Currency Recognition on Mobile Phones

by

Suriya Singh, Shushman Choudhury, Kumar Vishal, C V Jawahar

in

ICPR 2014
(*International Conference on Pattern Recognition*)

Report No: IIIT/TR/2014/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
August 2014

Currency Recognition on Mobile Phones

Suriya Singh[1] Shushman Choudhury[2] Kumar Vishal[1] C.V. Jawahar[1]

[1]CVIT, IIIT Hyderabad, India [2]IIT Kharagpur, India

***Abstract*—In this paper, we present an application for recognizing currency bills using computer vision techniques, that can run on a low-end smartphone. The application runs on the device without the need for any remote server. It is intended for robust, practical use by the visually impaired. Though we use the paper bills of Indian National Rupee (₹) as a working example, our method is generic and scalable to multiple domains including those beyond the currency bills. Our solution uses a visual Bag of Words (BoW) based method for recognition. To enable robust recognition in a cluttered environment, we first segment the bill from the background using an algorithm based on iterative graph cuts. We formulate the recognition problem as an instance retrieval task. This is an example of fine-grained instance retrieval that can run on mobile devices. We evaluate the performance on a set of images captured in diverse natural environments, and report an accuracy of 96.7% on 2584 images.**

Keywords*—Instance retrieval, currency recognition, indexing and retrieval, mobile vision.

I. Introduction

Visual object recognition on a mobile phone has many applications. In this paper, we focus on the problem of recognition of currency bills on a low-end mobile phone. This is an immediate requirement for the visually impaired individuals. There are around 285 Million people estimated to be visually impaired worldwide, out of which 39 Million are blind and 246 Million have low vision [1]. The differences in texture or length of currency bills are not really sufficient for identification by the visually impaired. Moreover, bills are not as easy to distinguish by touch as coins. Certain unique engravings are printed on the bills of different currencies but they tend to wear away.

We adopt an approach based on computer vision on mobile devices, and develop an application that can run on low-end smartphones. We consider the bills of Indian National Rupee (₹) as a working example, but the method can be extended to a wide variety of settings. Our problem is challenging due to multiple reasons. We want all the computations to happen on the phone itself and this requires appropriate adaptation of the recognition architectures to a mobile device. (For example Panda *et al.* [2] carry out a scalable retrieval on a mobile phone by appropriately modifying a retrieval solution.) Since our application is desired to be usable in a wide variety of environments (such as in presence of background clutter, folded bills etc.), we need a robust recognition scheme that can address these challenges. Also, visually impaired users may not be able to cooperate with the imaging process by realizing the environmental parameters (like clutter, pose and illumination).

The problem of currency recognition using computer vision techniques has been studied in the past. Neural networks have been used for recognition [3], [4]. Hidden Markov Model has also been exploited using texture characteristics of the bills as a feature [5]. Local Binary Patterns (LBP) have been utilized as a feature, by [6], along with a two-phase classification scheme using template matching. Adaptive boost or AdaBoost, along with weak classifiers have been used by [7] and [8]. While most of the above work has shown high accuracy for classification, the test cases have usually consisted of scanned or carefully captured bills. These test cases lack variations in illumination, environment, texture and dimension. Our problem setting is similar to [9], which has considered a variety of imaging conditions.

