iSee: An Android Application for the Assistance of the Visually Impaired

Milad Ghantous, Michel Nahas, Maya Ghamloush, and Maya Rida

Lebanese International University
{Milad.ghantous,Michel.nahas}@liu.edu.lb

Abstract. Smart phone technology and mobile applications have become an indispensable part of our daily life. The primary use however, is targeted towards social media and photography. While some camera–based approaches provided partial solutions for the visually impaired, they still constitute a cumbersome process for the user. iSee is an Android based application that benefits from the commercially available technology to help the visually impaired people improve their day-to-day activities. A single screen tap in iSee is able to serve as a virtual eye by providing a sense of seeing to the blind person by audibly communicating the object(s) names and description. iSee employs efficient object recognition algorithms based on FAST and BRIEF. Implementation results are promising and allow iSee to constitute a basis for more advanced applications.

1 Introduction

Smart phones technology has spread in the international market faster than any other technology in human history. Worldwide Smartphone vendors shipped 237.9 million units in 2013 compared to the 156.2 million units shipped in 2012 which represent 52.3% year-over-year-growth [1]. Android Operating System is one of the smart phones operating systems that have been on the top list of best-selling where 80% of the smart phones shipped ran android operating systems, while Apple's iOS and Microsoft Windows 8 Mobile take up the remaining 20%.

Any user can benefit from the wide range of mobile applications that help in organizing our daily life, connecting with people through social networks, help in education, etc. Only few of these mobile apps have surpassed the purpose of being an added value to our daily life, to becoming a crucial tool. Applications crafted for people with disabilities have emerged lately to help them achieve more what they could do before the Smartphone era. Visually impaired and blind people find it difficult to face their day-to-day problems because of their impairment. The World Health Organization (WHO) estimated that in 2002, 2.6% of the world's total population was visually imapried. In spite of the advancement in technology, there is no complete cure for such problems and they need some assistance to complete their daily tasks. Reading glasses, walking sticks, and Braille tools are few devices which could help them to complete their daily life. However, they don't provide a sense of "seeing" for the blind person. This paper discusses the development of "iSee", an Android based application that aims at serving as a virtual eye. iSee provides the sense of sight by

A.E. Hassanien et al. (Eds.): AMLTA 2014, CCIS 488, pp. 26–35, 2014.

letting the user hold the phone and point anywhere he/she desired and tap on the screen. The application's algorithm runs in the background and then communicates audibly, via a voice message, the object type, name and description. This helps the blind person recognize what's surrounding him/her, or in some cases, find an object he/she is searching for.

The remaining part of this manuscript is organized as follows: section 2 provides a survey over some existing apps in the Google play market and the AppStore. In section 3, we present the main algorithm and implementation details. Section 4 summarized the results and the possible enhancements, while section 5 concludes the paper.

2 Related Work

Several mobile applications on multiple platforms such as iOS, Android and windows were developed with the visually impaired in mind. These applications can be divided among three different categories: currency recognition oriented apps, object detection oriented apps and phone assistant oriented apps.

"eyeNote" [2] and "LookTel" [3] applications belong to the first category. The former recognizes currency bills (US dollars only) and communicates with the user via a voice message, while the latter does the same job but supports more currencies.

The "Blind sighted" fits in the second category. Its functionality is limited to buzzing whenever the user is in 2-3 cm vicinity of an object.

"Siri" [4] and "S voice" [5], iOS's and Samsung's famous voice assistant software belong to the third category. Users can communicate with them via voice messages to inquire about weather, stocks, messages, as well as perform a phone call or a Google search. Few independent applications, like "Blind Navigator" [6] and "Georgie" [7] were also developed with the same goal in mind. The former provides the user with several functions like the phone, calculator, alarm, color identifier and GPS, while the latter adds a TextToSpeech functionality to the above.

Despite the usefulness and effectiveness of the several of the aforementioned applications, the blind people still lack the ability to identify the objects that surround them. Getting to know the type and description of the objects near you not only provides you with information about that specific object but also provides a sense of seeing and overall scene re-construction.

One of the non-mobile approaches is proposed in [8]. A multi-camera network is built by placing a camera at important locations around the user. The developed algorithm can effectively distinguish different classes of objects from test images of the same objects in a cluttered environment.

3 Main Methodology

This manuscript introduces "iSee", an Android powered application for the visually impaired. iSee aims at helping the blind people recognize objects surround them using a voice message. The overall architecture of iSee is illustrated in Fig. 1.

