IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555
Vol. 3, No.1, February 2013

Cameraphone Recognition of Arabic Fingerspelling

Ibrahim Elhenawy
Department of Computer Science
Faculty of Computers and Informatics, Zagazig University
Zagazig, Egypt
henawy2000@yahoo.com

Abdelwahed Khamiss
Department of Computer Science
Faculty of Computers and Informatics, Zagazig University
Zagazig, Egypt
a.khamiss@zu.edu.eg

***Abstract*— The increasing ubiquity of today's Smartphones made them an ideal platform for many emerging mobile applications that range from navigation systems to healthcare monitoring. However, the majority of the mobile systems are being marketed primarily to the able-bodied users and most of them fail to be accessible to people with disabilities especially the deaf. As a way of enabling the deaf to interact naturally with the Smartphones, automatic hand sign recognition appears as a suitable means. In this paper, we propose,** AndroSpell**, vision-based Smartphones software to automatically recognize Arabic Fingerspelling signs. The system prototype was built entirely on the top of the cameraphone and was able to classify up to 10 postures with accuracy of 97%. The implementation and the evaluation of the system provide clear evidence that the emerging capabilities of the cameraphones could be fairly harnessed for use as an accessible technology for the deaf.**

mobile computing; mobile accessibility; posture recognition

I. INTRODUCTION

Technology is often created without regard to people with disabilities. This creates unnecessary barriers to a large number of the disabled who decided to employ technology in one way or another in their daily lives. As a natural consequence, Assistive Technologies come into the scene to bridge this gap.

However, most of the currently available assistive technologies are intimately connected with the desktop-based computers making their use limited to the stationary environments. The recent developments and the advanced capabilities of the Smartphone allowed for shifting some these technologies from the desktop to the mobile platforms and thus exploiting their inherent mobility, ubiquity and ease of use.

In the same vein, we developed AndroSpell (Figure 1) in an effort to enrich the Android Smartphone with static hand gesture recognition capability. We exploit one of the most ubiquitous sensors (the camera in Smartphone) and computer vision algorithms to interpret the finger spelling postures in a near real-time manner.

The paper is structured as follows. Section 2 presents the detailed description of the overall system architecture and the components representing various stages in hand posture recognition. This is followed by Section 3 that includes the main experimental results we got form the feature extraction and the classification stages. Section 4 lists related research efforts. Section 5 highlights the conclusion and the future work.

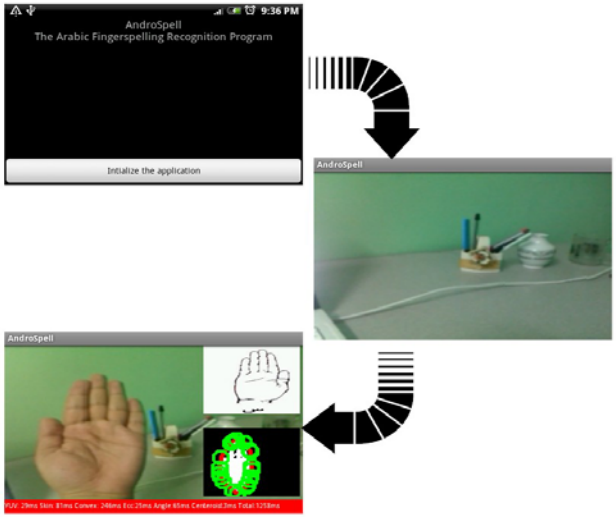

Figure 1. AndroSpell Prototype

II. SYSTEM ARCHITECTURE

The goal of our system is to correctly classify Arabic Finger spelling postures (Figure 2) of the user's bare hands using the mobile phone's built-in camera without any additional external sensors (e.g. instrumented gloves). The architecture employs a staged approach for posture recognition as illustrated in Figure 2.

We believe that a continuous recognition of hand gestures is the best option in the sense that it will be increasingly usable in a variety of potential AndroSpell applications and, more importantly, it represents the most convenient gestural interaction way for the user (who is supposed to use the system to interpret many postures in one setting). As we want to provide a kind of real-time recognition service, the system must handle camera frames processing on the fly. This means, on one hand, that the system should avoid any considerable latency in response time. On the other hand, the user will not use the capture button to provide an input for the program. Alternatively, there will be modules responsible for grabbing consecutive frames from the camera, deciding which of them may contain valuable information (not a frame representing the background for example) then forwarding it to the

classification module. Each of the overall system processes (figure 3) is summarized in the following sections.

