International Journal of Hybrid Information Technology
Vol.8, No.10 (2015), pp.141-152
http://dx.doi.org/10.14257/ijhit.2015.8.10.13

Augmented Piano Reality

Ihab Zaqout[1], Samar Elhissi[2], Aya Jarour[3] and Heba Elowini[4]

[1]Department of Information Technology,
[2,3,4]Department of Software Engineering
Faculty of Engineering and Information Technology, Al-Azhar University, Gaza, Palestine
[1]ihabzakout@yahoo.com, [2]samarelhissi@hotmail.com,
[3]betmoon.92@hotmail.com, [4]hebanowi@gmail.com

Abstract

As mobiles become cheaper and gain popularity, mobile applications are developing quickly and trying to utilize more techniques to make these applications more efficient and usable. Augmented reality is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data. This research develops an android mobile application that utilizes real-time image processing techniques to provide simulated functionality to a hand drawing of piano keys. A piano keyboard with fourteen white keys and ten black keys is drawn used a black thick pen on paper, and the mobile is set in an angel that gives the optimum view of the piano keyboard when the camera is turned on. When the application is started, a camera shot of the piano, without the presence of hands or fingers in the view, is taken. The user can then start playing on his drawn piano. The application detects keystrokes and decides which key is pressed; and then plays the corresponding tone. The application is developed using OpenCV library which focuses on real-time image processing and computer vision, since the target devices with android platform, a Java program is used to port the C++ code to android. The application is tested on multiple android-based devices with different specifications under good lighting conditions. The empirical results are impressive and comparable across different devices, despite all changes in lighting and background; all devices exhibited the same level of accuracy in detecting fingers and drawn pianos.

Keywords: *Augmented reality, border following, skin detection, openCV, piano detection, finger detection, keystrokes detection*

1. Introduction

1.1 Background

The piano (an abbreviation of pianoforte) is a musical instrument played using a keyboard. It is frequently used in jazz and classical music for composing and rehearsal, accompaniment, ensemble use, chamber music, and for solo performance. Although the piano is often expensive and not portable, its versatility and ubiquity have made it one of the world's most familiar musical instruments.

The grand piano and the upright piano are examples of modern acoustic pianos, with various styles of each. In addition, there are another three specialized and novelty pianos named electronic, electric, and digital. Electric pianos based on electromechanical designs, electronic pianos that synthesize piano-like tones using oscillators, and digital pianos using digital samples of acoustic piano sounds. The standard modern piano contains 88 keys and has a compass of seven full octaves plus a few keys; each key produces a different tone. Commercially, some musical keyboards sold have fewer keys than the ordinary piano [1].

Due to its popularity, many computer and mobile based applications are made to simulate the experience of playing the piano/keyboard. When a key is pressed, a generated or pre-recorded tone is played accordingly. The mobile applications that simulate the piano have different styles; each one try to increase the number of keys and the space for playing them, and generate sound to imitate the sound of real piano, however the size of mobile screens sets a limit to the number of keys that are displayed simultaneously, and this is the big challenge.

1.2 Statement of Problem

Musical applications on mobile devices are often suffered from limited screen sizes, which can only allow a small number of keys and a similarly small space for playing them. The externally connected keyboard allows more convenient way to play, along with an increased space for keys, however this is an expensive way and have a lack of portability. In light of this, a paper keyboard can prove to be adequate, as it can be easily carried around and even redrawn if needed, and no charging, wired connections or internet access is required.

Modern applications that allow for paper piano keyboard still suffer from several problems, limited keys and errors in detection keyboard or keystrokes, most of them allow for seven white keys only. This research aims to allow mobile devices to make use of a paper piano keyboard, as shown in Figure 1, with fourteen white keys and ten black keys by utilizing image processing techniques to recognize paper keyboard and use them as alternative input devices.

