined result. If new services appear that generate better results, those can be configured into the system quickly.

These reasons motivated our design choice to rely on multiple OCR engines. For this system, we opted for the three following open source OCR engines: Tesseract,¹² developed by HP in the 1980's, released as open source in 2005 and currently supported by Google, Ocrad¹³ maintained by the Free Software Foundation and GOCR.¹⁴ All engines take a binary image as input and produce various output formats, ranging from complex XML with character locations and probabilities, to simple text-only output. Multiple OCR engines are integrated into our system this way: We simply feed the corrected image processed by the text extraction and skew correction modules to every OCR engine and obtain as many output as there are engines. Ideally, all engines should have correctly recognized characters, yielding identical outputs. In reality, such a good coordination is rare, and we most likely have to post-process the OCR results and merge them into a unique answer, before we can proceed with the translation.

3.2.1 Combining OCR results

Combining OCR results is only feasible if they exhibit an equivalent output format. The first task we address here is therefore the unification of the multiple output formats into a common XML one. Since we use open source software OCR engines, we were able to get access to their source code and modify them easily. After modification, every OCR engine produces an XML output containing a list of bounding boxes for every detected character, along with a character guess and sometimes a second guess. A confidence score is also provided for each guess.

With the output format unified, results from the different engines can be compared. First of all, overlapping detections are merged. Then characters are clustered into words and words into lines. Isolated and low confidence character detections are dropped. At this stage, we integrated the spell checker method described in §3.3 into the combiner, in order to produce meaningful results. Thus, when character detections differ among engines, their confidence score and the spell checker hints are taken into account to solve the conflict. The final merged result is produced in text-only format.

3.3 Spelling Correction

Despite the usage of several different OCR engines, the combined OCR result is likely to still contain mistakes, especially when the input images are of low quality. Those errors are usually not a big challenge for humans, because they often consist of mixed visually similar symbols, such as 'o' and '0' (zero), or '1' and 'l' (one). Even spoiled with those errors, a text is still easily readable by a person.

However, the situation is very different when dealing with a translation engine. Those simple OCR mistakes have a strong impact on the translation performance, because misspelled words will not be matched in the translator's dictionary, and often ignored.

To address this particular problem, we propose the application of a spell correction program to the output of the OCR engines. Classical spell correction software is typically able to check if a word belongs to a dictionary, and otherwise suggest a list of possible replacements, sorted by occurrence probability.

In this work, we use the open source software spell checker Aspell,¹⁵ which is available both as a stand-alone program and as a library. It is widely used in text editing software and has a strong reputation at suggesting meaningful replacement for misspelled words. We decided to merge the spell correction step directly into the OCR engines combiner program. It makes much sense to do it at this stage, because of the confidence score provided for each character by the OCR engines. Using both the OCR confidence information and the replacement probability provided by Aspell, we can take better decisions while merging OCR results, as explained above.

The spell checker is also used during the merging process to help split long words that have been mistakenly merged by OCR engines or merge adjacent words that were wrongly split during character recognition.

The use of a software like Aspell to correct spelling mistakes made by OCR is not without its problems. Most spell checkers have been primarily designed to correct wrong human spelling, which turns out to be quite different from OCR mistakes. Human spelling mistakes are often the result of pressing the wrong keys of a keyboard, thus replacing a letter by another one closely located on the keyboard layout. Other common human spelling errors are wrong sequences of letters, for example writing 'teh' instead of 'the'. As discussed above, OCR rather tends to replace a character by another, which is visually close. Also OCR frequently glues two characters together – 'ot' could become 'd' – or misinterpret a small drawing for a character.

Therefore, the replacement suggestions made by Aspell, are not always adapted to the circumstance, and the distance score it provides with every replacement does not perfectly match our target application.

Nevertheless, applying a generic spell checker like Aspell still helps improve the performance in most cases. The use of specific dictionary-based spell checker,¹⁶ designed for correcting OCR mistakes, would definitely yield better results, and will be addressed in a future version of our system.