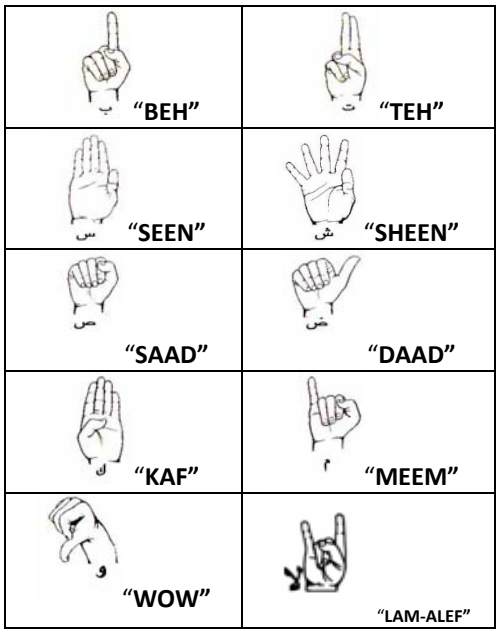

Figure 2. AndroSpell postures list

A. *Camera Frame Preprocessing*

The preprocessing stage aims at improving the quality of the input images to increase the performance (in terms of processing speed and accuracy) of the subsequent stages. In our system frames are grabbed natively in YUV420sp format and preprocessing is done in two steps:

1) **Discarding repeated frames**: We filter out the repeated frames using the "Baseline Differencing" approach tested here [1]. First we calculate the sum of absolute differences between successive frames, and then we check that sum against a predefined threshold. If it exceeds the threshold value then the frame is considered a key frame, otherwise it is rejected.
2) **Color Conversion**: YUV frames are converted to RGB.

B. *Hand Segmentation*

After preprocessing a frame it is then delivered to the hand segmentation module where the process of identifying the hand's area in the image takes place. First skin pixels must be labeled then a kind of background objects filtering technique is applied to increase the confidence of the hand segmentation. Color based skin detection is used to discriminate the hand pixels from the background. Working on the RGB color space, we used pixel based heuristic [2].

The output of this phase is a binary image where the foreground pixels (skin color candidates) are represented as white pixels (Figure 4)