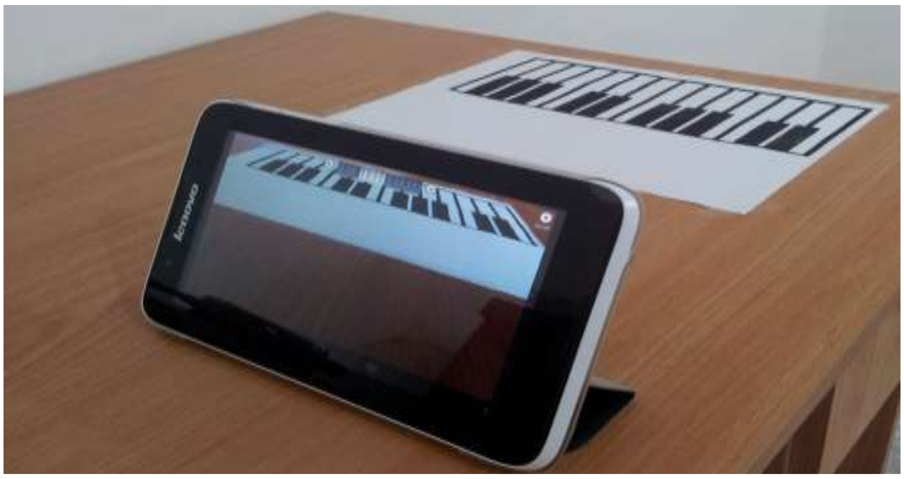

Figure 1. Paper piano

1.3 Theoretical Foundations

1.3.1 Edge Detection vs. Border Following: Edge detection aims to identify a set of points in a digital image at which the image brightness changes sharply. These points are typically organized into a set of curved line segments termed edges, so edges are computed as points that are extrema of the image gradient in the direction of the gradient. They can be thought of as the minimum and maximum points in a 1D function. In general, edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. Several edge detection techniques exist such as: Canny (as shown in Figure 2), Sobel, Laplace, and other methods [2 – 5].

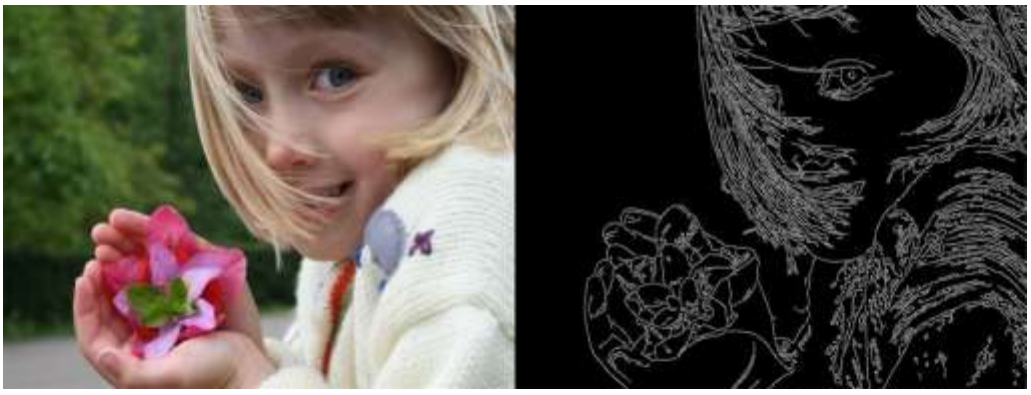

Figure 2. Canny Edge Detection Applied to a Photograph [5]

Borders, or contours, are closed curves that surround objects. Border following also known as contour tracing is a technique that is applied to digital images in order to extract the boundary of an object or pattern. The border following technique has been studied deeply, because it has a large variety of applications, including picture recognition, picture analysis, and image data compression. Borders can be obtained from edge detection methods or using border following techniques. While edge detectors can work with colored images, border following methods work mostly with binary images, also it helps in eliminating unwanted edges that might be detected if an edge detection technique is used instead [6].

1.3.2 Skin Detection: Skin detection is the fastest and the easiest among other methods used in segmentation, identification, localization, etc. It is the process of finding skin-colored pixels and regions in an image or a video, typically used as a preprocessing step to find regions that potentially have human faces and limbs in images. A skin detector typically transforms a given pixel into an appropriate color space and then uses a skin classifier to label the pixel whether it is a skin or a non-skin pixel. A skin classifier defines a decision boundary of the skin color class in the color space based on a training database of skin-colored pixels. Skin color is affected by the surrounding environment and light. For the same object, skin color could vary depending on the ambient light or the camera used, as different cameras produce different colors. In the literature of skin detection applications, a number of color spaces as well as a number of different methods have been proposed [7].