3.4 Translation

Up to here, all the efforts have been dedicated to extracting the text content from an image taken by a cell phone camera, and getting it as close as possible to the original. The final step, before sending the answer back, is the translation itself. To perform this task, we decided to rely on one of the numerous on-line translation tools. Those are attractive, because they are freely available and constantly updated for better performance. The drawback of this choice, as will be seen later in this section, is that they are usually very generic and designed to translate full texts. For our application, we selected Google Translation and Babel Fish Translation (Yahoo). They both provide free and reasonably good translation from English to the major other European languages and back. As can be deduced from the results of Fig. 6, we built our application for translating English into French. Of course, as long as we are dealing with languages using western characters, our system can be easily modified to deal with any couple of languages supported by the translation engines.

Using online translation engines in an automatic process is not straightforward, because they are built as websites and designed mainly for an interactive user experience. To be able to automate the process, we have to format our `http` requests to look as if they were originating from a web browser. We perform this task thanks to the open source library `cURL`,¹⁷ which makes it possible to create well formatted `http` requests and manages to send them to a web server and collect their answer.

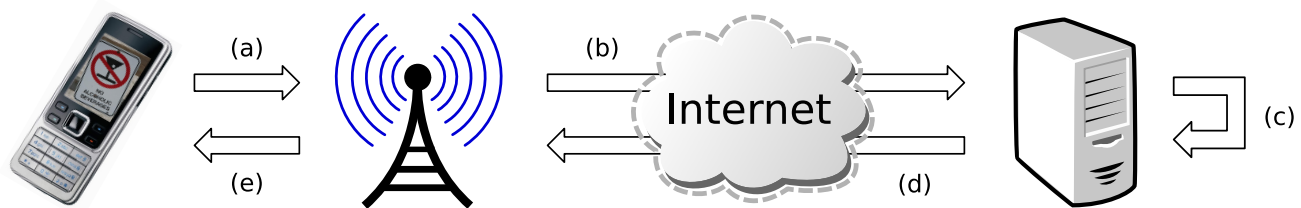


Figure 5. Network overview. The user picture is sent via MMS, whose destination is an email (a). The cellular carrier receives the MMS and forwards it (b) through Internet gateway or SMS aggregator to Internet. The email is then received at the back-end server and the translation is carried out (c), after which the server replies to the cell phone via email (d). Finally, the cellular carrier receives the email destined to the handset and converts it to SMS (e).

Using a web translation engine is a solution we chose out of convenience, in order to get a working prototype quickly deployed. In a commercial application, the use of a more specific type of translation tool would be advised. First of all, the format of the web pages can change without notice, thus requiring the adaptation of the script feeding the text in original language and extracting the translation. Second, those on-line translation solutions are very sensitive to misspelled words and to incorrect or incomplete punctuation. When a word is not found in their dictionary, it is usually copied ‘as is’, and no further effort is made. Third, their dictionary is very generic and fails to translate some specific expressions. For an application like ours, where input text often comes from public signs, the use of a dictionary more adapted to the context could improve the general performance. Finally, those translation engines are primarily targeted at translating whole and well formed sentences. However, in a practical application of camera phone translation, users are much more likely to translate small text originating, for example, from road signs. Those often consist of very simple sentences, with missing elements, such as verb, or subject. An example is depicted in Fig. 7. The fourth row, for instance, shows a simple text that has been perfectly OCRed, but for which translation is however imperfect, due to the simplistic sentence structure.

3.5 Network

All the processing we have discussed is implemented in the “cloud”, which consists of a series of network connected back-end servers. As introduced in §3, the image to be analyzed is sent to the server by MMS. The translated text is then sent back to the user in a SMS.

Dealing with MMS represents a challenge, as there does not exist any well accepted standard for their format and transmission. In practice, every telecommunication carrier has adopted a different method. Some, for example, embed the transmitted image in the message. Other just send an URL, and it is up to the receiver to fetch the image from a web server. In all cases, the message is sent in MMS format between the cell phone and the telecommunication company, and then converted to e-mail and transmitted to our back-end server. Fig. 5 illustrates this process.

An important task of our framework is therefore to extract the image sent in all possible MMS formats. For our application, we support MMS formats of the major telecommunication operators in the US. Once the translation achieved, our system also has to generate an SMS response.

