Journal of Signal and Information Processing, 2012, *, **-**
doi:10.4236/jsip.2012.***** Published Online *** 2012 (http://www.scirp.org/journal/jsip)

Scientific Research

Human Friendly Interface Design for Virtual Fitting Room Applications on Android based Mobile Devices

Cecilia Garcia Martin and Erdal Oruklu

Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, Illinois, USA
Email: erdal@ece.iit.edu

Received August 6th, 2012.

ABSTRACT

This paper presents an image processing design flow for virtual fitting room (VFR) applications, targeting both personal computers and mobile devices. The proposed human friendly interface is implemented by a three-stage algorithm: detection and sizing of the user's body, detection of reference points based on face detection and augmented reality markers, and superimposition of the clothing over the user's image. Compared to other existing VFR systems, key difference is the lack of any proprietary hardware components or peripherals. Proposed VFR is software based and designed to be universally compatible as long as the device has a camera. Furthermore, JAVA implementation on Android based mobile systems is computationally efficient and it can run in real-time on existing mobile devices.

Keywords: virtual fitting room; face detection; augmented reality; virtual reality; human friendly interfaces.

1. Introduction

Today, mobile commerce and online sales are increasing at a rapid rate. In 2011, mobile traffic on Black Friday was 14.3 percent of all retail traffic compared to 5.6 percent in 2010 [1]. Sales on mobile devices increased to 9.8 percent from 3.2 percent year over year [2]. Nevertheless, one area online sales traditionally struggled is fashion items and clothing. It is estimated that majority of the consumers don't buy clothing online because they don't want to take any risk with the sizes. In addition, a large percentage of the purchased items are returned. This brings an additional financial burden to retail companies. Therefore, the objective of this work is to develop a virtual fitting room (VFR) application that can run on any mobile device that has a camera and network connection. This VFR application can enhance the way customers shop online and help them to choose the correct type and size of the clothing item. The proposed algorithm is designed to be computationally efficient and it can be used with existing smartphone devices, improving the way users shop online for new clothes.

In the next section, we first discuss the existing approaches for virtual fitting room applications. Section III presents the detection and sizing of the user's body. In Section IV, we present a face detection method and augmented reality markers for determining the reference points. VFR software implementation details and the user interface are shown in Section V. Finally in Section VI, JAVA based Android application development is presented.

2. Background

Several commercial products exist for VFR implementation. Styku [3] presents a body scanner that creates a complete 3D model of the user. This 3D model is then used in other webpages to try the clothing items on. The model can be rotated, it can match any size and it even uses a color map to analyze the fit. The body scanning is implemented using Microsoft's Kinect and Asus' Xtion devices. A VFR implementation by JCPteen [4] gets an image of the user and using adobe flash player displays the clothing items. At the beginning, it shows a shadow on the screen where users have to fit themselves and after that the cloth is displayed. In this system if the user is moving, the item won't follow or track him. Zugara [5] offers a VFR that is similar to the JCPteens since the items don't move once they are displayed. It is based on the augmented reality concept.

The VFR doesn't consider the proportions of the user, only shows how it looks as a fixed template. Similarly, Swivel [6] is labeled as a Try-On system that let users to see how clothes and accessories look on them in real-time. On the Ray-Ban [7] web page, there is a Virtual Mirror where a user can see how the glasses fit on him. If the user turns his head, the model fits the glasses. User only needs to download a plugin and install it. The program works based on augmented reality: At the beginning the user has to match the face within a shape and position the eyes in a line that it is shown so it takes references of the head. After that it displays the model of glasses that have been chosen. On the Google Play there is one app for Android mobile devices, Divalicious [8], called itself as a virtual dressing room with more than 300 brands. It works by changing the clothes of a default model. Finally, there is AR-Door [9] which has also has a product based on Microsoft Kinect [10]. With this system, the camera tracks the person's body and a 3D copy of clothing is superimposed on top of the users' image.

The key difference in our approach is the lack of any proprietary hardware components or peripherals. Proposed VFR is software based (Java) and designed to be universally compatible as long as the device has a camera. For the Android application, the minimum API version supported is the 14. Additionally, proposed algorithm can track and resize the clothing according to user's spatial position.

In order to create the Android app, we have developed a *human-friendly interface* [11][12][13] which is defined as an interactive computing system providing the user an easier way to communicate with the machines In particular, this can be achieved through touch-screen operations and gestures similar to what people naturally feel with their five senses. Creating intuitive interfaces with a few buttons that illustrate the basic functionality to the user is paramount for the wider acceptance of the virtual reality applications. This was one of the key objectives of this study.