To differentiate between skin and non-skin pixels, decision rules are used to perform skin detection. A metric is introduced to measure the distance (or difference) of the pixel color to skin tone. This type of metric is defined by the skin color modeling method. Some of the methods used for skin modeling are: explicitly defined skin region, nonparametric skin distribution modeling, and parametric skin distribution modeling [8].

1.4 Open CV

Open CV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library, aims to accelerate the use of machine perception in the commercial products and to provide a common infrastructure for computer vision applications. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

There are more than 2500 optimized algorithms exist in the OpenCV library, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, classify human actions in videos, extract 3D models of objects, identify objects, track camera movements, stitch images together to produce a high resolution image of an entire scene, track moving objects, produce 3D point clouds from stereo cameras, find similar images from an image database, remove red eyes from images taken using flash, recognize scenery and establish markers to overlay it with augmented reality, follow eye movements, *etc*.[9].

OpenCV has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. OpenCV4Android is the official name of the Android port of the OpenCV library. OpenCV began supporting Android in a limited "alpha" fashion in early 2010 with OpenCV version 2.2. The first official non-beta release of OpenCV for Android was in April 2012, with OpenCV 2.4 [10].

2. Related Work

Piano augmented reality is a markerless augmented reality based piano teaching system used as a tracker of the real keyboard of the piano and draws virtual hands on the keyboard image highlighting the notes to be played. The geometrical parameters of the piano keyboard are detected to calculate the transformation matrix from keyboard coordinate to camera coordinate, instead of the marker in traditional AR system. At the beginning, all the contours of the image are extracted, from which possible contours of keyboard are found. Then the area of keyboard is identified by the structure of white and black keys predefined. The system operates at 15fps, and the average error worked out by the method was found to be about 1.97 pixels [11].

A new Android application for piano reality is introduced by [12], which the user can play a piano drawn or printed on paper. The application have too many instructions to work well, it include a piano with just seven white keys. However, as the application developers mention in its description, it isn’t compatible with all phones or Android versions it just tested on HTC devices so other phones may be problematic, due to variations in hardware.

The developers of the application say that they use just an intuition edge detection technique and thresholding with no education in image processing and not using openCV library. Reviewers on Play Store claim that it's functionally flawed; frequently failing to recognize drawn pianos, or playing wrong notes. Some reviewers complain of the app's handling of shadows, as it seems to recognize shadows as fingers, and thus interpret their movements as keystrokes.

Virtual reality piano is a new Android application similar to the one of this research, but work only on seven white keys, as shown in Figure 3, the developers of the project use image processing techniques and openCV library. The program was designed with Android 4.3 (Jelly Bean) being the intended target, and Android 2.2 (Froyo) being the minimum supported version. It was tested on a number of devices, with varying specifications, and in varying settings. Despite all changes in lighting and background, all devices exhibited the same level of accuracy in detecting fingers and drawn pianos. However, response speed varied from one device to another in a manner that is consistent with the difference in processor clock speed, the application has somehow overcome the problems of shadows and key-press detection.

The algorithm used in this research has been divided as follows: initialization, piano detection, finger detection, detecting finger location and keystroke detection.

- **Initialization:** In this phase, the captured frame is converted from RGB to grayscale, then a binary threshold is applied which reduces grayscale image into a bi-level image, as shown in Figure 3.

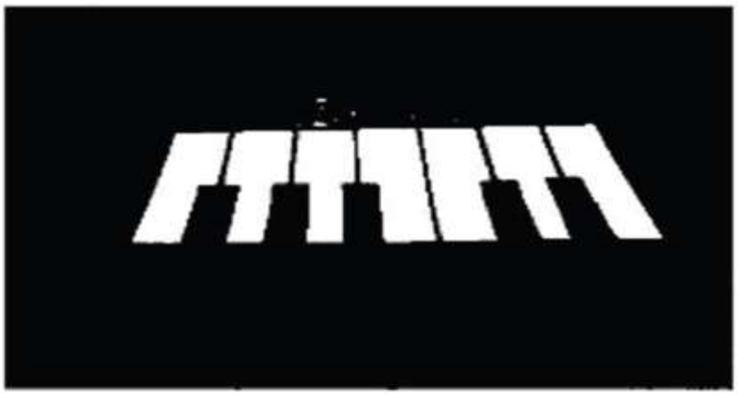

Figure 3. Binary Image

- **Piano Detection:** First, the white keys are detected by contour finding using a border following algorithm based on the work of Suzuki and Abe [13]. Those contours are approximated to polygons based on the Douglas-Peucker algorithm [14]. Second, construct a piano mask to invert the binary image and then detect contours of the black keys using the same method which applied to detect white keys.