The mechanics of transporting the image to the back end server, and the result SMS message back to the handset are accomplished via the email/SMS gateways provided by telecommunication carriers, or SMS/MMS aggregators. In either case, the desired translation is indicated via a specific destination for the request. In the case of email MMS, it would be `language@service.com`.

4. DISCUSSION

Here we start by analyzing the results of our application, as well as its performance. We then concentrate on the benefits of the client-server framework developed in this article, and discuss other potential applications taking advantage of it.

4.1 Results

Our back-end system was implemented on a 2.8 GHz Linux machine, acting as a server. The Postfix Mail Transport Agent accepts incoming email, and routes requests for translation via mail aliases handled by a single Procmail Mail Delivery Agent shell script. That script sequentially launches all the necessary processes involved in image extraction, text extraction and translation when an email is received. All the different programs dealing with the extraction process have been implemented in C++ using open source libraries. They can all be invoked as command line tools. Image extraction from a number of cell carrier MMS/email gateways is accomplished via a Perl script, while result email is generated and sent via a PHP script. The main script, sequentially invoking all the modules on the back-end server, is written as a Bash script.

To the best of our knowledge, there is no publicly available image database for scene text extraction and translation benchmarking. Related work typically base their evaluation on self built image databases. For this reason, we have created our own scene text image database, comprising a wide range of scene images, taken with different models of cell phone, as well as other cameras, and by several users during their daily activities. We gathered a total of 65 images mostly containing text from signs or advertisements, often mixed with non-text elements. Typical examples are shown in Fig. 6 and 7.

To obtain a quantitative measure of our system's performance, we applied it to all images in our database, and counted the number of successful text extraction. We do not evaluate the performance based on translation, but rather on text extraction, because the latter makes it easier to set up an objective criteria for success, while the former is much more ambiguous. Moreover, since we are relying on online translation engines, we have no control on their quality. We consider as successful the texts containing 90% or more of the characters retained by a human-generated ground truth. Out of the 65 images in our database, 59 were extracted successfully, according to the criterion described above, which corresponds to a rate of 90.8%. This performance is therefore comparable to related approaches.^{5,6} However, as opposed to monolithic systems, our approach is highly modular, making it extremely easy to upgrade any module for a more efficient one.

Because of the chosen architecture consisting of several individually connected components, the system can suffer from several types of errors, depending on the module that failed. Examples are shown on Fig. 7. In (a), the text extraction failed, due to the special font used, which is made of several smaller shapes. The text extraction of (b) was correctly performed, but the OCR failed, probably because of the special fonts, with changing width within the same word. In (c), text extraction and OCR did a decent job; however, the spell checker failed to correctly replace the word misspelled by the OCR and came up with a word with close spelling, but completely different meaning. Finally, in example (d), the text extraction and recognition modules performed correctly, but the translation engine was not able to translate correctly this incomplete sentence. These few examples clearly demonstrate that if one component fails on a query, it is very likely that the whole process yields a wrong or very approximate translation. Therefore, in order to improve the performance of our system, it is important to address shortcomings of every constituent module.



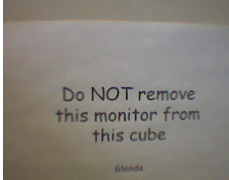
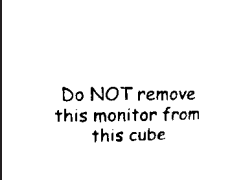
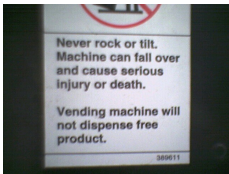
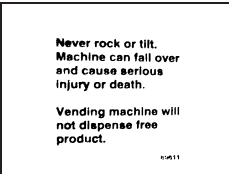