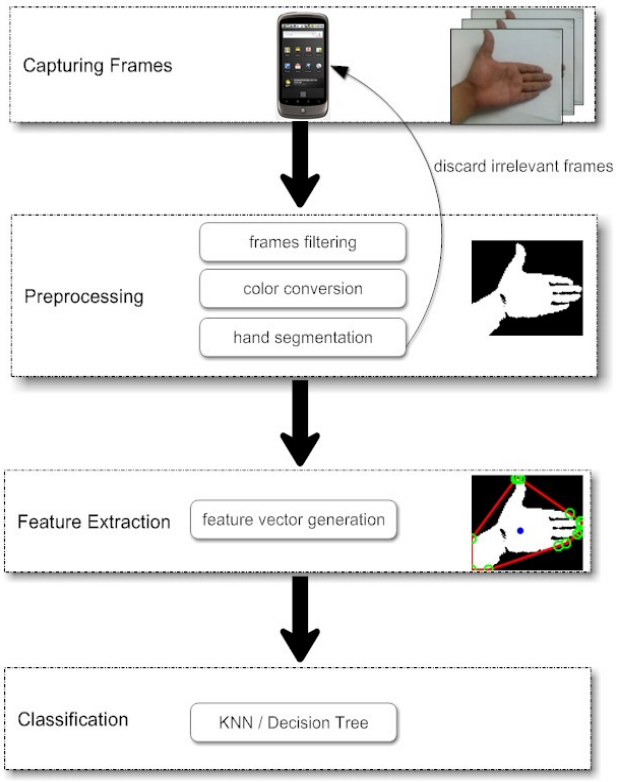

Figure 3. AndroSpell Architecture

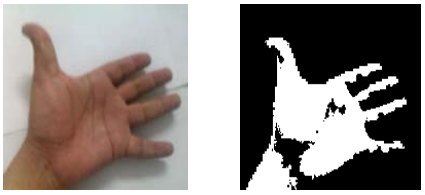

Figure 4. Skin Detection

The binary image resulted from the previous phase may contain background areas that have passed through the skin detector as potential skin pixels. We address this case primarily by analyzing the total area occupied by the skin pixels. The skin area percentage of the total image should fall between two (experimentally tested) thresholds t1, t2. Otherwise, the frame is considered to contain either nothing but background object(s) that exhibit skin-like properties or an actual hand that it is too distant/close from the camera to be recognized. In both cases, the frame is invalid and is ultimately discarded.

For labeling the various connected components we benchmarked two different Connected Component algorithms on the phone:

Identify applicable sponsor/s here. *(sponsors)*

The classical Two-Pass [3] algorithm that makes two passes over every single frame: one pass to assign provisional labels to object pixels and record the equivalence information among provisional labels (we assumed 8-connectivity) and the second to replace each temporary label by the label of its equivalence class. A union-find data structure is employed to keep track of equivalence relationships and manage the relabeling process.

The Run-based Two Scan algorithm [4] which, as opposed to the Two-Pass algorithm, works on run data (a "run" is a block of contiguous object pixels in a row) that are obtained in the algorithm's first scan and recorded in a queue. In the runtime, all provisional labels that are assigned to a connected component found so far during the first scan are combined in a provisional label set, and the smallest label is used as their representative label. Connectivity checks are applied to runs (not pixels).

Although the results (Figure 5) claim that the classical two-pass algorithm is more efficient than the run-based one, a general conclusion that can be reached here is that the component labeling process takes a considerable time in both cases. Even with the smallest resolution available natively from the camera (176X144) the two-pass algorithm needs 755 ms to process each frame.

The output of this phase is a list of the connected components sorted by the perimeter. The largest component (which is supposed to be the hand's blob) is picked and the remaining objects are discarded.

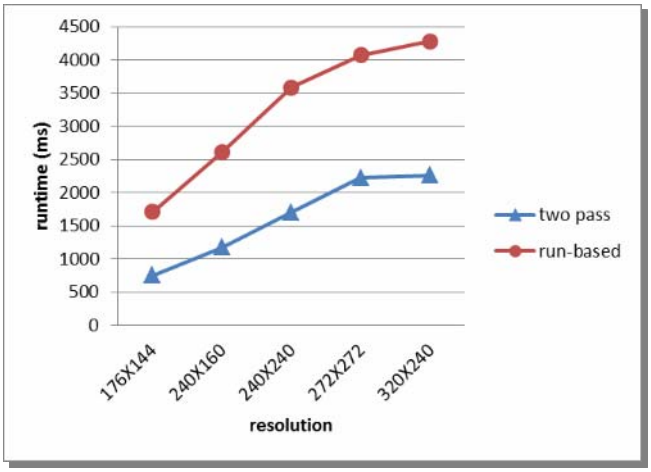

Figure 5. Component Labeling Algorithms Benchmark

C. Camera Frame Preprocessing

Most of the Arabic Finger spelling signs are performed using a single hand (contrary to others like Turkish Finger spelling for example [5]) (a fact that allows AndroSpell's user to hold the phone with one hand and perform the signs with the other). In the features calculation phase we seek to produce a list of various visual features (feature vector) that describe the geometric characteristics of the hand's binary object.

Good feature extraction is a challenging problem and a wealth of previous research work investigated the extraction of numerous features and testing their contribution to the accuracy of the recognition. However, in the constrained Smartphone platform case a careful tradeoff between the recognition rate and the performance must be considered. While the selection of features is critical to realizing robust classification, it is necessary not to expose the system to extracting computationally demanding features that may contribute the quality of the recognition on the expense of degrading the performance (for example our experiment to calculate SURF features [6] on the phone took 1700 ms on average for 320X240 frames and Hu Moments were computed in 802 ms for 240X160 frames).