- **Finger Detection:** To reduce the effect of shadows, a finger detection technique was used. First, the RGB captured frame is converted to HSV color space. Second, the HSV image is thresholded so that any point with a value included in the skin color range is converted to white, and any other point is converted to black, this process leaving undetected area of finger, so apply morphological transformation to remove small holes, as shown in Figure 4. Third, for each finger detected, fingertip position is determined and considered to be the effective position of that finger.

Figure 4: Finger Detection

- **Detecting Finger Location:** Each finger's effective point is tested against each key to check if it’s located inside the contour of that key using the Ray Casting algorithm [15]. This algorithm is remarkably fast, which makes it very suitable to real-time applications like the one at hand.

- **Keystroke Detection:** When the key is determined, the current effective position is checked against that of the two previous frames. A number of location-based rules is used to check for key-presses. When a key-press is detected the key number is returned.

3. Methodology

3.1 Initialization

The first frame captured from the video stream is considered to be the reference frame. Piano and keys detection is performed on this frame. Before piano detection can take place, the reference frame is converted from RGB (Red, Green, Blue) to grayscale (Eq. 1), which is required to later convert to binary. The conversion from RGB to grayscale is applied on a test image (Figure 5), and the results are as shown in Figure 7.

OpenCV handles conversion from RGB to grayscale using:

$$Y= 0.299R + 0.587G + 0.114B \qquad (1)$$

Where Y is the grayscale channel; R, G, B are the red, green, and blue channels respectively. After conversion to grayscale, a binary threshold is applied. Binary threshold function reduces a grayscale image into a bi-level image. Figure 3.3 shows the result of converting to binary according to a threshold value.

The threshold method which used in this research is OTSU's method [16]. It is automatically perform clustering-based image thresholding or the reduction of a gray-level image to a binary image. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal.

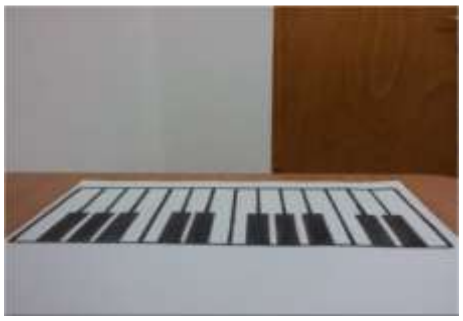
Figure 5. Original Image

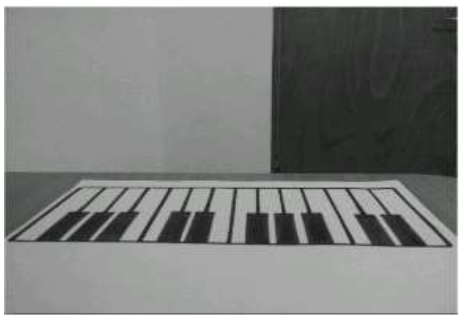
Figure 6. Graysacle Image

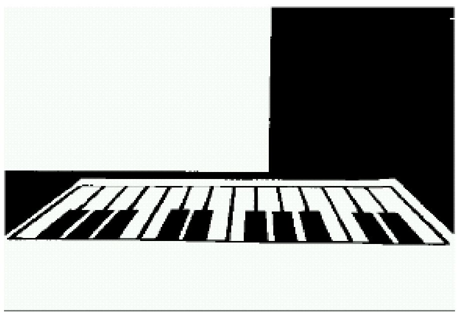
Figure 7: Binary Image

3.2 Piano Detection

The method adopted in this research for object detection to detect white keys is contour detection using the border following algorithm similar to work done by [13], as depicted in Figure 8. This method is deemed suitable for its fast performance and reliability. Those contours are approximated to polygons using the Douglas-Peucker algorithm [14]. A piano key can have no less than five sides, so any polygon that has less than five of them is discarded.

Then, the corner points for the rightmost and leftmost keys are determined. These are used as references to draw the piano mask as shown in Figure 9. A polygon is drawn to cover the piano area and serve as a mask, which is then applied to the binary reference image to get rid of piano borders and the background. The result of applying the mask to the original reference image is then inverted to detect the black keys as depicted in Figure 10.