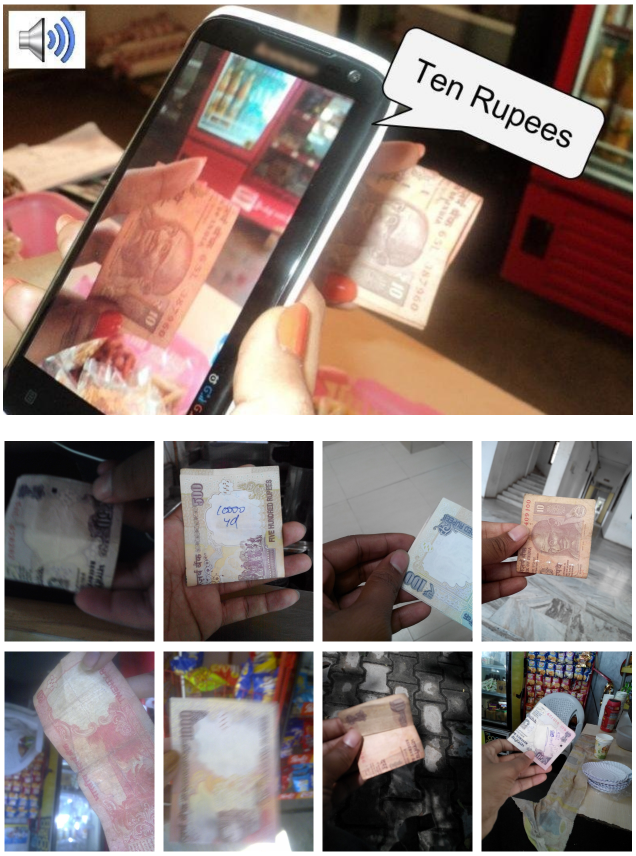


Fig. 1: The top row shows a typical use case of our currency recognition app. Our application recognizes a bill from an image with many distracting objects and speaks it out. The last two rows show images from the dataset for Indian National Rupee (₹), with bills in varying illumination and background.

Most of the previous methods formulate the solution as one which gets trained offline with enough positive and negative examples. However, our approach is based on the formulation as an instance recognition under clutter. We are able to use a thin index structure to make the application efficient and compact. Also note that our problem is considerably different from typical image instance retrieval (e.g. buildings). Firstly, the instances of interest are very similar to each other. The retrieval must be particularly discriminatory in choosing the correct result. Secondly, the real-world usage by the visually impaired introduces challenging queries in terms of the image quality, the portion of the bill visible, illumination and clutter. To deal with this, a fair amount of computation is spent on image pre-processing to reduce the effect of outliers. Also, an extensive database with bills of varying quality is compiled and used for instance retrieval.

Many of the methods discussed above (e.g. [3]–[6]) are intended for desktop systems; however there are commercial as well as non-commercial currency recognition applications available for mobile devices. Examples include LookTel Money Reader [10] and IDEAL Currency Identifier [11]. The LookTel Money Reader, though accurate, requires a high quality camera and a lot of light to properly recognize currency [10]. The IDEAL Currency Identifier requires the bills to be placed on a flat surface, in a horizontal position, in good lighting. It fails to recognize wrinkled and worn out bills [12]. These applications are computationally intensive and intended for high-end smartphones.

We have been motivated by a lot of work that has been done recently on computer vision for mobile phones, and its various applications. Robust detection and recognition of objects of interest has been one major area of study. Most of these applications use a server-client model and an Internet connection for communication. In this architecture, the client mobile system acts as the input/output device while performing minimal tasks [13]–[15]. Work that has involved processing solely on the mobile phone primarily focuses on robust detection and recognition of objects [2], [17], 3D reconstruction [18] and Augmented Reality. Applications have also been targeted for visually impaired users as well [19].

II. Design and Challenges

Working on a mobile platform brings with it a number of unique challenges that need to be taken care of. Primarily, the restrictions are in the memory, the application size, and the processing time. Currently, the average size of an iOS application is 23MB, while the RAM limit for a Windows phone application is 150MB. For an application to run on a mobile phone without affecting the others, it should not use more than 100MB of storage and 50MB of RAM.

Our application recognizes the bills in two major steps. First we segment the bill from the clutter. Then we look at the most similar bill in the database. Though both these problems can be solved with good performance using many state-of-the-art computer vision algorithms, they are not really mobile friendly. The recognition model and other necessary information for our application would typically require more than 500MB of storage and 200MB of RAM with a direct implementation. This exceeds practical limits by a large amount. To be practically useful, the application's response time should not be more than 4 seconds keeping in mind that the current average response time is 3.28 seconds.

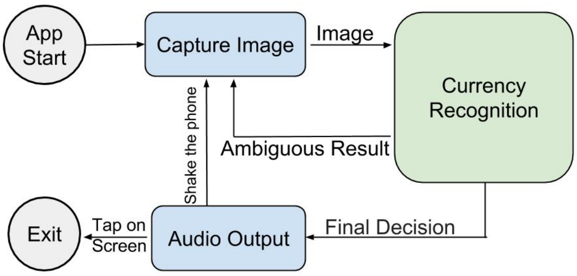


Fig. 2: A schematic of the high-level control flow diagram of the currency recognition application intended for a visually impaired user.