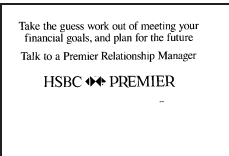
Original	Extracted text	OCR result	Translated text	Comments
		You can be prosecuted for drinking alcohol in public in this area. Maximum Penalty £500	Vous pouvez tre poursuivi pour boire de l'alcool dans public dans ce domaine Peine maximale f500	Correct text extraction. Small mistake in translation.
		Do NOT remove this monitor from this cube	Ne pas enlever ce moniteur de ce cube	Correct extraction and translation.
		Never rock or tilt. Machine can fall over and cause serious Injury or death. Vending machine will not dispense free product.	Ne jamais basculer ou incliner. La machine peut tomber plus d'et causer des dommages ou la mort sérieux. Le distributeur automatique ne distribuera pas le produit libre.	Correct extraction, small mistakes during translation but general meaning is there.
		Take the guess work out of meeting your financial goals, and plan for the future. Talk to a Premier Relationship Manager. HSBC PREMIER	Prenez la conjecture de votre reunion objectifs financiers, et de planifier pour l'avenir Parlez-en un Premier Relationship Manager HS BC PREMIER	Correct extraction, translation not perfect due to the absence of punctuation.

Figure 6. Results of our translation system on various images representing situations from everyday life. The first, third and fourth images were taken by cell phone cameras. The other images were shot by a digital camera and later down-sampled.

4.1.1 Speed

We built our application mainly as a prototype, and speed was not the main concern during development. Once a message is received on the server, text extraction and translation takes from 2 to 10 seconds, depending on the resolution and complexity of the image. Most of this time is spent by the OCR engines. Since we use external translation services, we have no control on the speed of those applications. However, our experience shows that they are usually quite fast and only account for an insignificant part of the time taken by the whole process.

User experience would not be correctly evaluated should we not consider the network delay, including the transmission of the initial MMS and the translation SMS. This delay depends on many factors, such as the carrier, the state of the network, the power of the phone, the technology used for transmission (2G, 2.5G and 3G), etc. In our experiments, we found that the network delay greatly varies from one trial to another, but is usually comprised between 10 and 30 seconds. Thus, this brings the total user waiting time for translation from approximately 15 to 40 seconds. Note that the phones used during our experiments were 2G and 2.5G. 3G networks are typically much faster, so we can reasonably expect significantly lower delays with this technology.

This performance is acceptable for our prototype application. However, should the framework be deployed commercially, the application server is likely to receive hundreds of requests per second and some work should be devoted to speeding up the back-end process. Since we use a very decoupled architecture, replacing one component of our system by a more powerful one is straightforward. This property would make it also much easier to perform load balancing between different machines.


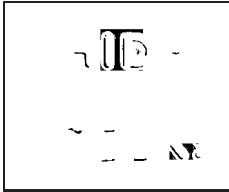






	Original	Extracted text	OCR & spelling	Translation	Comments
(a)			- Hy	- Hy	Text extraction failed due to fancy font.
(b)			Kass I GLOBING C H's I	L de Kass I GLOBING C h	OCR failed because of the font with changing style and size.
(c)			No SNOWING	Aucune CHUTE DE NEIGE	OCR found 'No SNOKING', which was wrongly corrected to 'No SNOWING' by the spell checker. Translation is correct.
(d)			NOT A THROUGH STREET	PAS A PAR LA RUE	Correct text extraction, but wrong translation (NOT 'A' THROUGH THE STREET).

Figure 7. Failure cases of different components: (a) text extraction, (b) OCR, (c) spell correction and (d) translation.

4.2 Framework

The framework we have chosen for our mobile translation application has many benefits over other approaches to similar systems. Our entire system is implemented in the back-end, meaning that the complete text extraction and translation is performed there. Moreover, the communication channels we chose (MMS and SMS) are very widespread. Virtually every cell phone and most PDAs used today are able to use those services. For instance, statistics³ in Japan show that in 2005 already, 70% of mobile phones customers subscribed to MMS service.

As a consequence of this choice of architecture, every cell phone equipped with a camera and able to send MMS – that is, almost all cell phones in usage today – can use our translation service, with no restriction on brands, models, carrier or technologies. Moreover, absolutely no software installation is required on the client side. The only thing a potential customer needs is the address of the translation service.

Another big advantage in such a simple interface between client and server is that the system can be upgraded easily, without the need to notify clients. Furthermore, the application is not restricted to handheld devices, but any terminal capable of sending a picture can be supported. For instance, it would be very easy to extend the current functionality to email.

Also interesting is the fact that, as opposed to other approaches,⁴⁻⁶ absolutely no user intervention is needed. This way a user does not have to learn any new manipulation, but can instead rely on the functions of his cell phone, which the user is accustomed to.

