Hand Pose Estimation for Markerless Interaction with Augmented Reality Applications

César Osimani[†1], Emiliano Kohmann[†2]
[†]Applied Research & Development Center on IT (CIADE-IT)
Universidad Blas Pascal (UBP)
Córdoba, Argentina
[1]cosimani@ubp.edu.ar
[2]ekohmann@ubp.edu.ar

***Abstract*— Augmented Reality is viewing the real world combined with virtual computer-generated elements, whose fusion leads to a mixed reality. It is important that these elements are aligned within the scene in a precise manner according to the type of application. Achieving this may require the identification of the orientation of a marker augmented reality or estimation of the perspective of the real scenario.**

This paper presents a markerless system appropriate for camera pose estimation in augmented reality applications. We propose a selection and detection of hand feature points to estimate their pose in the real world with the aim of rendering 3D virtual content aligned with the hand. Also, we expose a method for interacting with the application through movements of two fingers, particularly putting them together. This action exchanges the virtual content displayed, chosen from a set of 3D objects available in the system. To sum up, users can manipulate the virtual contents by using their hands.

***Resumen*— La Realidad Aumentada es la visualización del mundo real combinado con elementos virtuales generados por computadora, cuya fusión da lugar a una realidad mixta. Es importante que estos elementos virtuales sean alineados dentro de la escena de una manera precisa de acuerdo al tipo de aplicación. Lograrlo puede requerir identificar la orientación de un marcador de realidad aumentada o estimar la perspectiva del escenario real.**

Este trabajo presenta un sistema de realidad aumentada que realiza la estimación de la pose de la cámara. Se propone una selección y detección de ciertas características de la mano para estimar su postura en el mundo real con la finalidad de graficar contenido virtual 3D alineado con las manos. Además, se expone un método que permite al usuario interactuar con el sistema a través del movimiento de dos dedos, particulamente juntando los mismos. En este trabajo, esta acción realiza el intercambio del contenido virtual visualizado, elegidos desde un conjunto de objetos 3D disponibles en el sistema. En resumen, los usuarios pueden manipular el contenido virtual con sus manos.

***Keywords*— Augmented reality; Hand pose estimation; 3D virtual contents**

I. Introduction

Augmented Reality refers to a realistic integration of virtual elements into the sequence of images in real time. One of the aims of augmented reality is to improve understanding of real world providing additional information that could be used for example in marketing, education or entertainment. This augmentation of real scene requires image processing to solve position and orientation of a real element which must be aligned with the virtual element to achieve the augmentation of the reality. There are two main kinds of methods for augmented reality: markers based method and method without markers.

The method based on markers needs artificial elements into real scene to help the visual tracking and pose estimation. Those artificial elements (called fiducial markers) are easy to detect and contain codes to distinguish from one another. An open source library called ArUco [1] was used for a development of our own based on fiducial markers (briefly explained in Section II).

The method without markers uses natural features existing on real scene, such as corners, edges, line segments of real objects and, why not, user's body parts as well. In this paper we propose a method without markers based on hand pose that allows to recognize its orientation in 3D space to align virtual content on the palm of the hand. Furthermore, our method recognizes simple finger poses to interact with this virtual content.

II. Related works

A fiducial marker system uses the markers based method and it is composed by a set of valid markers and an algorithm which performs its detection. Several fiducial markers have been proposed, one of them is ArUco. It is a library for Augmented Reality applications based on OpenCV [2] for tracking known square physical markers as shown in Figure 1. ArUco operates by tracking the user's viewpoint in the real world from each frame of a video sequence and uses it to draw virtual object on the marker. The processed frame is then rendered back on the display as a background while augmenting the virtual object at the exact location of the marker to produce a seamless see-through effect video sequence. We have implemented ArUco in our augmented reality application (its source code is available online [3]).

Regarding the methods without markers, the technique presented in [4] tracks the position of hands to insert virtual content over the hand into de real scene. First, the hand is identified and isolated in the video through two techniques: Technique of detection of the skin (using YCbCr color space) to remove the sections without color of skin, and

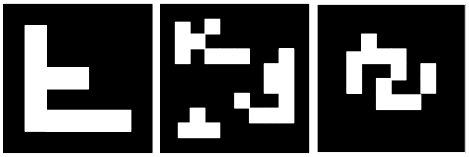

Figure 1. Examples of square fiducial markers

neural network to determine areas that correspond to a hand. The work in [5] proposes a pose hand estimation algorithm based on feature points which must be both coplanar and non-collinear for correct planar pose estimation. The position of fingertips and finger valleys are detected using the contour of the hand. In [6], authors proposed an augmented reality system based on the user's hand that can track the outstretched hand. And the fingertips are also detected using an algorithm based on the contour of the hand. The contour point with a high curvature value is sought as a candidate fingertip point.