The target audience being the visually impaired introduces additional challenges. The user is unaware of the condition of the surrounding environment — other objects, lighting, contrast, and even whether the bill is properly placed in the field of view of the camera or not. The system should be robust towards a wide variety of images that are likely to be captured by the target user. Using the application should be simple and intuitive for a person who cannot see (refer to Figure 2 for the architecture of our solution). It should have a custom camera that once started requires no input from the user. In short, the problem at hand requires innovative modules that can recognize the bill in diverse environments reliably, robustly and efficiently.

III. Method

A. Segmentation

As illustrated in Figure 1, the images might be captured in a wide variety of environments, in terms of lighting condition and background while the bill in the image itself could be deformed. Image segmentation is important not just for reducing the data to process but also for reducing irrelevant features (background region) that would affect the decision-making.

We start with a fixed rectangular region of interest (ROI) which is forty pixels smaller from all four sides than the image itself. We assume that a major part of the bill will be present inside this region. Everything outside this ROI is a probable background. Once this region is obtained, it must be extended to a segmentation of the entire object.

Let x be an image and let y be a partition of the image into foreground (object) and background components. Let $x_i \in \mathbb{R}^3$ be the color of the i^{th} pixel and let y_i be equal to +1 if the pixel belongs to the object and to -1, otherwise. For segmentation we use a graph cut based energy minimization formulation. The cost function is given by

$$E(x, y) = -\sum_i \log p(y_i|x_i) + \sum_{(i,j)\in\mathcal{E}} S(y_i, y_j|x)$$

The edge system $\mathcal{E}$ determines the pixel neighborhoods and is the popular eight-way connection. The pairwise potential $S(y_i, y_j|x)$ favours neighbor pixels with similar color to have the same label. Then the segmentation is defined as the minimizer arg $\min_y$ $E(x, y)$. We use the GrabCut algorithm [20], which is based on iterative graph cuts, to carry out foreground/background segmentation of the images captured by the user.

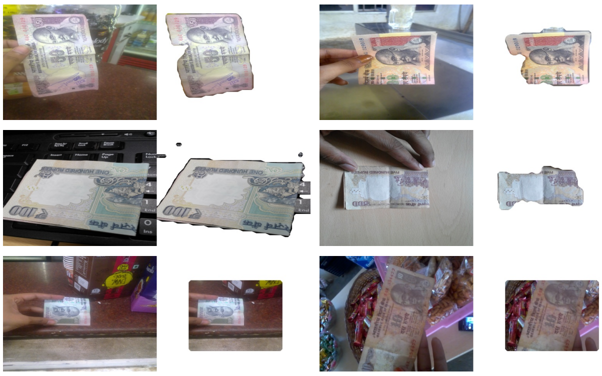


Fig. 3: Segmentation on a currency bill. The first and second rows show successful segmentation results while the last row shows segmentation failure where center region is marked as foreground.

The system should be able to segment the foreground object correctly and quickly without any user interaction. Whenever the foreground area is smaller than a pre-decided threshold, a fixed central region of the image is marked as foreground. This is illustrated in Figure 3 which shows correct segmentation as well as the failure cases.

B. Instance Retrieval

We use an instance retrieval pipeline to classify the bill in the image. We follow the instance retrieval approach of [21], the summary of which is illustrated in Figure 4.

1) Building a Visual Vocabulary: We first locate keypoints in the foreground region of the image (obtained from segmentation) and describe the keypoint regions, using any descriptor extractor like SIFT, SURF or ORB-FREAK. We obtain a set of clusters of features using hierarchical K-means algorithm. The distance function between two descriptors x_1 and x_2 is given by $d(x_1, x_2) = \sqrt{(x_1 - x_2)^\top \Sigma^{-1}(x_1 - x_2)}$, where Σ is the covariance matrix of descriptors. As is standard, the descriptor space is affine transformed by the square root of Σ so that Euclidean distance may be used. The set of clusters forms the visual vocabulary of images.