3. Detecting and Sizing the Body

First step of the proposed VFR method is the acquisition of the shape of the body to get reference points. Reference points are then used to determine where to display the clothes. In order to obtain the body shape, we applied several techniques: i) Filtering with thresholding, Canny edge detection, K-means, and ii) Motion detection or skeleton detection wherein multiple frames were analyzed for any movement. However, the results were unreliable and not good enough to obtain reference points for displaying cloths.

Therefore, we introduced a new detection methodology based on locating the face of the user, adjusting a reference point at his/her neck and displaying the clothes based on that point. In addition, another point of reference can be obtained by using an Augmented Reality (AR) marker. Details of this algorithm are explained in Section IV.

For obtaining the size of the user, we follow a similar automated body feature extraction technique as shown in [14]. The idea is to set up the user in front of the camera and hold him at the beginning at a certain predetermined distance. The algorithm extracts points on the shoulders and the belly. Measuring the distance between these points and knowing the distance from the user to the camera, the size of the user can be obtained. When the image (video frame) is acquired, a Canny edge detection filter is applied to obtain only the silhouette of the body. Canny edge detection is really susceptible to noise that is present in unprocessed data; therefore it uses a filter where the raw image is convolved with a Gaussian filter. After convolution, four filters are applied to detect horizontal, vertical and diagonal edges in the processed image. Morphological functions are also applied to obtain a closed silhouette. Finally, an 8-points Freeman chain code, shown in **Figure 1** is applied to assign a direction to each pixel.

deltax	deltay	8-code	corresp 4-code
0	1	0	0
-1	1	1	
-1	0	2	1
-1	-1	3	
0	-1	4	2
1	-1	5	
1	0	6	3
1	1	7	

Figure 1 - Freeman's codification

We can choose to apply 8 or 4 chain code, then the following formula can be used:

$$z = 4*(deltax + 2) + (deltay + 2) \quad (1)$$

which gives the sequence corresponding to rows 1-8 in the preceding table: $z = \{11,7,6,5,9,13,14,15\}$. These values can be used as indices into the table, improving the speed of computing the chain code. Each variation between consecutive numbers represents a variation of 45º so if the difference of direction between consecutive points is measured and if the change is more than two (90º) then a feature point is detected and marked in the image.

$$e_k = |d_{j+1} - d_j| = 2 \quad (2)$$

This is the same than saying that the absolute difference between two points is bigger than 2 as Eq. (2) states.

Finally the distance between them is measured in the image and related to the distance from user to the camera to obtain the size. **Figure 2** shows an example of feature points extraction and the red line that should be taken for the measure.

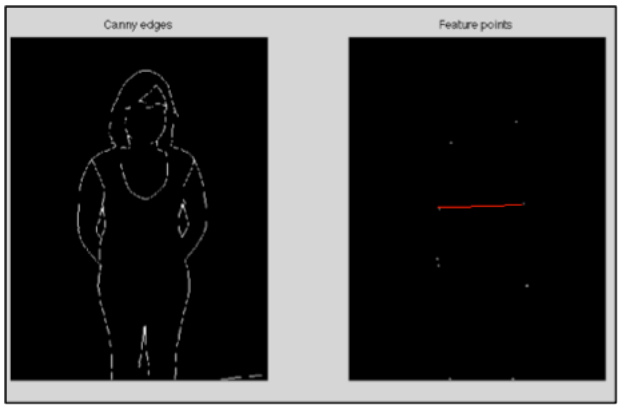


Figure 2 - Feature points extraction

4. Reference Points and Cloth display

A reference point is required to determine where the user is located. Using face detection, the neck (point of reference) is easily detected and the cloths can be fitted automatically for the user. On the other hand, by using an AR marker (point of reference), a user will have more freedom to choose how the cloths fit on her/him. Both reference points can be obtained and displayed by using the OpenCV (Open Source Computer Vision) library [15]. OpenCV is a library of programming functions for real time computer vision applications. This library provides a comprehensive computer vision infrastructure and thereby allows users to work at a higher abstraction layer. Additionally, library functions are optimized for fast and efficient processing.

4.1. Face Detection

In order to detect faces, we use the Haar-like features [16][21] that are digital image features used in object recognition. Other face detection approaches in the literature include methods using OpenCV [17], rectangular features [18] as well as improvements to make the algorithms faster for hardware implementation [19][20].