We experimented with a number of features and depended on the following geometric descriptors for our feature vector:

- **Centroid**: The centroid point of the hand area is given by:

$$\bar{x} = \frac{1}{|R|} \cdot \sum_{(u,v)\in R} u^1 v^0 \qquad \bar{y} = \frac{1}{|R|} \cdot \sum_{(u,v)\in R} u^0 v^1$$

Where |R| is the area and calculated as follows:

$$A(R) = |R| = \sum_{(u,v)\in R} 1$$

- **Orientation**: Also called the axis of the least inertia (θ_R) and by this we mean the direction of the major axis, which is the axis that runs through the hand object centroid and along the widest part of the hand's region. The orientation is calculated using central moments (Mpq(R)) as follows:

$$\theta_R = \frac{1}{2} \tan^{-1}\left(\frac{2.\mu_{11}(R)}{\mu_{20}(R) - \mu_{02}(R)} \right)$$

Provided that the central moment (Mpq(R)) is defined as:

$$\mu_{pq}(R) = \sum_{(u,v)\in R} (u-\bar{x})^p . (v-\bar{y})^q$$

- **Eccentricity**: similar to the region orientation, moments can also be used to determine the eccentricity Ecc(R) of the hand region:

$$Ecc(R) = \frac{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4.\mu_{11}^2}}{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4.\mu_{11}^2}}$$

The values of eccentricity are in the range [1,∞), where eccentricity = 1 corresponds to a circular region, and elongated regions have values > 1.

- **Convex Hull:** Convex Hull was employed in various hand gesture recognition systems [7, 8]. Calculating the hand's bounding convex hull in our system is based on Andrew's Monotone Chain [9] algorithm. It constructs convex hull of a set of 2D points in $O(n \log n)$ time. It does this by first sorting the points lexicographically (first by x-coordinate, and in case of a tie, by y-coordinate), and then constructing upper and lower hulls of the points.

  Although Andrew's algorithm requires the input set of points to be sorted, we handle this in our implementation by scanning the input frame properly. For example, PU constitutes the set points that will be fed into the upper hull procedure. We collect them by scanning the frame column by column from left to right and from top to down until the first white pixel is encountered then we record its position and move to the next column.

  Since the numerous vertices resulted from the convex hull algorithm will result in high dimensional feature vector, the classification task may become complicated. For tackling this difficulty, we forward the produced list of vertices to a convex hull smoothing procedure to generate a kind of less complicated and smoothed convex hull. The clustering approach described here [10] is used. The technique works by grouping the adjacent vertices into a single cluster using the k-means. Adjacency is determined based on the polar angle of each vertex.

  The previous geometric descriptors are combined into the feature vector. Convex hull descriptors are the distances from the centroid to the vertices of the smoothed convex hull after scaling them so that the largest is of them becomes equal to 1**.**

D. Classification

Previous research works have proposed and tested several classification techniques. Here we use two of the well-known classification algorithms.

K-Nearest Neighbors (KNN) that proved to work well in the hand gesture recognition problem [11]. Although KNN doesn't make any assumptions about the underlying data distribution, the data is assumed to be in a feature space and thus have a notion of distance function responsible for measuring similarity between instances. Euclidean distance is the most commonly used distance function and is the one used in our system.

Given a test instance (posture feature vector) x, its k nearest neighbors are found and a vote conducted to assign the most common class to x. KNN requires no training and thus all the training data are kept along with it in the runtime, this means that a linear (sequential) search of the training data would require nontrivial time for large amount of training data. We developed a KD-Tree [12] to keep/search the training instances during the program runtime.

Decision Tree classifier was adopted successfully in the hand gesture recognition [13]. We use it in AndroSpell as an alternative to the k-nearest neighbor. We constructed the classification decision tree for our 10 gestures list using the C4.5 algorithm [14]. Eccentricity was selected as the root of the tree. The orientation (angle) and some convex hull descriptors were involved in the classification criteria.