3.3 Finger Detection

Once initialization is done, the user can start playing notes on the piano. In this stage, each captured frame is scaled down to ensure fast processing and then checked for the presence of fingers. Piano mask is applied to the frame to further reduce processing time (Figure 11). Skin detection is used to find fingers. After experimenting with a number of methods for skin detection, the method used in this research is to convert each frame into YCbCr color space; Y is the luminance component and Cb and Cr are the blue-difference and red-difference chroma component (Figure 12).

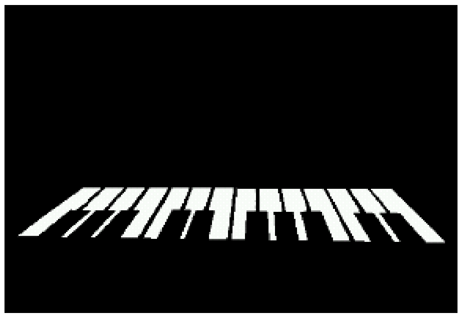
Figure 8. White Keys

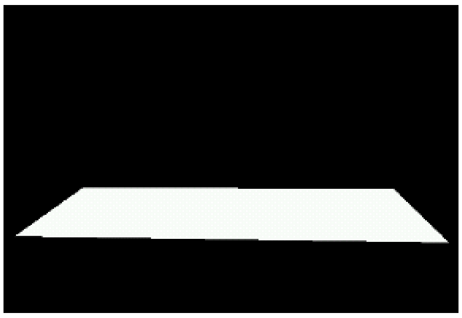
Figure 9. Piano Mask

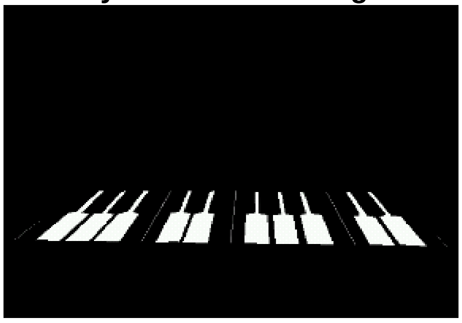
Figure 10. Black Keys

We have considered the Cr channel only for skin color detection. YCbCr yields a better results and performance under varying lightening conditions and changes in illumination against RGB and HSV color spaces [17]. Each pixel in skin color is converted into white and the others are converted into black (Figure 13), and for each key pressed, the effective point is calculated from the bottom of the used finger.

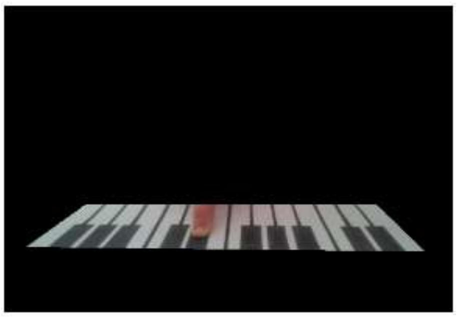
Figure 13. Mask Utilization

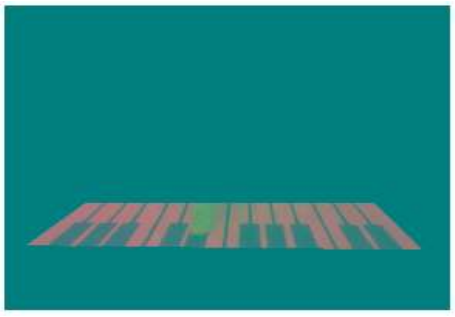
Figure 14. YCbCr Image

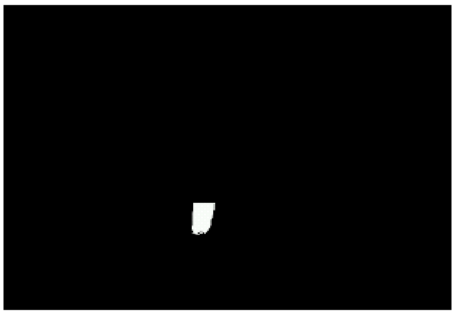
Figure 15. Finger Detection

3.3.1 Finger Location: Each finger's effective point is tested against each key to check if it's located inside the contour of that key using the Ray Casting algorithm [15]. The algorithm states that for a given point to be tested against a polygon, a ray is casted starting from that point and going in any fixed direction. If the test point is inside the polygon, the ray intersects an odd number of polygon's edges. If the point is outside, the number of intersections is even. However, if the point lies on the edge, the results might be incorrect. This is not a critical issue here as the keys are not expected to be pressed from the edges. This algorithm is remarkably fast, which makes it very suitable to real-time applications like the one at hand.

