Mobile Image Recognition: Architectures and Tradeoffs

Jonathan J. Hull, Xu Liu, Berna Erol, Jamey Graham, Jorge Moraleda
Ricoh Innovations, Inc.
California Research Center
Menlo Park, CA
650-496-5723
hull@rii.ricoh.com

ABSTRACT

We argue that the most desirable architecture for mobile image recognition runs the complete algorithm on the mobile device. Alternative solutions that run the recognizer on a remote server will not be as desirable because of the delay between image capture and receipt of a result that can cause users to abandon the technique. We present a method for mobile recognition of paper documents and an application to newspapers that lets readers retrieve electronic data linked to articles, photos, and advertisements. We show that the index for a reasonable collection of daily newspapers can be downloaded in less than a minute and will fit in the memory of today's mid-range smart phones. Experimental results show that the recognition system has an overall error rate of less than 1%. We achieved a run time of 1.2 secs. per image with a collection of 140 newspaper pages on an HTC-8282 Windows Mobile phone.

Categories and Subject Descriptors

I.7 [Document and Text Processing]: General.

H.5.2 [Inf. Interfaces and Presentation]: User Interfaces

General Terms

Algorithms, Design, Human Factors

Keywords

Camera Phone, Document Retrieval, Visual Search

1. INTRODUCTION

Mobile image recognition provides a way for anyone with a camera phone to retrieve electronic data linked to physical objects. The recognition algorithm determines the object in a picture and retrieves data associated with that object. The result can be displayed on the device, emailed to the user, or stored in a log file. The choice of object, data type and action associated with it can enable a number of commercial applications.

Examples of mobile image recognition in the market today include the linking of book covers to reviews and purchasing options [1]. As device technology improves and bandwidth increases, the number of commercial applications will increase to include the linking of arbitrary content with larger and larger numbers of objects. Already, research results report the reliable recognition of a million objects [2]. Putting that technology in the hands of end users will provide a real-world point-and-click capability. Anyone with a camera phone will be able to retrieve and add information to the objects they encounter every day.

General mobile image recognition has already been applied to several commercial applications e.g. Google Goggles (www.google.com/mobile/goggles) which recognizes landmarks and logos and SnapTell (www.snaptell.com) which recognizes book and CD covers. An important class of object is the paper document. We use them every day and they have the desirable characteristics of requiring no power, being completely portable, and extremely low cost. However, they do not provide the instantaneous access to related information that a web page does. This to some extent has led to the decline in popularity of paper documents in spite of the fact that they can be used even when the user is not in front of PC.

We have developed several recognition algorithms that allow for the creation of links from nearly arbitrary parts of paper documents [3, 4]. Both text patches and blocks of image data can be associated with individualized information. Furthermore, the recognition algorithms reliably recognize the low quality images produced by most mobile cameras. This allows us to provide "clickable paper" that can be utilized almost anywhere. The user only needs a paper document, a camera phone, and a network connection. An alternative solution would use 2-d bar codes such as QR codes. However, QR codes take up valuable space on a page, are impossible to change after a document is printed, and disrupt its appearance. Moreover, visual search can index arbitrary locations within a passage of text. Many QR-Codes could be needed to provide the same capabilities, which is unrealistic.

Until now, the only system architecture that is practical is shown in Fig. 1. Individual frames are transmitted to a server where they are recognized and the results returned to the phone.

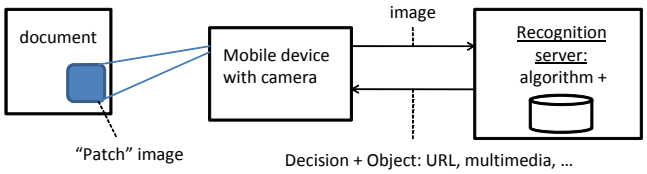


Fig. 1. Client-server architecture for mobile image recognition

HotMobile '10, Feb. 22-23, 2010, Annapolis, M.D.