2) Image Indexing Using Text Retrieval Methods: For every training image, after matching each descriptor to its nearest cluster, we get a vector of frequencies (histogram) of visual words in the image. Instead of directly using visual word frequencies for indexing, we employ a standard 'term frequency - inverse document frequency' (*tf-idf*) weighting. Suppose there is a vocabulary of k words, then each image is represented by a k-vector $V_d = (t_1, \ldots, t_i, \ldots, t_k)^\top$, of weighted word frequencies with components

$$t_i = \frac{n_{id}}{n_d} log(\frac{N}{n_i}).$$

Here n_{id} is the number of occurrences of word i in document d, n_d is the total number of words in the document d, n_i is the total number of occurrences of term i in the whole database and N is the total number of documents in the whole database. The weighting is a product of two terms: the *word frequency* $\frac{n_{id}}{n_i}$, and the *inverse document frequency* $log(\frac{N}{n_i})$. However, retrieval on this representation is slow and requires lots of memory. This makes it impractical for applications on mobile phones. Therefore, we use an inverted index for instance retrieval.

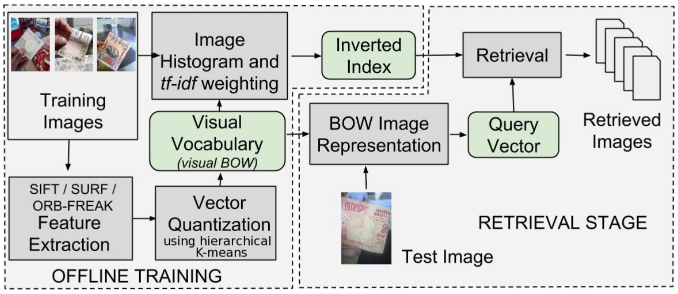

Training Images
Image Histogram and tf-idf weighting
Inverted Index
Retrieval
Retrieved Images
Visual Vocabulary (visual BOW)
BOW Image Representation
Query Vector
SIFT / SURF / ORB-FREAK Feature Extraction
Vector Quantization using hierarchical K-means
OFFLINE TRAINING
RETRIEVAL STAGE
Test Image

Fig. 4: The full pipeline of visual Bag of Words instance retrieval illustrating both the training module and the query module for test images.

3) Retrieval Stage: At the retrieval stage, we obtain a histogram of visual words (query vector) for the test image. Image retrieval is performed by computing the normalized scalar product (cosine of the angle) between the query vector and all *tf-idf* weighted histograms in the database. They are then ranked according to decreasing scalar product. We select the first 10 images for further processing.

4) Spatial re-ranking: The Bag of Words (BoW) model fails to incorporate the spatial information into the ranking of retrieved images. In order to confirm image similarity, we check whether the keypoints in the test image are in spatial consistency with the retrieved images. We use the popular method of geometric verification (GV) by fitting fundamental matrix (adopted from [16]) to find out the number of keypoints of the test image that are spatially consistent with those of the retrieved images.

5) Classification: In the voting mechanism, each retrieved image adds votes to its image class (type of bill) by the number of spatially consistent keypoints it has (computed in the previous step). The class with the highest vote is declared as the result.

C. Adaptation to Mobile

We were able to adapt the above solution to a mobile environment by making very significant reductions in complexity, as much as possible, without sacrificing the effective accuracy. This allows us to achieve the best possible performance, given the severe restrictions in various aspects of the pipeline that we have to contend with.

Segmentation using iterative graph cuts is generally slow and typically takes more than 1 second for a 800×600 image on a 2.2 GHz Dual Core processor. We overcome this problem by performing segmentation at 1/5th of the original image size, thus taking 0.12 seconds. The mask boundary is not accurate as it has to be interpolated to the original scale, but we are still able to remove much of the background in most cases (see Table I (a)).

The recognition model needed for retrieval cannot be used directly on a mobile phone because of the memory requirement. A vocabulary of size 1000K used for instance retrieval along with an inverted index requires approximately

TABLE I: The results of mobile adaptation (a) show the proportion of the different kinds of segmentation that can arise and (b) show the final storage and memory requirements for the application on the mobile phone.