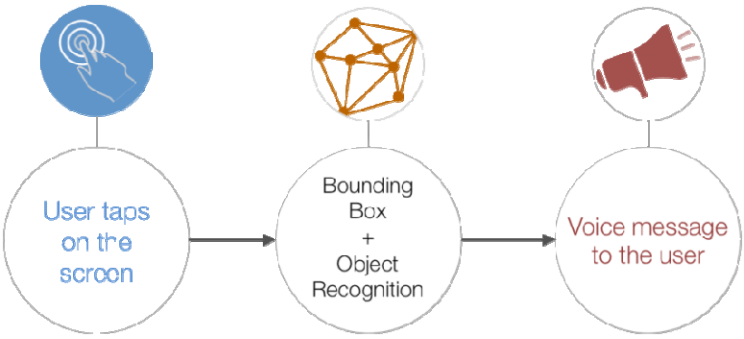

Fig. 1. Overall architecture of iSee

The user interface of iSee is crafted to be user friendly, considering the limitation of the intended users. First, the user opens the application by tapping the app icon or by using any assistant software such as S Voice (or siri for its iOS counterpart). The built-in back camera opens automatically occupying the whole screen, and the user is welcomed with an audio note notifying him/her of the application readiness. At any time, the user can point the phone anywhere he/she desires and taps anywhere on the screen using a single quick tap. An audio note is then given to the user announcing that a snapshot has been captured and ready to be analyzed. In case of a shaky result, the user is asked to re-do the process. Once the snapshot is ready, the application applies object detection and recognition algorithms. If a match is found, the name of the object is communicated to the user via a voice message. Otherwise, the user is notified with a "not found" message and an invitation to try again.

3.1 Object Detection and Recognition

The object detection algorithm constitutes the core of iSee's framework. The majority of algorithms start by finding interest points and their corresponding descriptors in both the scene and a reference image/object. A matching process is then carried between the two images to filter out the good matches. A wide variety of approaches has been proposed in literature [9-14]. Two main approaches were considered: SURF [15] and ORB [16].

SURF, or Speeded Up Robust Features, is a scale and rotation invariant detector and descriptor. SURF employs Hessian-Laplace matrix detectors to find interest points. A feature vector (or descriptor) is then computed to describe the neighborhood of the interest point, by means of local-sensitivity hashing (LSH) [17,18]. SURF relies on Euclidean distance between the matched descriptor and the most similar one. Distances below a certain pre-determined threshold are chosen as good matches. SURF has a computational reduction advantage over its predecessors (mainly SIFT [19]) due to the use of integral image in the computation of the Hessian detector [20]. This property is very attractive for real-time systems as well as power-limited battery-operated devices such as phones, tablets and phablets.

On the other hand, ORB (Oriented FAST and Rotated BRIEF) offers comparable performance to SIFT and SURF while offering two orders of magnitude in speed-up. ORB builds on the FAST detector [21] for interest point generation and on BRIEF [22] for the descriptor computation. FAST relies on the intensity threshold between the center pixel and a circular ring around it. FAST-9 employs a circular ring with a radios of 9 pixels and offers good performance. BRIEF on the other hand uses binary tests between pixels in a smoothed image patch. It is robust to lighting, blurring and distortion but sensitive to rotation. Rotated BREIF (rBRIEF) was proposed to overcome this shortcoming. Fig. 2 depicts a performance comparison between BRIEF, SURF, SIFT and rBRIEF. According to [16], ORB outperforms its competitors in terms of computation time, averaging at 15.3 ms per frame, while SIFT and SURF require 5228 and 217 ms, respectively.

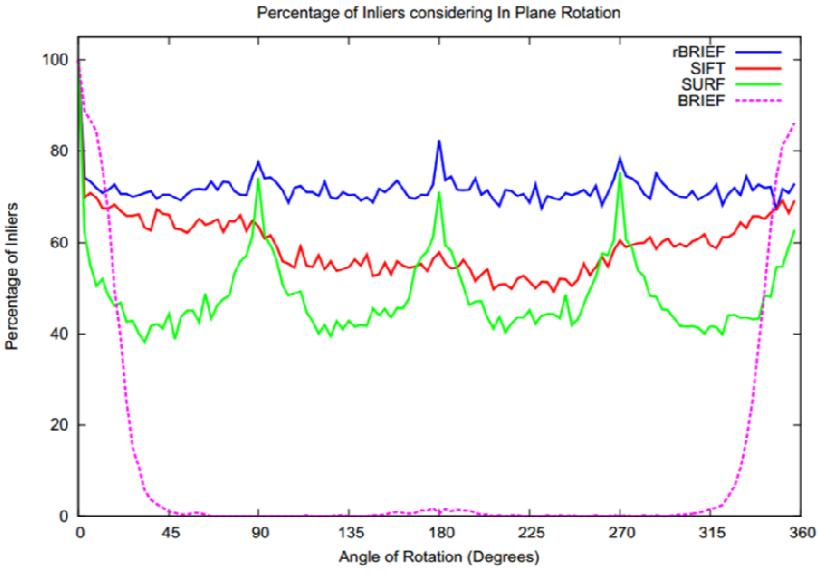

Fig. 2. Matching performance of SIFT, SURF, BRIEF with FAST, and ROB (oFAST + rBRIEF) unser synthetic rotations [16]