This client-server architecture poses a serious problem to end-users. The time required to transmit a single image, recognize it and return the result to the phone can be long enough that a user may give up on the application. Or, if the first or second image they send to the server is not recognized, the user often thinks the application failed when in fact the application succeeded. It is the user that failed to send an image that could be recognized. In any case, the user is frustrated and stops using the application. To some extent this can be mitigated by improving the accuracy of the recognition algorithm, reducing the amount of data transmitted to the server, or including a pre-classifier that only transmits images that are likely to be recognized. However, inevitably some images will be rejected because it is impossible to have a database that includes images of everything a user can point their phone at. Also, there will always be some delay because of network latency.

This paper describes our experience with the architecture shown in Fig. 2 in which the database is on the phone and the recognition algorithm executes completely on the phone. Our client runs with the camera in video mode in which 5 to 10 frames are acquired per second. The user can move the phone smoothly over a document as video frames are captured and recognized. The fact that any particular frame is rejected does not negatively impact the user experience, because subsequent frames are processed until one of them is recognized. This eliminates the network latency and overcomes a significant source of user frustration. The database is periodically updated on the database generator and transferred to the phone. Since different documents could be recognized every day (e.g., newspapers), we propose that updates should occur sometime late in the evening.

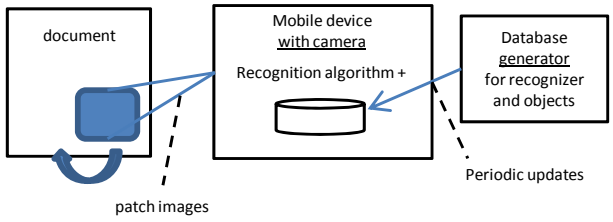


Fig. 2. Client-only architecture for mobile image recognition

An application scenario for our new architecture is shown in Fig. 3. In this case, multimedia content is made available to a mobile user by way of traditional paper media, a newspaper in this case. As the user moves the camera phone above the newspaper, successive image frames are recognized on the phone. When the page is identified, a thumbnail image of the document is displayed with "hot spots" (gray box in Fig. 3c) overlaid. Each hotspot is linked to some electronic content. If the user clicks on a hot spot, the electronic content is displayed (Fig. 3d shows a movie being played).

2. System Design

We developed an algorithm designed for the recognition of "patches" of text and images (2-inches square and larger) in two-dimensional planar document images and optimized it for a mobile device. The objective is to provide real-time interactivity between users and documents that makes each x-y location on a document with enough features individually addressable. Furthermore, as the speed of the recognition technology increases, the user will get the "point and click" experience of a bar code reader except that they can point at any location on a document and retrieve information customized to that location.

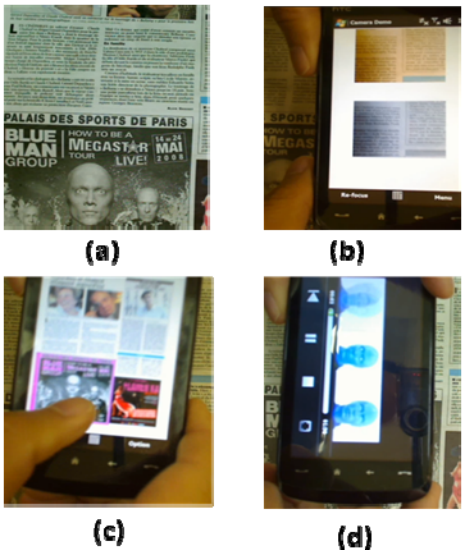


Fig. 3. Multimedia content linked to newspapers:
(a) A newspaper page indexed in the database
(b) User hovers above the page with a camera phone
(c) Successive frames recognized until document identified. Hotspots shown in purple.
(d) Multimedia content (a video clip) delivered to the user

We use the image recognition algorithm shown in Fig. 4 that identifies the locations of features and computes a feature vector around each location. We construct a 4x4 grid around each location and accumulate a weighted gradient orientation histogram with 8 bins in each grid cell (i.e., each bin corresponds to one of the eight compass directions). During a training phase, the feature vectors for document pages are stored together with an identification for the page and the x,y location where they occurred on the page. These (pageid, x, y) triples are stored in the terminal nodes of a tree data structure. To provide some redundancy, we store two copies of each triple at different terminal nodes.