The Haar-like features are so called because they are computed similar to the coefficients in Haar wavelet transforms. A set of these features can be used to encode the contrasts exhibited by a human face and their spatial relationships. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in these regions and calculates the difference between them. This difference is then used to categorize subsections of an image by comparing it to a learned threshold that separates non-objects from objects. Since a Haar-like feature is only a classifier, a large number of Haar-like features are necessary to describe an object with sufficient accuracy. The database to obtain a strong learner for the object that we want to detect and where the Haar-like features are organized is called classifier cascade.

In the case of proposed VFR algorithm, the encoded contrasts are the contrasts of the human face as well as their spatial relationships. This classifier needs to be trained with hundreds of samples of a particular object, which will represent the positive examples. Negative examples are also trained with samples that are not considered as the object to detect. All the samples must have the same size (for example 20x20). The classifier must be easily resizable to be able to detect the object with different size in the image. Therefore, the matching procedure has to be executed on several scales.
OpenCV uses an xml file that contains all of the characteristics to detect the face. This xml file is read by the function "cvHaarDetectObjects" and it is compared with a region of interest of the input image and the classifier returns 1 if the object is detected, 0 otherwise. If every simple classifier is positive, the cascade classifier is positive, otherwise it is negative. In other words, the face detection is made with a sum of these detected samples in a predefined position. Once the face is detected a rectangle is plotted around the face and the location of a reference point is chosen based on the supposition that the neck is placed at the middle of the head, half of the rectangle's height. Also it is taken that it measures approximately a third of the head height, hence a third of the rectangle height. Note that only one face will be detected and it will be the one closest to the camera.

4.2. Marker Detection

An augmented reality marker is used to display (superimpose) the cloths over the users' image. In order to detect the marker and obtain a reference, an algorithm with seven stages has been used.

- *Divide image in regions:* When the image is received, it is divided into regions of 40x40 pixels and there are horizontal and vertical scan lines every 5pixels.
- *Detect edges in regions:* For each region, a Gaussian derivative filter is used to detect the black/white edges. The filter used is [-3 -5 0 5 3]*A. Once an edge is detected a Sobel operator is

used to determine the orientation of the edges. The Sobel operator used is:

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A, \qquad G_x = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 0 & 2 \\ 1 & 0 & 1 \end{bmatrix} * A \tag{3}$$

In **Figure 3** we can see the edges in blue if they are vertical and green if they are horizontal.

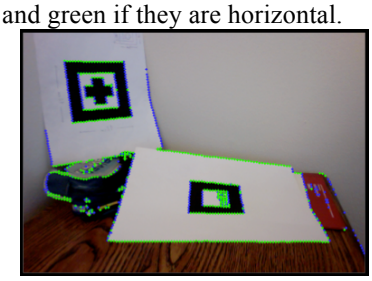


Figure 3 - Detected edges after Sobel operator

- *Find Segments:* A Random Sample Consensus grouper algorithm is then used to create the line segments in the regions. This algorithm groups series of points that can be fitted into lines. First, it arbitrarily takes 2 points of the same region with the same orientation, then the close points that have a compatible orientation are added to the line segment. Finally, lines containing at least 4 segments are considered as a detected line. This can be observed in **Figure 4.**

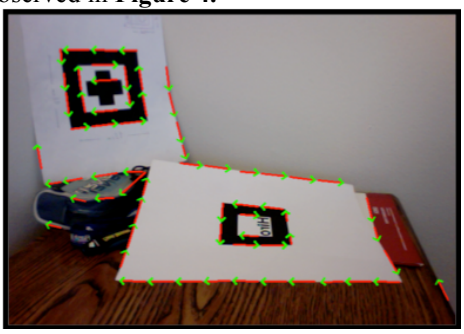


Figure 4 - Detected lines

- *Extend lines along edges:* So far only pixels on scan lines were scanned. Now, the lines are extended pixel by pixel until a detected corner or until there is no edges. The green lines displayed on **Figure 5** represent the extension of the previous red lines from one corner to another corner.