Due to the restrictive platform, on which iSee runs, ORB is chosen as the underlying object detection algorithm. The algorithm has 2 inputs: a training object provided by the application (or can be uploaded by the user) and the scene which is the image captured by the user using the tap described before. Interest point detection and descriptor computation take place in the next steps, followed by a nearest neighbor matching process to filter out the good matches. Fig. 3 illustrates the overall iSee object detection flowchart.

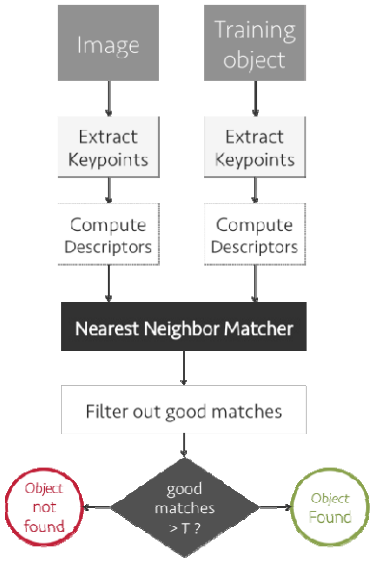

Fig. 3. iSee Matcher Flowchart

The matcher depicted in Fig. 3 is then used to recognize the object(s) found in the scene. A training set of images is provided containing a number of objects. The objects were chosen to be household objects but can be expanded to cover a wider range. Note however, that different postures of the same objects are provided too (front, back, side). Each object in the training set is matched against the scene captured by the user, by means of the aforementioned matcher. A thresholding approach based on the number of matches, is then employed to eliminate all the "bad" matches. The remaining matches are then compared to choose the maximum. At this stage, the object name/type is then communicated to the user. Implementation details are discussed in section 3.2. The overall process is depicted in Fig.4.

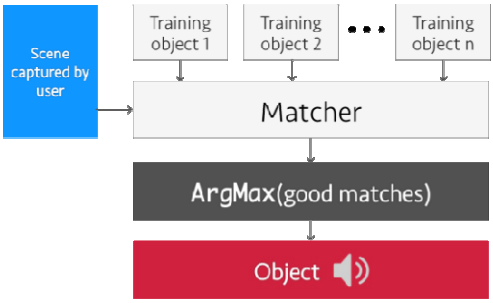

Fig. 4. iSee object recognition process

3.2 Implementation Details

The development of iSee involved several implementation tools and libraries. Android 4.3 SDK (Software developer kit) is chosen as the underlying platform. The interface of the app as well as the camera functions (initialization of the lens, screen tap, internal storage management and text to speech) are all performed using the Java libraries provided by the SDK, and developed using Eclipse IDE. On the other hand, all image processing functionalities were implemented using OpenCV 2.4.6 SDK for Android. OpenCV (Open Source Computer Vision [23]) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel. Two implementation options were available for openCV: JavaCV or the native C++ alternative. Both options can be compiled and used with the Android SDK. Our choice however was set to using the native C++ due to its faster performance and wider library functions. This performance improvement came at the expense of additional implementation complexities. First, Android NDK (Native Development Kit) had to be installed and linked with Eclipse. The NDK is a toolset that allows parts of an app to be implemented using native-code languages such as C and C++. This capability helps in reusing existing code libraries written in these languages. The NDK is suitable for CPU-intensive workloads such as game engines, signal processing, physics simulation, and so on. Second, some of the libraries that are already available in openCV (e.g. SURF) are not available for its Android counterpart. To solve this problem, the libraries were compiled and added manually to the openCV SDK for Android.

The code of iSee proceeds into three main phases: (1) the welcome activity and the camera initialization, (2) the processing part in C++, and (3) the results announced to the user. During the first phase, the openCV libraries are loaded using the `system.loadLibrary` functions. The camera is then initialized and a listener is set up to watch for screen taps. Once a tap is detected on the screen, the scene image is captured and converted into a matrix, S. 'S' becomes the input for phase 2. In the second phase, matrix S and objects from the training set (O_i, $i=1,\ldots,n$) undergo the process depicted in Fig.4 using the following functions: `OrbFeatureDetector()`, `OrbDescriptorExtractor()` and `KnnMatcher()`. Once this process is terminated, the results are passed back to Java in phase 3. During the last phase, the name of the object is converted to a voice message using the `textToSpeech()` utility. Note that all the objects in the training set are labeled with a corresponding name/description.