At recognition time, the features in a "patch" image are matched to the tree data structure and we identify terminal nodes that represent the nearest neighbors to the input feature vector. The (page id, x, y) triples at those nodes are input to a geometric verification algorithm that identifies the page id that best accounts for the configuration of feature points. If this result has a sufficient number of inliers, we declare it as the decision that best matches the input image. Otherwise we reject the image.

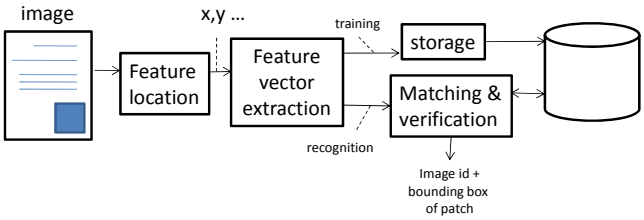


Fig. 4. Image recognition algorithm

The accuracy of the recognition algorithm was evaluated with an implementation on a desktop PC. We indexed 140 300-dpi French newspaper images each of which contained about 4000 features. We captured a test set of 513 patch images on a mobile phone at 1200x1600 resolution and down sampled to 400x533. The patches were chosen from various regions on the newspaper and were classified into five groups: 1. text (190 images); 2. photos (136 images); 3. headlines (41 images); 4. warped and shadow (44 images); 5. not in the index (102 images). An example of an image in each category is shown in Fig. 7.

The results in Table 1 show that the algorithm works well on the content it was designed for – text patches were almost always recognized correctly. Performance dropped for photos (85% correct, 0% error), headlines (88% correct, 5% error), and warped images (71% correct, 2% error). However, this was to be expected. A particularly satisfying characteristic is the algorithm's reliability – it almost always rejected patch images selected from documents not in the index.

	Text	**Photos**	**Headlines**	**Warped**	**Not in Index**
N	190	136	41	44	102
correct	99%	85%	88%	71%	0%
reject	1%	15%	7%	27%	99%
error	0%	0%	5%	1%	1%

Table 1. Recognition accuracy

3. Phone-based implementation

The complete recognition algorithm is implemented in C on an HTC 8282 Windows Mobile phone. Indexed documents are represented with binary tables that are designed for fast download. We tested the current version with the same collection of 140 French newspaper pages that were used for accuracy testing. Each indexed page contained approximately 4000 features and 23.8 MB were used for the index tables on the phone. With patch images captured on the phone at 320x240 resolution, 1.2 secs. are needed to recognize each image. Of this, 300 ms are needed for feature extraction. While the 1.2 secs. is more than the ideal "real time" of the 15-per-second frame capture rate, it significantly improves the user experience since there is no communication delay between recognition attempts.

The effect of the client-only architecture on efficiency was evaluated with a user study. Five members of our lab were given a newspaper that was in the 140-page index on the phone. The subjects were between 28 and 40 years old and had at least one smart phone so they do not have a learning curve to operate the software. They were told to choose any page and try and "recognize" the page by moving the camera phone over the page as the recognition algorithm tried to recognize successive image frames. The recognizer stopped only when it identified the page in view.

Of course, most users were cooperative and tried their best to make it work. They quickly learned the distance from the page that results in successful recognitions. For example, a user might try to recognize a photo close up and the application was not able to make a decision because the camera was too close. But, simply moving away from the page an inch or two was sufficient to recognize the photo. There was a short learning curve for this behavior and users quickly became better operators of the recognizer.