(a)

Segmentation Output	Share
Accurate segmentation	76%
Central region marked as foreground	14%
Less than 15% of background removed	7%
Currency bill marked as background	3%

(b)

Component	Size
RAM use (on average)	23.5MB
Inverted index	20.5MB
Vocabulary (10K)	5.3MB
Keypoints location	11MB
Annotations	6.9KB

2.4GB of storage space and 1.5GB RAM. We perform vocabulary pruning on a 10K vocabulary (adopted from [2]) to make it feasible to run on mobile phones without affecting accuracy. This entails removing less significant and less discriminating visual words from the vocabulary. The pruned vocabulary along with the inverted index being used requires merely 27MB of space (see Table I (b)), which without pruning would require 158MB. Instead of using a flat vocabulary, we use a vocabulary tree built with hierarchical K-means clustering, which highly expedites the assignment.

IV. Experiments

A. Dataset

The various denominations of Indian Rupee bills differ in size and color, apart from the printed denomination and other texts which makes for easy visual identification. However, for the visually impaired, text and color do not help at all and size can lead to confusion because of the similar dimensions of the various bills.

There is currently no available dataset of images of Indian Rupee bills in various configurations that would be suitable for the possible use cases of a visually impaired user. Therefore, part of our work involved creating such a collection (Figure 1). Figure 5 shows statistics of this dataset. The images are captured using popular mobile phone cameras, with different resolutions, viz. VGA, 1.3 megapixels (MP), 2 MP and 5 MP, all on default settings. For each bill, there are 4 different half-folds and 2 full-length configurations. For each denomination we consider at least 12 different bills, across 6 different indoor environments and 7 different outdoor environments, while collecting the dataset. This introduces many variations in illumination, background and pose in the dataset. The dataset contains images of both new and worn out bills, as well as bills with scribbles on them.

B. Implementation

An Android application is designed to run on versions 2.3 and above. It requires a camera of at least 1.3 MP with an autofocus feature. We have utilized the OpenCV library for computer vision and image processing related tasks. While the camera interface is coded in Java, the language of Android, the image processing in the application is done in native C++ code.

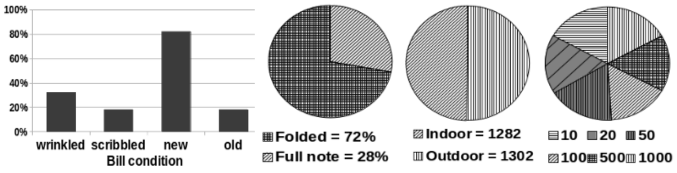


Fig. 5: Various statistics that reflects the dataset's comprehensiveness.

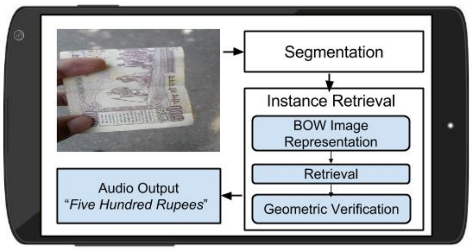


Fig. 6: A conceptual schematic of the back-end of the application; this shows the sequence of processing steps going into decision-making.

Since the application is intended to be a stand-alone one that works upon installation, the various resources required are packaged with the application installation file. They include three files — the vocabulary of visual words, the inverted index, and the annotations. They are saved into the internal memory the first time the application is used, and are loaded into RAM each time it starts. Another resource is the spatial coordinates of all the keypoints. This need not be in the RAM, unlike the others, so we keep it in the phone memory. By doing this, though we sacrifice speed to some extent, we reduce the RAM requirement.

The working of the application is simple and intuitive for a person who cannot see. Once the application has started, so does a camera preview. The preview continues until the user holds the phone still for around 3-4 consecutive seconds, after which it focuses and takes a picture. This gives the user ample time to align and position the bill correctly.

When the picture is captured, the preview freezes while the result is processed. In a few seconds, an audio message is played which has either the denomination of the bill, or a message asking the user to capture the image again. Once the message has been received, the user can close the application by touching the screen, or if he/she needs to continue, the device needs to be shaken 3-4 times and the preview will restart (Figure 2).