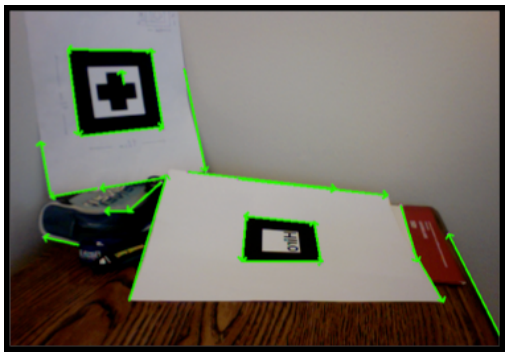


Figure 5 - Extended lines from one corner to another

- *Keep lines with corner:* The lines with at least one corner are kept.
- *Find markers:* Chains of 3 or 4 lines are kept. A chain of line is found when the end of one line corresponds to the beginning of the second one. The rectangles with a black inside are checked as markers. The two markers on **Figure 6** are now detected. However, we want to detect only our marker.

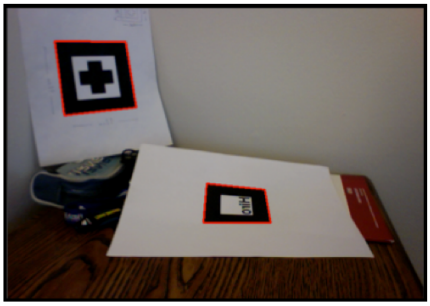


Figure 6 - Markers Detected

- *Identify markers:*_ The last step is to identify the inside of the marker to check that it is our marker. As we can see on the left picture, this step only checks that the marker has a black center and is white on the inside of the boundaries as we can see in **Figure 7**.

Figure 7 - Points recognized to identify the marker.

- The result of all these steps combined is shown on **Figure 8**. We can see that only our marker is detected and we have the desired reference point.

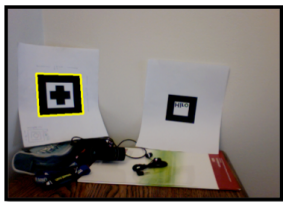


Figure 8 - Result of marker detection

Keeping the marker in a position near the belt or even using one with the AR marker allows obtaining the reference point and can properly place clothes. This also would be comfortable to the user because he will have free hands.

4.3. Cloth Display

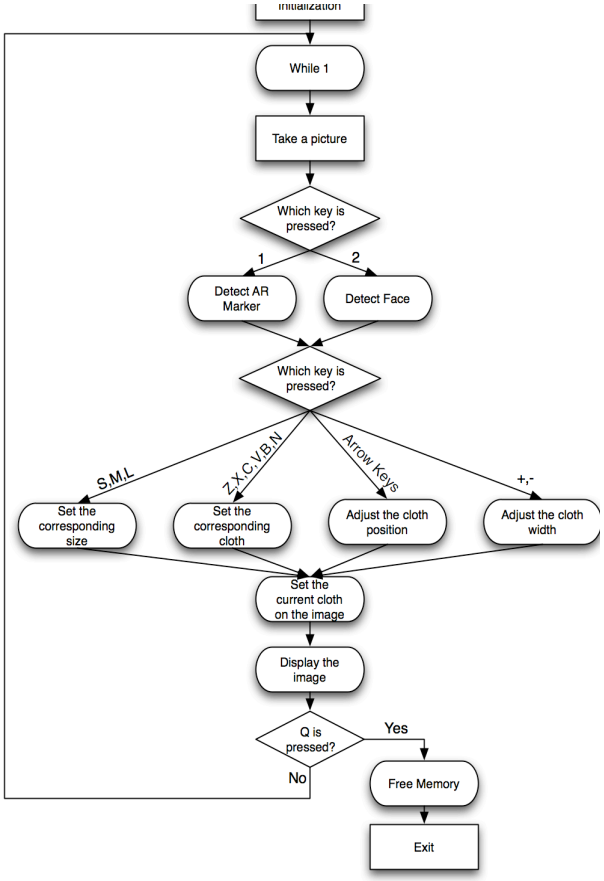
Cloth masks are needed to determine which pixels should be displaying the clothes and which ones not. Each cloth should have a mask; there is an example of a mask in **Figure 9.** A stabilization method is used since face detection provides a rectangle for identification that moves quickly in position and size. As a consequence, the mean of 5 previous images is shown.

Figure 9 - Cloth and its mask

4.4. VFR Application Flow Chart

Figure 10 shows the flowchart of the VFR functionality. The algorithm includes several options to change parameters with user input. By pressing 1 or 2, the detection method is changed (1 for AR marker, 2 for face detection). Choosing between S, M and L changes a coefficient and it changes the size of the selected item. By pressing a key between Z and N the user will be able to try different items available in the fitting room. + and – make the width of the cloth bigger or smaller. Finally, with the arrow keys, incremental changes can be done and the position of the item can be modified to make it more accurate. **Figure 11** shows a typical VFR application example running on a laptop computer with webcam.