The number of frames that had to be recognized before a decision was reached is shown in Fig. 5. Out of the 102 recognition "sessions" (about 20 per subject), on average about 8 frames were processed before a decision was reached. However, the curve is highly skewed with most sessions needing between four and six trials. It's important to examine the tail of the curve where we see that in some cases up to 25 frames were processed before the decision was reached. We speculate that in a client-server system, most users would have given up after two or three tries and they would have missed most of the value of the application. Indeed, it's hard to imagine anyone trying to recognize anything 25 times. The client-only version of the application easily allows them to keep trying since they only need to move the camera over the page.

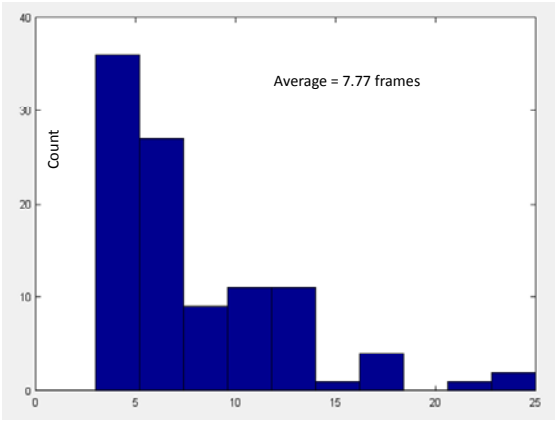


Fig. 5. Number of frames processed for each recognition

The total clock time needed for a user to recognize a document is shown in Fig. 6. We see that on average each of the 102 recognition sessions needs about 6 seconds. This is 33% less than the 9.3 seconds predicted by the product of the average number of frames (7.77) and overall runtime (1.2 seconds per frame) because users seem to get better at operating the recognizer over time and they intuitively learn type of image that the recognizer prefers (in focus with many features, a certain distance from the page) on their own after some experience. This is reflected in the grouping of many of the total run times less than the average.

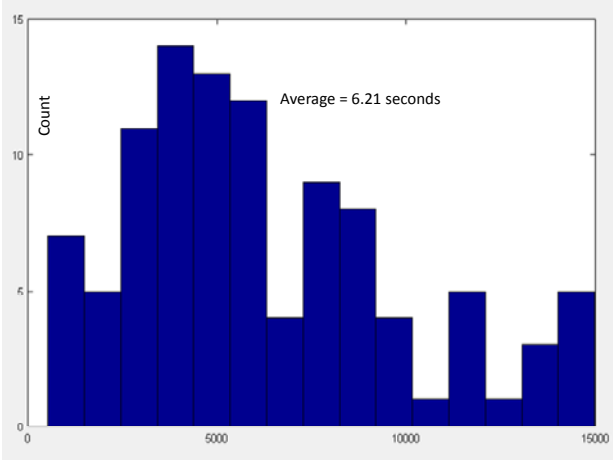


Fig. 6. Time (ms) for each successful recognition

4. Database Update

The client-only architecture assumes there is a database on the phone. The size of the database is determined by the number of objects that need to be recognized and the number of features they contain. An important part in user acceptance of this technology will be the amount of time required to update the database on the phone and the frequency of updates. Since the update time is determined by the size of the database and the application being supported, we assumed an application in which newspaper images are recognized and that the user would be comfortable with a once-daily update of the database, perhaps occurring in the early morning when the phone is charging.

The number of objects or newspaper pages in this case is determined by the newspapers that a user would read in one day. We estimated this by counting the number of pages in four local newspapers on Wed. Oct. 14, 2009. The New York Times had 62 pages, the Wall Street Journal had 58 pages, the San Jose Mercury News had 56 pages, and the San Francisco Chronicle had 52 pages. In total, these four publications had 228 pages or an average of about 57 pages each.