Our application recognizes the bill in two major steps (see Figure 6). First, the application removes the background of the image captured by the user as mentioned in Section III. Then it detects SIFT keypoints inside the foreground area of the image. If there are not enough keypoints, it rejects the image and asks the user, via audio message, to capture the image again. The 128-bit SIFT descriptors for the test image are then quantized into visual words using the vocabulary. The application then does a quick *tf-idf* based scoring of all the images in the dataset using the inverted index and finds ten best matches from the database. These images are spatially verified and re-ranked. The audio message corresponding to the final result is then played. However, if the result is ambiguous, the application again asks the user to capture another image, but the result is not discarded and is used in subsequent decision-making.

TABLE II: Classification Accuracy using SIFT, SURF and ORB-FREAK as the feature, with segmentation, for various sizes of the vocabulary.

Feature	Vocabulary Size						
	2K	5K	10K	50K	100K	500K	1000K
SIFT	81.2%	87.6%	87.8%	93.9%	96.1%	96.3%	96.7%
SURF	68.7%	71.4%	72.8%	79.6%	84%	92%	92.4%
ORB-FREAK	49.8%	55.8%	56.6%	65.2%	66.1%	69.3%	71.1%

C. Evaluation

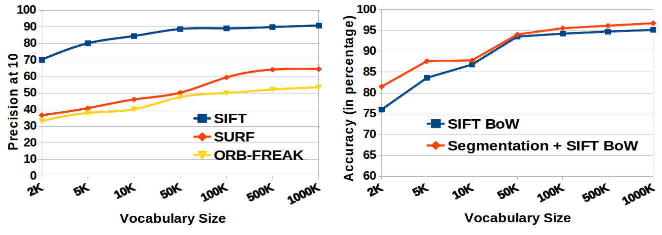
For the purpose of evaluating our approach, we have divided our dataset into two parts — one for training, and the other for testing. We measure the performance in terms of the classification accuracy over 510 test images.

The decision of classifying a test image is done by voting. We retrieve the top 10 most similar images (as described earlier) from the database and perform geometric verification. Each retrieved image votes for its class by the number of spatially consistent keypoints it has. The class with the maximum votes is then declared as the response by the system. If the number of votes is less than a threshold, the image is marked as ambiguous, and the user is asked to take the image again.

D. Results and Discussions

We have experimented with the accuracy tests for various feature detectors and extractors like SIFT, SURF and ORB-FREAK. ORB-FREAK is faster for mobile applications because it is the least computationally intensive among the above-mentioned ones. However, SIFT is far better in terms of accuracy, with 96.7% of the responses being correct using a vocabulary of size 1000K, while still not being infeasible in terms of time. For the same vocabulary size, SURF was able to correctly recognize 92.4% of the test cases whereas for ORB-FREAK it was only 71.1% (see Table II). Moreover, we also see that for precision at 10, SIFT is better than the other alternatives (see Figure 7 (a)). This makes SIFT the most suitable choice for our application.

Using segmentation with instance retrieval clearly improves retrieval performance. In our case, segmentation helps in removing irrelevant keypoints which results in faster retrieval as well as a reduced error rate (Figure 7 (b)). However, a wrong segmentation (marking the note as background) may result in a classification error.



(a) (b)

Fig. 7: (a) Precision at 10 using each of SIFT, SURF and ORB-FREAK as a feature with segmentation for various sizes of vocabulary. (b) Comparison between accuracy of SIFT BoW + GV retrieval pipeline and that of segmentation followed by SIFT BoW + GV retrieval pipeline.

TABLE III: Time analysis for currency recognition application on a mobile phone with 1.2 Ghz CPU and 1GB RAM.

Module	Time in seconds
Read data and load application (one time)	2.4 s

Module	SIFT BoW+GV Time in seconds	
	without segmentation	with segmentation
Segmentation	-	0.27 s
SIFT keypoints detection	0.25 s	0.25 s
SIFT descriptor extraction	0.27 s	0.13 s
Assigning to vocabulary	0.01 s	0.01 s
Inverted index search	0.12 s	0.12 s
Spatial re-ranking	0.61 s	0.31 s
Total Recognition Pipeline	1.26 s	1.09 s

Although the difference in recognition accuracy between both the pipelines viz, SIFT BoW + GV retrieval pipeline and segmentation followed by SIFT BoW + GV retrieval pipeline, is very less for larger vocabulary (around 2%), we see that segmentation helps in reducing the processing time (Table III) in consecutive steps while increasing the accuracy by some amount (see Figure 7 (b)). This is due to fewer keypoints being considered for description and geometric verification, which takes longer than segmentation.