5. VFR Implementation and Interface



For universal compatibility across different mobile devices, we developed a Java Applet [22], which presents an ideal solution to enable every customer to be able to run the Virtual Fitting Room. Java provides easy functions to create a graphical user interface to select size, different cloths and adjust the position of the clothes. On the other side, the OpenCV code needs to be adapted from C++ to Java [25]. In order to be able to execute the algorithm in Java, JavaCV is used which is a java wrapper for OpenCV library [23][24]. JavaCV includes the commonly used libraries by researchers in the field of computer vision, such as OpenCV, FFmpeg, libdc1394, PGR, FlyCapture, OpenKinect, videoInput, and AR-ToolKitPlus. Hence, the same functions that had been used in C++ can be now used in Java. JavaCV also comes with hardware-accelerated displays, easy-to-use methods to execute code in parallel on multiple cores, user- friendly geometric and color calibration of cameras and

Figure 10 - VFR Flowchart

projectors, and detection and matching of feature points. The final look of the VFR java applet and the user interface can be seen in **Figure 12**.

Figure 11 - VFR example

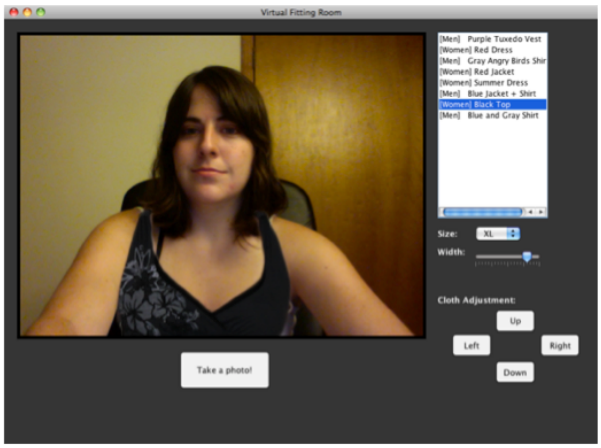


Figure 12 - VFR graphic interface

6. Android Based App Development

The Android app has been developed using Eclipse and Java. Android SDK [26] used in combination with ADT plugin in Eclipse present a flexible environment to built and test any Android applications. This app is available for any device with front camera but will work at full with two cameras.

Once the Android VFR app is started, the user can choose between different options. First he has to select the size, between XSmall, by default, and XLarge. After that he selects the clothing that he wants to try. The app detects the face of the user and it displays the cloth using as reference a rectangle that is drawn around the face of the user (Green rectangle in Figure 13). The size of this rectangle depends on the user's face: if the user is close to the camera the rectangle will be bigger, on the other hand if he is farther it will be smaller. By using this rectangle and references, the clothes that are going to appear in the screen are scaled by the app.

The maximum distance that the user can reach with his arm sometimes isn't enough to obtain a clear view so here the app offers two possibilities. One is leaving the phone in a fixed position and make himself fit into the screen and see how the clothes fit on him or use the button to switch to the back camera and let someone else hold the phone and take a picture of the user to be able to check how the cloth looks in him.

The clothes that are displayed on the screen follow and track the user similar to desktop computer implementation shown in Section V. In order to calculate the position where the image has to be displayed, the measurements from the face rectangle and the image width and height has been used. The image is displayed setting the origin, reference point, at the top-left corner. X coordinate is obtained by acquiring the X center of the rectangle and subtracting half of the image's width. Y coordinate is obtained starting from the bottom point of the rectangle and adding one third of the image's height. This is represented in **Figure 13**. The equations applied are:

$$posX = Rect.centerX - imageWidth/2 - clothes[index].offsetX$$

$$posY = Rect.bottom + clothes[index].offsetY \quad (4)$$

Both equations have an offset in X and Y coordinates since the images used may have different sizes.

As expected, the camera follows the user in real time. In order to implement the face detection, the Android's face detection API is used, released in the Android 4.0 SDK, which can detect the face in the previous frame and specify facial features as the position of the eyes or the mouth. To use this, a *FaceDetectionListener* has been implemented and it returns an array containing all the faces that have been detected with their characteristics.