III. EVALUATING ANDROSPELL

AndroSpell prototype was implemented as a self-contained piece of software that runs on Android phones. We evaluate our system by conducting various experiments on the prototype we developed. All the experiments were carried on using the HTC Legend phone equipped with 600 MHz ARM 11 processor Qualcomm MSM7227 chipset and Android v2.1 (Eclair) as the operating system. The camera is configured to grab 176X144 frames. As a finger spelling recognition system we care about the accuracy of the system concerning how many postures were identified correctly. Also the embedded constrained nature of the Smartphone platform requires measuring and evaluating the computational performance and, more importantly, the power consumption of the software.

A. Classification Accuracy

We collected 1000 gesture images for training and validating the classification modules in AndroSpell. The images were uniformly distributed (100 images per gesture) and 66% of the collected data used for the training while the remaining portion used as the test set. The overall classification results (Figure 8) showed the decision tree to be the most robust classifier with a precision of 97%. This outperforms the 1NN that achieved a recognition rate slightly less than 96%. However 1NN becomes the best classifier when compared to 2NN, 3NN and 4NN respectively. Note that in the cases of 2NN, 3NN and 4NN we select the class of the gesture based on the majority voting.

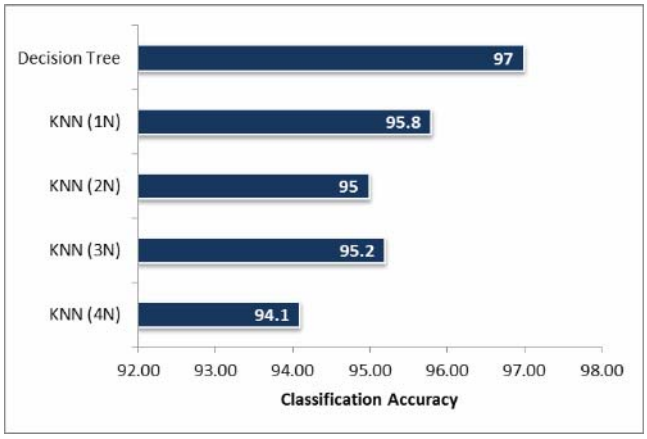

Figure 6. Classification Precision

B. *Computational Performance*

The computational performance is a key criterion in assessing the usability of any gesture recognition program and our system is not an exception especially that its components depend on the computationally demanding vision algorithms. However it is challenging to keep the system response time within an acceptable threshold time. This happens because of three main reasons: 1) the preprocessing modules consume a considerable time of the total frame processing time (for example the YUV conversion stage takes approximately 250 milliseconds per frame), 2) the connected component filtering (CCF) is going to take at least 755 milliseconds (as mentioned in section 2.2), 3) the features we extract make extensive use of floating point arithmetic that proved to be 2x slower than integer arithmetic on Android devices [15].

An earlier study [16] suggests that Android applications can become up to 10 times faster if they utilized native code. Android applications are usually programmed in Java using the Android SDK while native component could be developed using the Android NDK. We reprogrammed the frame preprocessing code using C++ and ported it to our application using Java Native Interfaces (JNI). A considerable performance gain (Figure 9) that is equal to 386 milliseconds was achieved. A second optimization step that was taken is down-sampling the captured frame to 100X100 pixels in order to shorten the CCF and the feature extraction period. The ultimate result is that the total frame processing and recognition time is 1200 milliseconds when using decision tree and 1400 milliseconds with 1NN classifier.

Figure 10 summarizes the frame processing profiling results for AndroSpell. Clearly, KNN classification consumes a considerable time compared to the decision tree classification that takes 1.3 ms (approximately 0%) on average of the total frame processing time. The KD-Tree implementation used to speed up the search process is still performing slower. The technical reasoning to this phenomenon is that the decision tree once trained becomes a set of rules that could be evaluated very efficiently in the run time.