The part of the database most affected by the addition of new pages is the list of page id numbers and x,y coordinates associated with the features in the indexed images. The size of this list was estimated by indexing the front pages of between 30 and 300 different newspaper front pages. This includes a sampling of newspapers from across the U.S. and some foreign cities. Each pdf was rendered as a 300 dpi png image and run through our indexing algorithm. The results in Table 2 show the number of (page id, x, y) triples for each set of page images. We see that 250,091 triples are needed for 30 newspaper pages and this increases to 2,634,925 triples for 300 pages. The amount of storage was estimated by assuming 16 bits for each page id and 8 bits for each coordinate. This gives a range of between 1.0 MB for 30 pages and 10.5 MB for the 300 pages. These are all very modest and are easily accommodated in the program memory of today's mid-range smart phones.

# pages	#xy pairs (x1000)	MB proj.	# pages	#xy pairs (x1000)	MB proj.
30	250	1.0	180	1,534	6.1
60	480	1.9	210	1,834	7.4
90	722	2.9	240	2,117	8.5
120	988	4.0	270	2,371	9.5
150	1,250	5.0	300	2,635	10.5

Table 2. Storage for point lists on a mobile device

The time required to download these databases on a 3G network at 2 Mbps varies from 4 seconds for 30 pages to 42 secs. for 300 pages [5]. When 4G is available with its 100 Mbps transfer rate, transfer time will be almost negligible: 80 ms for 30 pages and 840 ms for 300 pages. At these speeds it will be feasible to build applications that download databases to the phone at start-up time. That is, we will no longer need to install them overnight. The user will have almost instantaneous point-and-click capability for any publication they subscribe to.

An alternative system architecture that some might suggest would stream image frames or features to a server over the network. However, the upstream bandwidth is currently always a fraction of the downstream speeds we assume in Table 2. In 3G, 0.15 Mbps is the maximum we can expect. Furthermore, upstream communication quickly drains the battery on the device. Since battery technology is advancing much more slowly than bandwidth and processor power, it's likely that the client-only architecture proposed in this paper will be the preferred solution for real-time image recognition on mobile devices for the foreseeable future.

5. Related Work

In recent years several techniques have been introduced that can link nearly arbitrary locations on paper documents to electronic data without bar codes [3]. These methods convert configurations of image features to links. Nakai et al. [5] extracted affine invariant features from the bounding boxes around neighboring words; Liu and Doermann [6] used the orientation of triplet words together with shape coding to identify the patch; Erol et al. [4] encoded the structure of word boxes as the signature for document retrieval. However, all of these approaches assume the printed materials are in a Latin language and will probably fail when processing non-Latin languages. For example, Chinese characters are square and no unique feature can be extracted from their layout since they align both vertically and horizontally. We use a recognition technology that can recognize documents in any language without OCR.

Recognition algorithms such as bar code readers have been embedded on mobile devices for some time. Recently, with the expanded technical capabilities of mobile devices, augmented reality algorithms have been adapted to mobile devices [8].

However, these methods are designed for fixed vocabularies of objects. A recognition algorithm that runs completely on a mobile device but is constrained to text in Latin languages is described in [4]. The need for on-demand update of a system's database was used in a method for GPS-triggered building recognition [9]. In contrast, we describe a technique for the recognition of arbitrary content in documents and an architecture in which the database on the device can be periodically downloaded to the phone so it is synchronized with a collection of objects (newspapers) that change on a daily basis.

6. Conclusions

We presented an architecture for mobile object recognition that runs the complete algorithm on the device and thereby solves the problem that potentially dooms present-day alternatives: the latency between image capture and recognition. Our technique for near real-time interaction between paper documents and camera phones allows users to retrieve electronic information linked to nearly arbitrary regions on a page. Although our system has been thoroughly tested with paper documents including newspapers, magazines and books, the algorithm is not restricted to paper material. It can also recognize from any flat media including posters and E-Reader devices such as Amazon Kindle and Sony Reader. The recognition algorithm was shown to have a high enough accuracy that it almost never makes a mistake. Thus, users can point the phone at almost anything and it will return a result only when the object is an indexed document. The combination of high accuracy and near real-time response provides a new method of interacting with paper documents that seamlessly links paper documents with electronic information.