Based on these observations, we may note that good results are obtained through the following methods: hand segmentation using skin color detection, hand features points detection using shape analysis and pose estimation through perspective projection transformation. We propose combine these methods to estimate the hand pose in 3D space. First, the hand is isolated using skin color detection, on condition that the background has uniform color. The following step is to identify fingertips and finger valleys through hand shape analysis, and then obtain hand features points that must be coplanar and non-collinear. These points are selected in such a way that they cover the whole palm and then this allow to draw 3D elements in the scene. In addition, an method that emphasizes the interaction between real and virtual word is proposed.

The rest of this paper is structured as follows: In Section III, the method is described in detail, skin color detection, hand shape identification, feature points extraction, pose estimation and drawing virtual elements. In section IV, we show experimental results of the system and present examples of 3D scenes employing the hand's user. We discuss benefits and limitations of our method in Section V. The conclusions are presented in Section VI.

III. Method Description

The proposed approach intends to achieve an Augmented Reality System based on a real component of the scene (on this occasion we use the user's hand) that means without introducing fiducial markers. The overall flow of the system presented in this paper is as illustrated in Figure 2. The detailed descriptions of the methods used are included in the subsections that follow.

A. Skin Color Detection

For simplicity, we assume that this system will be used over a desk. So, only a hand appears at the scene. This allows to detect a hand very easily through skin color detection. To do this, we propose the *Lab* color space which is a model with

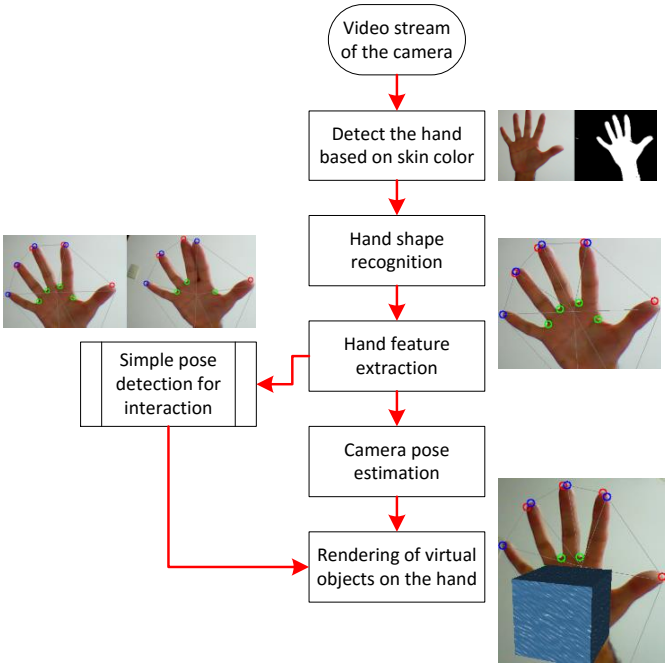

Figure 2. Workflow of the proposed system

a linear representation according to human color perception. The dimensions in *Lab* color space are the luminosity L and the two color components a and b, covering red to green and blue to yellow axes respectively.

The conversion from RGB to *Lab* requires first converting the color space XYZ. The formulas are listed as follow:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$X \leftarrow \frac{X}{X_n} \quad \text{where } X_n = 0.950456$$

$$Z \leftarrow \frac{Z}{Z_n} \quad \text{where } Z_n = 1.088754$$

$$L \leftarrow \begin{cases} 116 * \sqrt[3]{Y} - 16 & \text{for } Y > 0.08856 \\ 903.3 * Y & \text{for } Y \leq 0.08856 \end{cases}$$

$$a \leftarrow 500(f_{(X)} - f_{(Y)})$$

$$b \leftarrow 200(f_{(Y)} - f_{(Z)})$$

where

$$f_{(t)} = \begin{cases} \sqrt[3]{t} & \text{for } t > 0.08856 \\ 7.787\ t + \frac{16}{116} & \text{for } t \leq 0.08856 \end{cases}$$

All captured images are converted from RGB to *Lab* color space. Each pixel is then classified as either skin color pixel or non-skin color pixel based on a fixed range of component a of *Lab* color space. Good results are obtained by selecting the range between 109 and 133 for component a. The binary

images are obtained as follows:

$$dst_{(x,y)} = \begin{cases} 255 \text{ (white)} & \text{if } 109 \leq a_{(x,y)} \leq 133 \\ 0 \text{ (black)} & \text{otherwise} \end{cases}$$

where

$dst_{(x,y)}$ is pixel value of binary image on (x, y)

$a_{(x,y)}$ is component a pixel value of input image

With this we isolate the hand to generate a binary image (Figure 3) and then continue with the hand shape identification.