A transformation has to be applied to the rectangle's coordinates that come from (-1000, -1000) top-left to (1000,1000) bottom-right in order to be able to adjust it to any android screen. Using the API has been chosen over using JavaCV and the Haar-like classifier because this one has been optimized for Android systems. To adjust the clothes, it has been settled as *ImageView* and displayed with *addView()* method from that class.

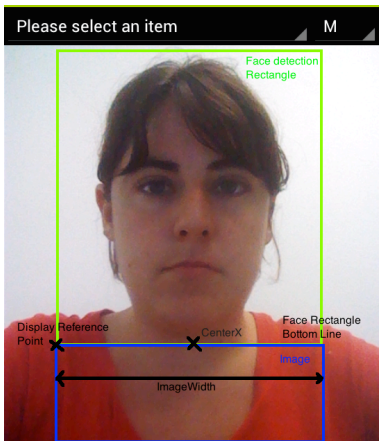


Figure 13 - Calculations for the position

The VFR app works using *activities*; these are single and focused "actions" that the user can do. Each activity has its own lifecycle and as soon as the app is started the main activity comes to the foreground of the system. In **Figure 14**, the complete lifecycle of an activity can be seen. The visible lifetime of the app happens between *onStart()* until a call to *onStop()*. The foreground lifetime, interacting with the user, occurs between *onResume()* and *onPause()*. It has to be noted that we have to declare in the *AndroidManifest* all the hardware that it is going to be used in our device as well as the fundamental characteristics of our app. The user interface has been created in a hierarchical way using View and different Layouts provided.

In order to show the camera, a class *CameraPreview* extending *SurfaceView* has been used to draw the camera surface embedded inside a view hierarchy. This class works as a secondary thread that can render into the screen and also has an activity cycle as the Activity does. In this case, it follows *SurfaceCreated()*, *SurfaceChanged()* and *SurfaceDestroyed()* where *Created()* and *Destroyed()* sets the visibility of the window and *Change()* looks for any change on it.

In the main activity of the VFR, there are the listeners for the user interface buttons as well as the method to adjust the position of the clothes depending on the face position. In **Figure 15**, the final view of the Android VFR can be seen.

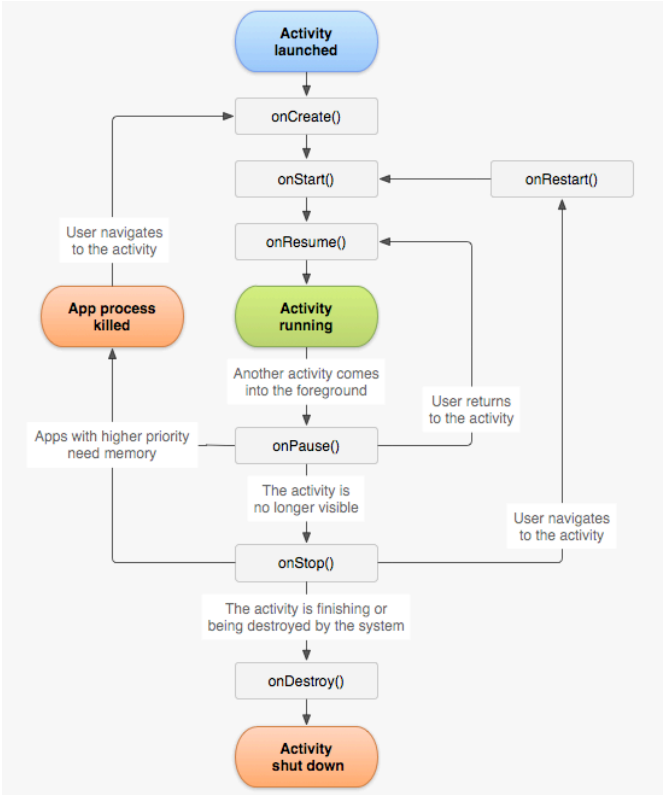


Figure 14 - Android Activity lifecycle

Figure 15 - Final view of Android VFR

Finally, the performance of the application has been analyzed using the debugging tool called DDMS (Dalvik Debug Monitor Server). When DDMS starts, it connects to Android Debug Bridge (adb). When a device is connected, a VM monitoring service is created between adb and DDMS, which notifies DDMS when a VM process is initiated or the changes that have been done through an assigned port that usually is the 8600.