REFERENCES

[1] Commercial offerings that recognize individual images submitted from camera phones include *www.doog.mobi* and *www.snaptell.com*.

[2] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," Proc. of the IEEE CVPR, 2007.

[3] J. J. Hull, B. Erol, J. Graham, Q. Ke, H. Kishi, J. Moraleda, and D. Van Olst, "Paper-Based Augmented Reality," 17th Int. Conf. on Augmented Reality and Telexistence, Esbjerg, Denmark, Nov. 28-30 2007, 205-209.

[4] B. Erol, E. Antunez, and J.J. Hull, "HOTPAPER: multimedia interaction with paper using mobile phones," *Proc. of the 16th ACM Intl. Conf. on Multimedia*, Vancouver, Canada, 2008, pp. 399-408.

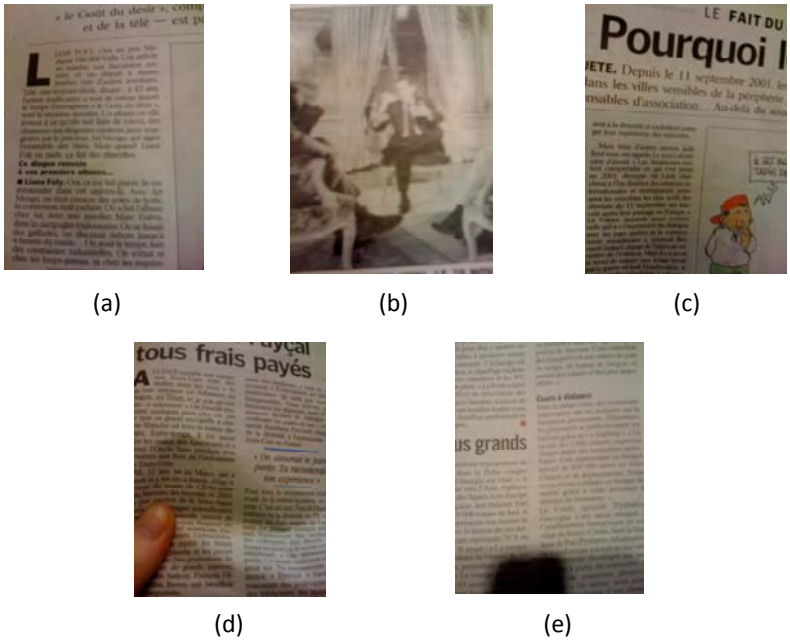
[5] S. Dekleva, J. P. Shim, U. Varshney, and G. Knoerzer, "Evolution and emerging issues in mobile wireless networks," Comm. ACM, v. 50, no. 6, June 2007, pp. 38-43.

[6] T. Nakai, K. Kise, and M. Iwamura, "Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval," *Lecture Notes in Computer Science (7th International Workshop DAS2006*, vol. 3872, 2006.

[7] X. Liu and D. Doermann, "Mobile Retriever: access to digital documents from their physical source," *International Journal on Document Analysis and Recognition*, vol. 11, 2008, pp. 19-27.

[8] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose tracking from natural features on mobile phones," Proc. of the 7th IEEE/ACM Int. Symp. on Mixed and Augmented Reality (Sept. 15 - 18, 2008), pp. 125-134.

[9] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," *ACM International Conference on Multimedia Information Retrieval (MIR'08)*, Vancouver, Canada, Oct. 2008.



(a) (b) (c)

(d) (e)

Fig. 7. Images in the recognition set: (a) "good" text, (b) pictures, (c) mixed headlines and pictures, (d) warped and shadows, (e) not in the database.