Figure 3. Skin color detection and binary image

In order to reduce the complexity for the detection of hands, the video camera is used to capture images with a white background where only the hand and forearm come into play. If conditions do not allow to have a uniform color background, it would be necessary to use background substraction methods to remove the background and possibly also use methods that use descriptions of the visual features such as HOG (Histogram of Oriented Gradients).

Background subtraction is a widely used approach for detecting moving objects in videos from static cameras with which an image's foreground is extracted. It detects the moving objects from the difference between the current frame and a reference frame, often called "background image". Its use can be applied to the detection of moving objects such as cars on the road [7], and our work can take advantage of it for including complex backgrounds [8].

Another hand detection method which can improve the accuracy with complex backgrounds is that referred above HOG. In the work [9] HOG is used to identify hand gestures of alphabets. This method increases the accuracy in segmenting hand and avoids errors caused by using skin color detector when the region of interest extracted includes forearm.

B. Hand Shape Identification and Feature Points Extraction

Hand geometry extraction begins by using the OpenCV `findContours` function which returns a set of points that form the contours in the binary input image. Some contours may be contained within others. However, only the outermost contours are retained and they are filled in with a solid color. This way, a new binary image is created which works like a mask to obtain a delineated image of the hands.

The points on the convex hull of this image are likely to be fingers. However there will also be other convex points because part of the arm may appear in the image. One way to identify the convex points corresponding to finger tips is to make use of convexity defects. The points that make up the convex contour are found by making use of the `convexHull` function, and the points in the convexity defects are calculated using the `convexityDefects` function. The convex hull includes the contour of the hand and by selecting the points in each segment of it that are separated the most from this hull, the points forming the bottom of the spaces between the fingers are found. An example is shown in Figure 4.

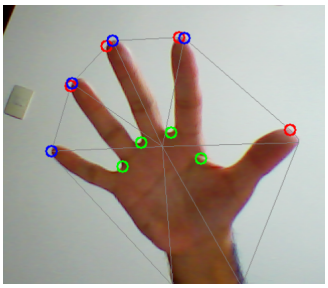

Figure 4. The convex hull and convexity defects

One advantage of this finger detector is that the algorithm is very simple, and its disadvantage is its low precision in determining the number of extended fingers.

To avoid false positives in determining the number of fingers extended, a condition for the depth of the concavity and length between vertex of convex hull is defined. This condition eliminates the possibility of detecting finger valleys where there are none, e.g. in the case shown in Figure 5. The standard deviation of the concavity depth is calculated:

$$s_{depths} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(d_i - \bar{x})^2}$$

where

$\{d_1, d_2, \ldots, d_N\}$ are the depth values (see Figure 5)

N is the size of the sample

$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} d_i$ is the mean value

In addition, the standard deviation of the distances between each vertex of the convex hull is calculated (denoted by $s_{distances}$).

Thus, we take the valleys of the fingers for which the following condition is met:

$$depth_i > s_{depths} \wedge distance_i < s_{distances}$$

In our system [3] we used fiducial markers where the its four vertexes are benchmarks. Thus it is possible to identify the plane where the marker is located. In this paper, the palm is the benchmark, and we could then imagine a marker at the center of the palm.

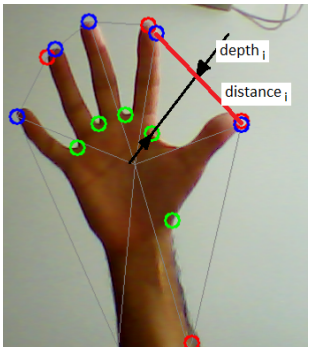

Figure 5. Condition to avoid detection of the wrist instead of finger valley

C. Hand pose estimation

Given a set of m points corresponding to

$$q_m \leftrightarrow Q_m$$

where q_m denotes a homogeneous 2D coordinate of a point in the camera image plane

$$q_m = \begin{bmatrix} x_m & y_m & 1 \end{bmatrix}^T$$

and Q_m denotes the corresponding 3D homogeneous coordinate of the same point in the physical world

$$Q_m = \begin{bmatrix} X_m & Y_m & Z_m & 1 \end{bmatrix}^T$$

q_m and Q_m are related by so-called pinhole camera model which describes the mathematical relationship between the coordinates of a 3D point and its projection onto the image plane using a perspective transformation (see Figure 6).