With our approach we have been able to report a recognition accuracy of 96.7% on our dataset of Indian Rupee (₹) while the average processing time remains 1.09 seconds on a typical smartphone with 1.2 Ghz CPU and 1GB RAM running on the Android operating system. Furthermore, we have noticed that the detection (0.25 seconds) and description (0.3 to 0.6 seconds) of SIFT keypoints takes the most time, followed by the step of geometric verication. For the complete time analysis, see Table III. For an illustration of the application in action, see Figure 8.

We also try to provide insight on why our system fails in certain situations (see Figure 9 for failure cases). On the obverse (front) side of each bill (see Figure 9), the image of Mahatma Gandhi is imprinted. There are very few distinguishing features if that half-fold of the bill is considered, less so if the fingers of the user cover some of the surface area. Color being highly sensitive to illumination and fading cannot serve as a reliable feature in such a case. Therefore, such positions often lead to incorrect/ambiguous results. When there are bills of multiple denominations in the view of the camera, the result is bound to be ambiguous. Since the user may be unaware of

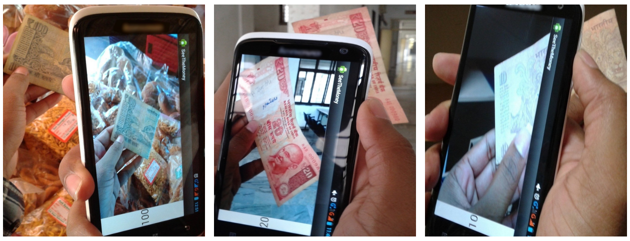


Fig. 8: Currency recognition application at work in various configurations and environments. The application gives a correct result for all the cases.

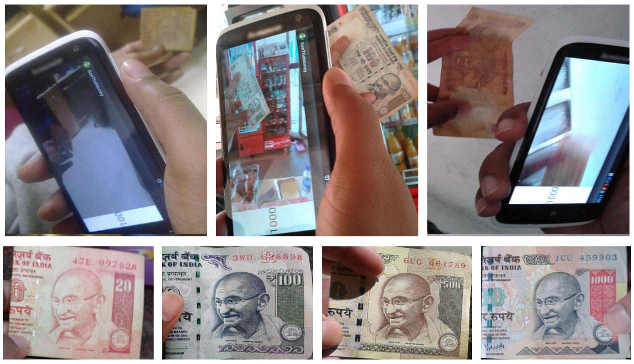


Fig. 9: First row: Failure cases. On the left and middle, the bill occupies less than 40% of the image, leading to a segmentation failure and incorrect response. On the right, the image captured is highly blurred. Second row: The obverse (front) side of bill showing the portrait of Mahatma Gandhi which is common in every denomination. There are very few distinguishing features on this side.

the surroundings, we have no workaround for this situation.

Another case where failure is common, is when the autofocus has not functioned properly, or the phone has been shaken or moved while capturing the image. This results in an image being blurred or the bill being out of focus. In cases where the image is blurred and the system fails to detect keypoints, the image is rejected. However, when the bill is not in focus, the result is either ambiguous or incorrect.

Numerous experiments have been conducted with bills held at different distances and positions relative to the device's camera. They show that strong results are usually reported if the image of the bill spans at least 40% of the total area of the captured image, or if the bill is at a distance of not more than 1 foot or 1 arm's length from the camera for the full configuration (1/2 foot for other configurations). Furthermore, at least 50% of the area of the bill that has been captured, in whatever configuration, should be in the image, to boost the chances of correct recognition.