4 Results and Future Enhancements

Prior to the Android implementation, thorough experimentation was performed to verify the correctness as well as the accuracy of the algorithm. The algorithm was implemented in Xcode on a MacBook Pro running Intel's core i5 processor. A set of 24 objects constituted the test set, each having 3 different postures, resulting in a total of 72 images. Images were chosen to be daily use objects and categorized in 3 groups: coffee mugs, sunglasses and laptops. The algorithm's performance is depicted in Table 1. The average accuracy of all the categories was found to be 91.33%.

Table 1. iSee's performance under the desktop implementation

Category	No. of images	Correctly detected	False positives	Accuracy
Coffee mugs	24	23	1	95.8 %
Sunglasses	24	22	2	91.6 %
Laptops	24	21	3	87.5 %

A snapshot of the running algorithm is depicted in Fig. 5. The top row of the figure denotes a match success between an object and a scene with high number of good matches, while the bottom row represents a mismatch between the object and the scene with a low number of good matches.

Fig. 5. (a),(c) Training object, (b) scene containing the object (d) scene without the object

The algorithm is then implemented on the Android platform as detailed in section 3.2. The match success rate was found to be slightly less than its desktop implementation averaging at 89%. Table 2 depicts the algorithm accuracy per category. Fig. 6. Shows few snapshots of the application running on a Samsung Note 3.

Table 2. iSee's performance under Android Implementation

Category	No. of images	Correctly detected	False positives	Accuracy
Coffee mugs	24	22	1	91.6 %
Sunglasses	24	21	2	87.5 %
Laptops	24	21	3	87.5 %

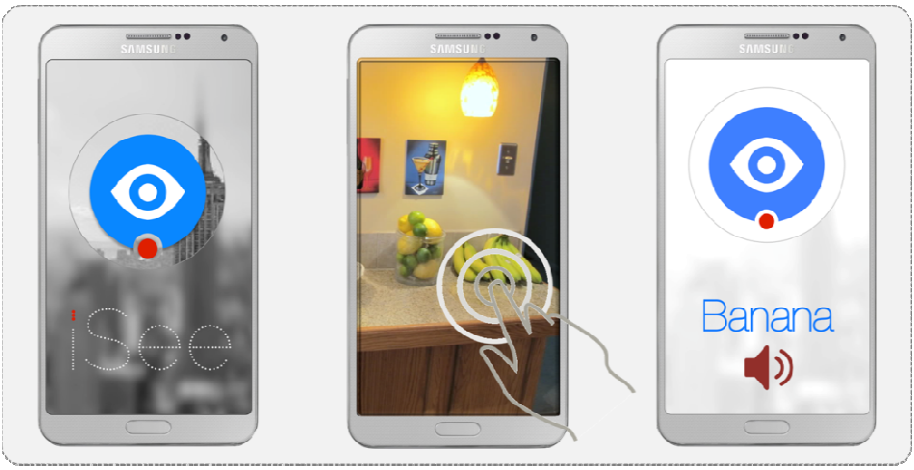

Fig. 6. Running iSee app screenshots

The processing time of iSee is finally analyzed based on the size of the training library and compared against its desktop counterpart. The results are illustrated in Fig.7. iSee was able to recognize an object within 1.8 seconds given a library size of 75 images, while its desktop counterpart outperformed it by 0.8 seconds. We strongly argue though, that with recent advancements in processor speeds employed in currently released smartphones, discrepancies between desktop and mobile performances will continue to decrease.

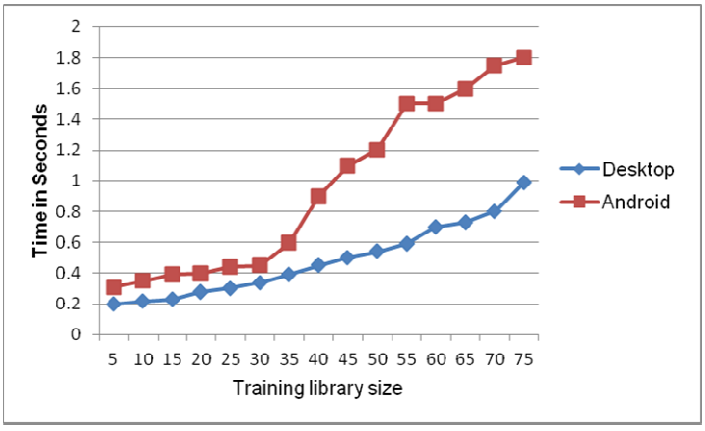

Fig. 7. Training library size versus processing time for Desktop and Android implementations