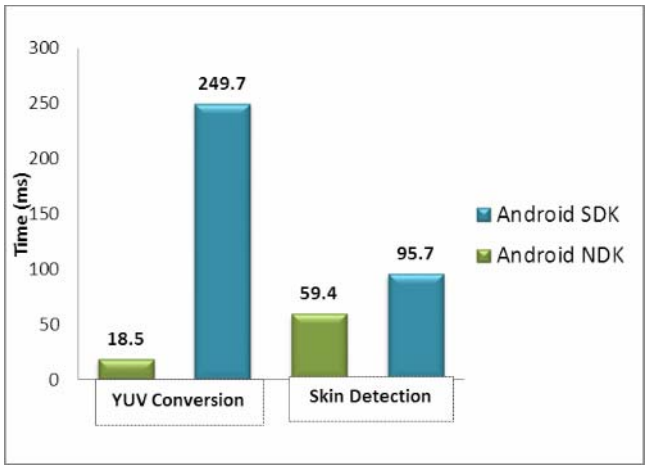

Figure 7. Native Code Performance Gain

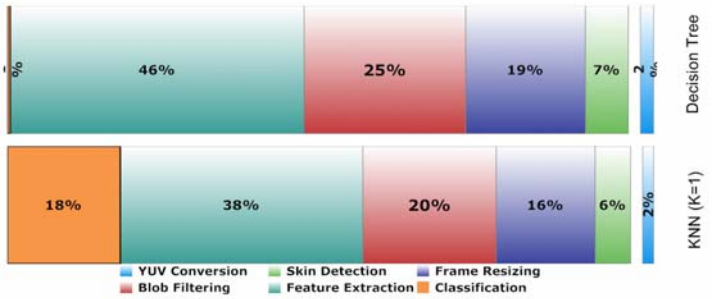

Figure 8. Frame Processing Profiling

IV. RELATED WORK

The body of the work presented here is developed mainly to be mobile accessibility software, but it also falls in the broader context of static hand gesture recognition.

Work on applications and systems for mobile accessible and assistive technology is growing in importance. Applications of accessible mobile and portable systems range from text-to-speech output and screen magnification to audio amplifiers and hearing aid compatibility. For example, there exist systems for the visually impaired that can guide a person in the case of an emergency [17] , help him to find his way [18] and allow him to remotely control home appliance using voice recognition [19].

A number of existing systems addressed the issue of mobile accessibility for the deaf. Jemni et. al. [20] proposed an application for automatically translating text messages into MMS containing avatar-based sign language animation to be sent to the deaf. MobileASL [21] is another research effort that develops a sign-aware video encoder to be used by the deaf in their video calls. While the previously mentioned examples pursued a line of research that aims to provide the deaf with accessibility to the mobile communication, our system focuses instead on offering accessibility to the Smartphone platform itself which, we believe, will form a framework for future Smartphone hand gesture based applications.

Recently, a work by [10] studied the feasibility of hand gesture recognition on Smartphone. Our work goes beyond the idea of viability to propose a robust continuous hand gesture recognition mobile prototype.

In the context of static hand gesture, various works have already dealt with this challenging field using various techniques. Garg [22] summarizes the main approaches to hand gesture recognition. The whole process of gesture recognition in these systems can be coarsely divided into few sequential phases with each phase responsible for performing specific task and delivering the result to the next phase. Our work offers a system with the same purpose and architectural similarity. However, the design choices were adapted to enable successful deployment and operation on the Smartphone.

V. CONCLUSION AND FUTURE WORK

In this paper we presented the design, implementation and evaluation of AndroSpell; the cameraphone Arabic Finger spelling recognition system. Our results indicate that the system performance and the classification accuracy enabled the Smartphone to robustly interpret static hand gestures. A number of challenges and limitations will be considered for future developments of the system. New ways to optimize the total frame processing time should be investigated. We will test another set of features and move more of computationally expensive code to the native side. Further work will consider the recognition of the dynamic (multiple frames) gestures. Hierarchical classification may be used in the cases where confusion between signs takes place. We will use the framework presented here as a base for building interactive mobile gesture-based applications like an educational finger spelling game targeted to children.