The advantages of our server-side approach extend well beyond additional workflows and messages. By tying the user's camera experience to the back-end service, there can also be a personalization aspect. If the user so chooses, the service can be tied to the user's identity, and access to the user's history granted when the connection is made. Thus, a personalized

experience – automatic message translation, automatic website translation, location-based information, etc. – all can be provided, per the user's preferences.

4.2.1 Other Applications

The framework on which we built our mobile translation system is very generic and well suited for image-based mobile applications. We thus list here a number of potential applications, which would be well adapted to our framework.

Image-based search is an area which would perfectly fit into our framework. Image-based search approaches^{18,19} can relatively easily be converted to our platform, creating a system potentially deployable on all current mobile terminals. Along the same lines as our translation project, we could implement a language identification application.

Our application framework makes many other interesting applications possible, especially for retail and consumer situations. For example, a customer may want expert advice as to home décor selections of paint and interior textiles. The user can take pictures of the current wall covering and flooring and send them off for a color analysis. An SMS with the exact paint brand can be returned to the user quickly. The camera phone makes the *in situ* data collection very easy to do, considering that it is very hard to describe a color. Our mobile imaging framework has already been used to recommend cosmetics,²⁰ which can easily be extended to hair color and clothing.²¹ A unique capability of the system is to give personalized advice that is based on the image sent. Similarly, we can imagine health care and gardening applications as well.

Another emerging application that could benefit from *in situ* image-based services is security printed products, such as over the counter medications. A shopper could collect product interrogation and authentication information by reading the security marking on the package.²² Security printing includes the use of printed materials, such as 2D bar codes, that already have a useful function on the product (e.g. for point-of-sale). Additional information added to the packaging/labeling is used to provide a unique ID (e.g. mass serialization), copy-protection (e.g. a digital watermark) and/or tamper evidence (e.g. damage due to environment for environmentally-responsive inks), often simultaneously.

4.3 Future Work

Although our prototype application yields good results, there is nevertheless room for improvement, in order to make the whole translation process more robust to the numerous glitches that may appear at any moment of the process.

First off, the text extraction could be made more discriminant between small drawings and characters. False positives and false negatives rate could be decreased by applying more sophisticated scene text extraction techniques.²³

Our method for finding tilt angle of the text works well. However, in addition to rotation, real life images are often affected by perspective effects, because users are rarely perfectly facing the text they are photographing. Those are harder to detect than rotation, because one needs to look for vertical as well as horizontal alignment cues in the image. Nevertheless, OCR results could certainly be improved if perspective effects were corrected. Several research articles^{7,24,25} propose ways to tackle this problem.

Then, OCR results could be improved by dealing with a larger number of different engines. Some work could also be devoted to refining the output of the OCR combiner. Translation results would benefit if the combiner was analyzing the document structure and was able to parse the text into sentences accordingly.

Another area where improvement could have a direct impact on the translation performance is spell correction. The usage of a spell correction algorithm specifically designed for post-processing OCR output is likely to be more powerful than the generic spell checker we use in our prototype.

Along the same lines, replacing the online translation engines by a local one, calibrated to the target application would bring the combined benefits of increasing its speed and reliability, and improving results relevance.

5. CONCLUSION

In this paper we have presented an imaging based service for mobile phones that provides fast easy text translation services from casually taken photos. While we have focused initially on translation services, once text is extracted, many more services can be applied. The text can be used for general text based search engines, but also specialized searches on local information or tourist information, etc. Text and the images can be used for security, authentication, and verification from printed packages. The use of the system is quite simple: the source and target languages are specified in the MMS

destination. The system tolerates casually taken pictures as you would expect from consumers. We feel that this is an important step in increasing the number of services that are easily available to mobile users without requiring special devices or service providers. We expect to extend our image understanding system to other domains such as health care, or home décor color coordination. Each of these applications is intended for novice users seeking an expert opinion from camera phone images.