We have performed various tests to analyze the sensitivity of recognition to segmentation failure. There are two kinds of segmentation failures, one where the background is not fully removed and another where the bill has been wrongly marked as the background. The first type of failure leads to many false positive keypoints, and generally results in an answer of less confidence, which is sometimes incorrect. The second type of failure is more serious. It rejects a number of true keypoints while adding many false positives, often leading to an incorrect result. We have no way of retrieving the former, but we can use a distance threshold to reject the latter while assigning the keypoints to the vocabulary.

V. Conclusion

We have succeeded in our aim to develop a system that can be used to recognize currency for a visually impaired user. We have ported the system to a mobile environment, working around difficulties like limited processing power and memory, while still achieving high accuracy and low reporting time. Currency retrieval and thereafter recognition is an example of fine-grained retrieval of instances which are highly similar. This requires segmentation for removal of clutter. Through our experiments, it has been established that segmentation is helpful for the retrieval process as it reduces the chance of reporting erroneously as well as the overall processing time, and also that the instance retrieval method ensures results swiftly. The methods used work well on noisy images captured from a mobile phone. We expect our system to easily adapt to other currencies of the world as well as a collection of various currencies simultaneously while keeping a similar level of accuracy and speed.

References

[1] Visual impairment and blindness fact sheet, World Health Organisation, 2013. http://www.who.int/mediacentre/factsheets/fs282/en/

[2] Jayaguru Panda, Michael S. Brown, and C. V. Jawahar. Offline mobile instance retrieval with a small memory footprint. *ICCV*, 2013.

[3] K.K. Debnath, J.K. Ahdikary, and M. Shahjahan. A currency recognition system using negatively correlated neural network ensemble. *International Conference on Computers and Information Technology*, 2009.

[4] Ebtesam Althafiri, Muhammad Sarfraz, and Muhannad Alfarras. Bahraini paper currency recognition. *Journal of Advanced Computer Science and Technology Research*, 2012.

[5] Hamid Hassanpour and Payam M. Farahabadi. Using Hidden Markov Models for paper currency recognition. *Expert Systems with Applications: An International Journal*, 2009.

[6] Junfang Guo, Yanyun Zhao, A. Cai. A reliable method for paper currency recognition based on LBP. *Conference on Network Infrastructure and Digital Content*, 2010.

[7] Hai-dong Wang, Leye Gu, and Linping Du. A paper currency number recognition based on fast Adaboost training algorithm. *International Conference on Multimedia Technology*, 2011.

[8] Xu Liu. A camera phone based currency reader for the visually impaired. *ASSETS*, 2008.

[9] F.M Hasanuzzaman, Xiaodong Yang, and YingLi Tian. Robust and effective component-based banknote recognition for the blind. *Annual Wireless and Optical Communications Conference*, 2011.

[10] http://www.looktel.com/

[11] http://www.wirelessrerc.org/node/160

[12] https://play.google.com/store/apps/details?id=org.ideal.currencyid

[13] http://www.google.com/mobile/goggles/

[14] D.M. Chen, S.S. Tsai, R. Vedantham, R. Grzeszczuk, and B. Girod. Streaming mobile augmented reality on mobile phones. *International Symposium on Mixed and Augmented Reality*, 2009.

[15] B. Girod, V. Chandrasekhar, D. M. Chen, N. man Cheung, R. Grzeszczuk, Y. Reznik, S. Tsai, G. Takacs, and R. Vedantham. Mobile visual search. *IEEE Signal Processing Magazine*, 2011.

[16] Jayaguru Panda, Shashank Sharma, and C V Jawahar. Heritage App: Annotating Image on Mobile Phones. *Indian Conference on Vision, Graphics and Image Processing*, 2012

[17] Wei Di, Catherine Wah1, Anurag Bhardwaj, Robinson Piramuthu, and Neel Sundaresan. Style Finder: Fine-grained clothing style detection and retrieval. *CVPR Workshops*, 2013.

[18] Petri Tanskanen, Kalin Kolev, Lorenz Meier, Federico Camposeco, Olivier Saurer, and Marc Pollefeys. Live metric 3D reconstruction on mobile phones. *ICCV*, 2013.

[19] Rabia Jafri, Syed Abid Ali, and Hamid R. Arabnia. Computer vision-based object recognition for the visually impaired using visual tags. *IPCV*, 2013.

[20] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004.

[21] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. *ICCV*, 2003.