REFERENCES

1. Cherniavsky, N., Chon, J., Wobbrock, J.O., Ladner, R.E. and Riskin, E.A. Activity analysis enabling real-time video communication on mobile phones for deaf users. Proceedings of the 22nd annual ACM symposium on User interface software and technology. ACM, 2009, pp. 79-88
2. Kovac, J., Peer, P. and Solina, F. Human skin color clustering for face detection. EUROCON 2003. Computer as a Tool. The IEEE Region 8. 2003, Vol. 2, pp. 144 - 148 vol.2
3. Shapiro, L.G., Stockman, G.C., Shapiro, L.G. and Stockman, G. Computer Vision. Prentice Hall, Hardcover, 2001, pp. 69–73
4. He, L., Chao, Y. and Suzuki, K. A Run-Based Two-Scan Labeling Algorithm. Image Analysis and Recognition Springer Berlin / Heidelberg, 2007, Vol. 4633, pp. 131-142
5. Altun, O., Albayrak, S., Ekinci, A. and Bükün, B. Turkish Fingerspelling Recognition System Using Axis of Least Inertia Based Fast Alignment. AI 2006: Advances in Artificial Intelligence, 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006, Proceedings. Springer, 2006, Vol. 4304, pp. 473-481
6. Bay, H., Tuytelaars, T. and Gool, L.V. Surf: Speeded up robust features. In ECCV 2006, pp. 404-417
7. Manresa, C., Varona, J., Mas, R. and Perales, F.J. Hand Tracking and Gesture Recognition for Human-Computer Interaction. Hand The, Citeseer, 2005, Vol. 5(3), pp. 96-104
8. Tan, T. and Guo, Z. Research of hand positioning and gesture recognition based on binocular vision. VR Innovation (ISVRI), 2011 IEEE International Symposium on 2011, pp. 311 -315
9. Andrew, A.M. Another Efficient Algorithm for Convex Hulls in Two Dimensions. Information Processing Letters, 1979, Vol. 9, pp. 216-219
10. Tarrataca, L., Santos, A.C. and Cardoso, J. a.M.P. The current feasibility of gesture recognition for a smartphone using J2ME. Proceedings of the 2009 ACM symposium on Applied Computing. ACM, 2009, pp. 1642-1649
11. Savaris, A. and von Wangenheim, A. Comparative evaluation of static gesture recognition techniques based on nearest neighbor, neural networks and support vector machines. Journal of the Brazilian Computer Society, Springer London, 2010, Vol. 16, pp. 147-162.
12. Moore, A.W. An Intoductory Tutorial on Kd-Trees. 1991
13. Nisar, S., Khan, A.A. and Javed, M.Y. A statistical feature based decision tree approach for hand gesture recognition. Proceedings of the 7th International Conference on Frontiers of Information Technology. ACM, 2009, pp. 27:1-27:6
14. Quinlan, J.R. C4.5: Programs for Machine Learning. 1993
15. http://developer.android.com/guide/practices/design/performance.html
16. Batyuk, L., Schmidt, A.-D., Schmidt, H.-G., Camtepe, A. and Albayrak, S. Developing and Benchmarking Native Linux Applications on Android. MobileWireless Middleware, Operating Systems, and Applications Springer Berlin Heidelberg, 2009, Vol. 7, pp. 381-392
17. Amemiya, T. and Sugiyama, H. Design of a Haptic Direction Indicator for Visually Impaired People in Emergency Situations. Computers Helping People with Special Needs Springer Berlin / Heidelberg, 2008, Vol. 5105, pp. 1141-1144
18. Ivanchenko, V., Coughlan, J. and Shen, H. Crosswatch: A Camera Phone System for Orienting Visually Impaired Pedestrians at Traffic Intersections. Computers Helping People with Special Needs Springer Berlin / Heidelberg, 2008, Vol. 5105, pp. 1122-1128
19. Yoshida, R. and Yasumura, M. A New Cell Phone Remote Control for People with Visual Impairment. Computers Helping People with Special Needs. Springer Berlin / Heidelberg, 2008, Vol. 5105, pp. 1145-1152
20. Jemni, M., Ghoul, O.E., Yahia, N.B. and Boulares, M. Sign Language MMS to Make Cell Phones Accessible to the Deaf and Hard-of-hearing Community. CVHI 2007
21. Cavender, A., Ladner, R.E. and Riskin, E.A. MobileASL: intelligibility of sign language video as constrained by mobile phone technology. Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility. ACM, 2006, pp. 71-78
22. Garg, P., Aggarwal, N. and Sofat, S. Vision Based Hand Gesture Recognition. World Academy of Science Engineering and Technology, 2009, pp. 972-977