3.3.2 Keystrokes Detection: After determining finger location it checked if a key press occur or not, this done by notice the finger motion. When the key is determined, the current effective position is checked against that of the two previous frames. A keystroke occur when the effective point of a finger moved down then up again while stay on the same key, or if the effective point stayed at the same position (± a few pixels) for two consecutive frames. To avoid false keystrokes detection when use the whole hand inside the keyboard, we have chose just one finger that have the most *y*-position, considering that it's the finger expected to press on a key. When a keystroke is detected the key number is returned, and gets the correct note matched that key.

4. Experimental Results

The program is designed with Android 4.3 (Jelly Bean) being the intended target, and Android 2.2 (Froyo) being the minimum supported version. It was tested on a number of devices, with varying specifications, and in varying settings.

Playing notes on key-presses is handled by the Java part of the code. Android provides three classes that allow the app to play sounds; Media Player, Audio Track, and Sound Pool. The Media Player class can be used to control playback of audio/video resources. It is suited for music player apps and background music. The Audio Track class is useful when playing large sound files, but can play one audio resource at a time. The Sound Pool class is suited for short sound clips and can play multiple sound clips simultaneously. Comparing the three classes, Sound Pool was found to be the most suitable for the piano application. Sound Pool was found to behave inconsistently when handling WAV files. However, when those were converted to OGG files, Sound Pool performed correctly. Hence, all music notes used in the app were converted to OGG files.

The application also provides two types of recording: MIC and MIDI recording. MIC recording means that it will record all sounds in the surrounding environment and save it in mp3 file, this is done easily by open the phone MIC and recording, another type is MIDI recording which means record just the sound of piano keys played and save it in midi file, this is done by using library that allow to deal with midi files and write musical notes on it, each of the piano keys have its own pitch, velocity, duration and tick and the tempo of the file determined also which is the speed of the music.

Despite all changes in lighting and background, all devices exhibited the same level of accuracy in detecting fingers and drawn pianos. However, response speed varied from one device to another in a manner that is consistent with the difference in processor clock speed. In table 6.1, we list test devices along with their main specifications and test results, which are rated on a scale of 0 to 3:

A rating of 3 indicates lack of any perceivable lag.

A rating of 2 indicates low lag (0.5 to 1 second); playable.

A rating of 1 indicates high lag (1 to 2 seconds); only functional.

A rating of 0 indicates extreme lag; using the app is impossible.

Table 1. Application Test Results for 9 Devices

Device	OS	Clock	Camera	Result
Galaxy mini	2.3	Single-core 0.6 GHz	3.15 megapixels	0
Galaxy ace	2.3	Single-core 0.5 GHz	5 megapixels	0
Sony XperiaTipo	4.0	Single-core 0.8 GHz	3.15 megapixels	1
Galaxy s II	4.1	Dual-core 1.2 GHz	8 megapixels	3
Galaxy s III	4.3	Quad-core 1.4 GHz	8 megapixels	3
Lenovo a7-30 a3300	4.2	Quad-core 1.3 GHz	2 megapixels	2
Sony xperia z	5.0	Quad-core 1.5 GHz	13.1 megapixels	3
Galaxy note II	4.1	Quad-core 1.6 GHz	8 megapixels	3

Galaxy s III mini	4.1	Dual-core 1 GHz	5 megapixels	3

We can see that 800MHz of clock speed were sufficient to run the application, but only very slowly, while most devices with a processor clock speed of 1.0GHz or more were able to run the application with no lag.

5. Conclusion and Future Work

In this research we proposed to support mobile devices by the capability to recognize paper piano and use it as alternative input device by utilizing image processing techniques. After analyzing related work, we proposed to use border following technique instead of edge detection methods to give more powerful way in recognizing paper keyboards, and to use skin detection technique to detect fingers and precisely identify keystrokes.