REFERENCES

- [1] Budde P. *et al.*, [Global Mobile Communications – Statistics, Trends and Forecasts], BuddeComm (February 2008).
- [2] Infotrends, [Worldwide Camera Phone Forecast: 2007-2012] (July 2008).
- [3] Kindberg, T., Spasojevic, M., Fleck, R., and Sellen, A., “The ubiquitous camera: An in-depth study of camera phone use,” *IEEE Pervasive Computing* **4**(2), 42–50 (2005).
- [4] Watanabe, Y., Okada, Y., Kim, Y., and Takeda, T., “Translation camera,” in [International Conference on Pattern Recognition], **1**, 613–617 (August 1998).
- [5] Watanabe, Y., Sono, K., Yokomizo, K., and Okada, Y., “Translation camera on mobile phone,” in [International Conference on Multimedia and Expo], **2**, 177–80 (July 2003).
- [6] Haritaoglu, I., “Scene text extraction and translation for handheld devices,” in [Computer Vision and Pattern Recognition], **2**, 408–413 (2001).
- [7] Pilu, M. and Pollard, S., “A light-weight text image processing method for handheld embedded cameras,” in [British Machine Vision Conference], (September 2002).
- [8] Yang, J., Gao, J., Zhang, Y., Chen, X., and Waibel, A., “An automatic sign recognition and translation system,” in [Proceedings of the Workshop on Perceptive user interfaces], 1–8 (2001).
- [9] Yang, J., Gao, J., Zhang, Y., and Waibel, A., “Towards automatic sign translation,” in [Proceedings of the first international conference on Human language technology research], 1–6 (2001).
- [10] Chen, X., Yang, J., Zhang, J., and Waibel, A., “Automatic detection and recognition of signs from natural scenes,” in [IEEE Transactions on Image Processing], **13**, 87–99 (January 2004).
- [11] Canny, J., “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8**(6) (1986).
- [12] Smith, R., “Tesseract OCR,” (1985-2008). <http://code.google.com/p/tesseract-ocr/>.
- [13] Free Software Foundation, “GNU Ocrad,” (2003-2007). <http://www.gnu.org/software/ocrad>.
- [14] Schulenburg, J., “GOOCR,” (2000-2008). <http://jocr.sourceforge.net/>.
- [15] Atkinson, K., “GNU Aspell,” (2004-2008). <http://aspell.net/>.
- [16] Dengel, A., Hoch, R., Hönes, F., Jäger, T., Malburg, M., and Weigel, A., [Handbook of Character Recognition and Document Image Analysis], ch. Techniques for improving OCR results, World Scientific (1997).
- [17] Stenberg, D., “cURL.” <http://curl.haxx.se/>.
- [18] Yeh, T., Tollmar, K., and Darell, T., “Searching the web with mobile images for location recognition,” in [Computer Vision and Pattern Recognition], **2**, 76–81 (June 2004).
- [19] Tollmar, K., Yeh, T., and Darrell, T., “Ideixis - image-based deixis for finding location-based information,” tech. rep., MIT CSAIL (2004).
- [20] Jain, J., Bhatti, N., Baker, H., Chao, H., Dekhil, M., Harville, M., Lyons, N., Sang, H., Schettino, J., and Süssstrunk, S., “Color match: An imaging based mobile cosmetics advisory service,” in [International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI)], (September 2008).
- [21] Bhatti, N., Baker, H., Chao, H., Clearwater, S., Harville, M., Jain, J., Lyons, N., Marguier, J., Schettino, J., and Süssstrunk, S., “Mobile cosmetics advisor, an imaging based mobile service,” in [International Workshop on Mobile Computing Systems and Applications (HotMobile)], (2009).
- [22] Simske, S., Aronoff, J., Sturgill, M., and Golodetz, G., “A comparison of thermal inkjet, dry electrophotography and liquid electrophotography,” *Journal of Imaging Science and Technology* **52** (Sept/Oct 2008).
- [23] Lim, J., Park, J., and Medioni, G., “Text segmentation in color images using tensor voting,” *Image and Vision Computing* **25**, 671–685 (2007).
- [24] Clark, P. and Mirmehdi, M., “Rectifying perspective views of text in 3d scenes using vanishing points,” *Pattern Recognition* **36**, 2673–2686 (2003).
- [25] Chen, X., Yang, J., Zhang, J., and Waibel, A., “Automatic detection and recognition of signs from natural scenes,” *IEEE Transactions on Image Processing* **13**, 87–99 (January 2004).