4.1 Possible Enhancements

The work presented in this manuscript constitutes a basis for possible enhancements and app ideas. One of the possible enhancements to this app is to let the user or his/her assistant upload their personal belongings and objects to create a "personalized" library rather than a generic one. Despite that this change defeats the app idea in the first place, it constitutes a powerful and fast tool parallel to iSee.

iSee could be also turned around to let the user ask for specific object to search for using a voice command. He/she then holds the phone and scans the surrounding until the phone buzzes denoting a match success. This will help the visually impaired find his belongings in an easier way. Fig. 8 illustrates this idea.

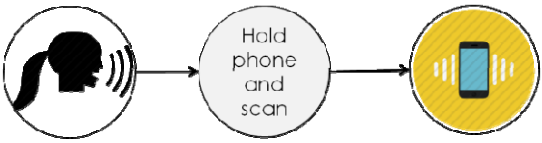

Fig. 8. Alternative use to iSee

A more advanced research can lead, not only to a better recognition rate but also to scene behavior, in which, the app can describe to the visually impaired what's happening in real time in his/her surrounding.

5 Conclusion

This paper presented the development of iSee, an Android powered application for the visually impaired. iSee lets the user taps on the screen and then communicates via a voice message the name and type of the object in the scene. iSee employs ORB as the underlying object recognition algorithm due to its computational efficiency on mobile platforms. The implementation results of iSee were promising and constituted a basis for more advanced developments.

References

1. Aaron, S.: Smartphone Ownership. Pew Research Center's Internet & American Life Project (2013), http://pewinternet.org/Reports/2013/Smartphone-Ownership-2013.aspx (accessed June 2014)
2. EyeNote App, Apple Appstore (2010), http://www.eyenote.gov/
3. Sudol, J., et al.: LookTel- A comprehensive platform for computer aided visual assistance. Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition (2010)
4. Apple Siri (2011), https://www.apple.com/ios/siri/
5. S Voice (2011), http://www.samsung.com/global/galaxys3/svoice.html

6. Blind Navigator App, Google Play (2013), https://play.google.com/store/apps/details?id=com.Blindnavigator
7. Georgie App, Google Play (2012), http://www.georgiephone.com/
8. Yi, C., et al.: Finding objects for assisting blind people. Network Modeling Analysis in Health Informatics and Bioinformatics 2(2), 71–79 (2013), doi:10.1007/s13721-013-0026-x
9. Linderberg, T.: Feature detection with automatic scale selection. Computer Vision 30(2), 79–116 (1998)
10. Lowe, D.: Distinctive image features from scale-invariant keypoints, cascade filtering approach. Computer Vision 2(60), 91–110 (2004)
11. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part I. LNCS, vol. 2350, pp. 128–142. Springer, Heidelberg (2002)
12. Ke, Y., Sukthankar, R.: PCA-SIFT: A more distinctive representation for local image descriptors. Paper presented at CVPR Computer Vision and Pattern Recognition (2004)
13. Tuytelaars, T., Van Gool, L.: Wide baseline stereo based on local, affinely invariant regions. Paper presented at the British Machine Vision Conference (2000)
14. Matas, J., Chum, O., Pajdla, T.: Robust wide baseline stereo from maximally stable external regions. Paper presented at the British Machine Vision Conference (2002)
15. Hebert, B., et al.: Speed-up Robust Features (SURF). Computer Vision and Image Understanding 110(3), 346–359 (2008)
16. Rublee, E., et al.: ORB: an efficient alternative to SIFT or SURF. Paper presented at the IEEE International Conference on Computer Vision (2011)
17. Qin, L., et al.: Multi- Probe LSH: Efficient Indexing for High-Dimensional Similarity Search. In: Proceedings of Very Large Database Conference, Vienna (2007)
18. Har-Peled, S., Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. Theory of Computing 8(1), 321–350 (2012)
19. Lopez-de-la-Calleja, M., et al.: Superpixel-Based Object Detection Using Local Feature Matching. In: Proceedings of the 29th Conference of the Robotics Society of Japan, Toyosu (2011)
20. Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. Paper presented at the International Conference on Computer Vision (2001)
21. Rosten, E., Drummond, T.: Machine learning for highspeed corner detection. Paper presented in European Conference on Computer Vision (2006)
22. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 778–792. Springer, Heidelberg (2010)
23. openCV (2014), http://opencv.org/