The previous researches show some weakness and recommend for further future works, while in this paper we have developed the existing algorithms. Firstly, we have increased the number of keys which can be played. An OTSU's method is implemented for binarization operation, followed by the skin detection method. The YCbCr color space is chosen instead of HSV color space because it is more resistant in different light conditions and the red channel (Cr) is only selected for further processing. Secondly, we have extended the application functionality with features to record and playbacks tunes and allow playing normal piano.

Our methodology is implemented on most test devices with delays and acceptable performance, correctly identifying the keys and playing the notes with no perceivable lag in a variety of lighting and background conditions.

While our application fulfills our original vision well, there's still room for further improvement. Possible future work may focuses on:

- Adding support for multiple simultaneous key-presses; allowing users to play with more than one finger at the same time.
- Optimizing the program to further reduce load on the processor, in order to eliminate lag in less powerful devices.
- Improving the skin detection algorithm to better detect skin color in all lighting conditions.
- Further work on the keystroke detection algorithm to try to detect all keystrokes and not detect accidently keystrokes.
- Improving sounds; get sounds that are more similar to real piano and try to reduce audio files

References

[1] B. Joseph, "The Pianist's Guide to Pedaling", Indiana University Press, Bloomington and Indianapolis, **(1992)**.

[2] E. U. Scott, "Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIP tools", 2nd ed., Boca Raton, FL: CRC Press, **(2011)**.

[3] H. G. Barrow and J. M. Tenenbaum, "Interpreting Line Drawings as Three-Dimensional Surfaces', Artificial Intelligence Center, Elsevier, vol. 17, issues 1-3, **(1981)**, pp. 75-116.

[4] L. Tony, "Edge detection", in Encyclopedia of Mathematics/[ed] Hazewinkel Michiel, Kluwer Academic Publishers, Springer, **(2001)**.

[5] J. Canny, "A computational Approach to Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, **(1986)**, pp. 679-714.

[6] A. Yilmaz, O. Javed and M. Shah, 'Object Tracking: A Survey', ACM Computing Surveys, **(2006),** vol. 38, issue 4, no. 13, **(2006)**.

[7] A. Elgammal, C. Muang and D. Hu, 'Skin Detection: A Short Tutorial', Encyclopedia of Biometrics, Springer-Verlag Berlin Heidelberg, **(2009)**, pp. 1218–1224.

[8] B. D. Zarit, B. J. Super and F. K. Quek, "Comparison of Five Color Models in Skin Pixel Classification", RATFG-RTS '99 Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time System, IEEE, Washington, DC, USA, **(1999)** September 26, pp. 58 – 63.
[9] "OpenCV" [Online], **(2014)** November 15, Available from: http://itseez.com/.
[10] "OpenCV Documentation", [Online] **(2014)** November 15, Available from: http://docs.opencv.org/modules/core/doc/intro.html.
[11] F. Huang, Y. Zhou, Y. Yu, Z. Wang and S. Du, "Piano AR: A Markerless Augmented Reality Based Piano Teaching System", Third International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), IEEE, Washington, DC, USA, vol. 2, **(2011)** August 28, pp. 47 – 52.
[12] "Piano Reality", [Online], **(2014)** September 20, Available from: https://sites.google.com/site/pianoreality/.
[13] S. Suzuki and K. Be, "Topological Structural Analysis of Digitized Binary Images by Border Following", Computer Vision, Graphics, and Image Processing, Elsevier, vol. 30, no. 1, **(1985)**, pp. 32-46.
[14] D. H. Douglas and T. K. Peuker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature", The Canadian Cartographer, John Wiley & Sons, vol. 10, no. 2, **(1973)**, pp. 112 – 122.
[15] E. Haines, "Point in Polygon Strategies", Graphics gems IV Ed., Academic Press Professional, Inc. San Diego, CA, USA, **(1994)**, pp. 24-46.
[16] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Transactions on Systems, Man and Cybernetics, vol. 9, no. 1, **(1979)**, pp. 62-66.
[17] E. Rewar and S. Lenka, "Comparative analysis of skin color based models for face detection", International Journal on Signal & Image Processing (SIPIJ), vol. 4, no. 2, **(2013)**, pp. 69-75.

Authors

Ihab Zaqout, He received Ph.D. degree from Malaya University, Kuala Lumpur, Malaysia, in 2006 in computer science. He is currently an associate professor at the Faculty of Engineering and Information Technology, Al-Azhar University - Gaza, Palestine. His research interests are in image processing, information retrieval, and data mining applications.