In order to measure the performance of the tool, the method profiling has been started. This tool tracks certain metrics about a method, such as number of calls, execution time, and time spent executing the method. Here two panels are obtained: the timeline panel that describes when each thread and method started/stopped and on the other hand the profile panel that provides a summary of what happened inside a method. With the profile panel comes a table that shows exclusive time, that is the time spent in a method and the inclusive time that is the time spent in the method plus the time spent in any other called method (child).

Based on this profiling, face detection algorithm takes only 0.1% exclusive time and 13.7% of inclusive time. This is due to the fact that its child has to display the cloths depending upon the position of the face. The method for drawing (i.e., displaying) clothes shows an exclusive time of 0.2% and 29.7% of inclusive time. Therefore, it can be seen that the most expensive processes, in terms of computation time, are the ones related with image displaying.

7. Conclusion

In this work, a virtual fitting room application for mobile devices was implemented successfully. The main objective to obtain a real time, platform independent application was achieved. Users are able to select sizes from XS to XL and chose between different cameras on the device to implement the VFR. In addition, the algorithm can track and scale the clothing according to user's position and movement. By deploying it to the Android Market or Apple Store, this application can be used by retail companies for increasing their online presence.

REFERENCES

[1] IBM Coremetrics Benchmark Reports, Available at: http://www-01.ibm.com/software/marketing-solutions/benchmark-reports/index-2011.html. Accessed on Feb 10, 2012.

[2] IBM Press release, November 29, 2011. Available at: http://www-03.ibm.com/press/us/en/pressrelease/36113.wss

[3] Skytu, http://www.styku.com/business/

[4] JCPteen, http://www.seventeen.com/fashion/virtual-dressing- room

[5] Zugara, http://zugara.com/

[6] Swivel, http://www.facecake.com/swivel/index2.html.

[7] RayBan, http://www.ray-ban.com/usa/science/virtual-mirror

[8] Divalicious, http://www.divaliciousapp.com

[9] Topshop, http://ar-door.com/dopolnennaya-realnost/?lang=en

[10] Microsoft Kinect, http://www.microsoft.com/en-us/kinectforwindows/

[11] M. Popa, "Hand gesture recognition based on accelerometer sensors", *International Conference on Networked Computing and Advanced Information Management*, pp. 115-120, June 2011.

[12] J. Liu, "A new Reading Interface Design for Senior Citizens", *Instrumentation, Measurement, Computer, Communication and Control*, pp. 349-352, October 2011.

[13] N. Kubota, D. Koudu, and S. Kamijima, "Human-Friendly Interface Based on Visual Attention for Multiple Mobile Robots", *Automation Congress World*, pp. 1-6, July 2006.

[14] Y. Lin, Mao-Jiun and J. Wang, "Automated body feature extraction from 2D images", *Expert Systems with Applications*, vol. 38, no. 3, pp. 2585-2591, 2011.

[15] Intel Inc., "Open Computer Vision Library". http://opencv.org/

[16] Haar-like feature: http://opencv.willowgarage.com/wiki/FaceDetection

[17] M. Zuo, G. Zeng, and X. Tu, "Research and improvement of face detection algorithm based on the OpenCV", *International Conference on Information Science and Engineering (ICISE)*, pp. 1413-1416, December 2010.

[18] D. Lee, "A face detection and recognition system based on rectangular feature orientation", *International conference on System Science and Engineering(ICSSE)*, pp. 495-499, July 2010

[19] L. Acasandrei, and A. Barriga, "Accelerating Viola-Jones face detection for embedded and SoC environments", *2011 Fifth ACM/IEEE international conference Distributed Smart Cameras (ICDSC),* pp. 1-6, August 2011.

[20] S. Rigos, "A hardware acceleration unit for face detection", *Mediterranean Conference on Embedded Computing (MECO),* pp. 17-21, June 2012.

[21] C. Chai, and Y. Wang, "Face detection based on extended Haar-like features", *International Conference on Mechanical and Electronics Engineering (ICMEE*), pp. 442-445, August 2010.

[22] Java documentation: http://download.oracle.com/javase/6/docs/api/

[23] S. Audet, "Java interface to OpenCV", accessed July

2011. http://code.google.com/p/javacv/

[24] Java Wrapper: http://code.google.com/p/javacv/

[25] S. Audet, Hints for Converting OpenCV C/C++ code to JavaCV http://code.google.com/p/javacv/wiki/ConvertingOpenCV

[26] Android developers http://developer.android.com/develop/index.html