$$q_m = A \begin{bmatrix} R|t \end{bmatrix} Q_m$$

or

$$\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}$$

where

A is a camera matrix, or a matrix of intrinsic parameters

f is the focal length

(c_x, c_y) is camera centre point

$\begin{bmatrix} R|t \end{bmatrix}$ is the matrix of extrinsic parameters

The matrix $\begin{bmatrix} R|t \end{bmatrix}$ is used to translate coordinates of a point (X, Y, Z) to a coordinate system, fixed with respect to the camera. This matrix describes the camera's location in the world, and what direction it's pointing. Those familiar with OpenGL know this as the "view matrix" or "modelview matrix". It has two components: a rotation matrix R and a translation vector t, but these don't exactly correspond to the camera's rotation and translation. This matrix describes how to transform points in world coordinates to camera coordinates. The vector t can be interpreted as the position of the world origin in camera coordinates, and the columns of R represent represent the directions of the world-axes in camera coordinates.

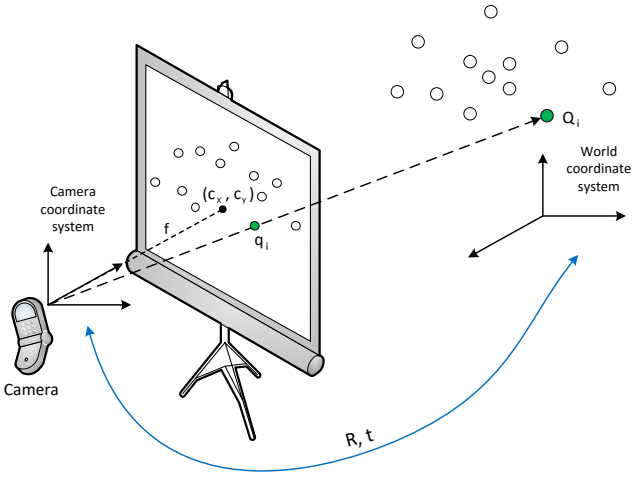

Figure 6. Relationship between a 3D point and its projection onto a plane

Considering the palm as a 2D plane positioned perpendicular to z axis whose coordinate in z axis is 0, the 3D coordinates of all the feature points of the hand can be calculated. This will be used as the model points during the pose estimation process. Our system uses nine feature points (five fingertips and four finger valleys). Note that, the points calculated are from the user's hand that will be used for experiments. Different users might have different measurement model points of the hand. With these nine hand feature points we estimate the transformation matrix of the camera.

D. Simple pose for interaction

In order to allow the user can interact with the system, a finger valleys counter is implemented. We know that four finger valleys are detected in an outstretched hand and three valleys when two fingers are together (see Figure 7). In this way we want to approach to idea of manipulate the virtual objects like if we really hold them in our hands. The closest thing to feeling that we could touch and manipulate them. In this work, this detection of two fingers together is performed to switch between a graphed 3D element and other.

E. Graphics Rendering Process

We use OpenGL (Open Graphics Library) for the representation of 3D objects. OpenGL is a 3D graphics software package in common use, based on computer graphics, consistent with the principles of optics and vision. It makes available to the programmer a small set of geometric primitives (points, lines, polygons, images, and bitmaps) and provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered [10].

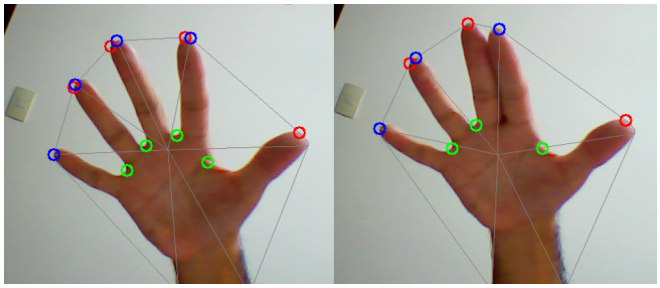

Figure 7. Finger valleys counter for interaction

The 3D objects we used can be classified into some categories: User defined simple objects, GLUT objects, and complex three-dimensional models. GLUT is OpenGL Utility Toolkit which includes a number of functions for create a set of 3D objects such as sphere, cubes, cones and many others. Each GLUT object has associated properties or attributes such as radius of a sphere and size of a cube. Each object can also be rendered in a wireframe form or a solid-like object. The professional three-dimensional modeling software has strong modeling capabilities and modeling vivid. There is a proposal [11] to use the 3dsMAX to create complex three-dimensional models and export the models created in 3ds file format, then import the models into OpenGL program for interactive controlling and rendering, and accomplish the creation of the virtual scene together. In the Figure 8 we show the 3D objects of our system.

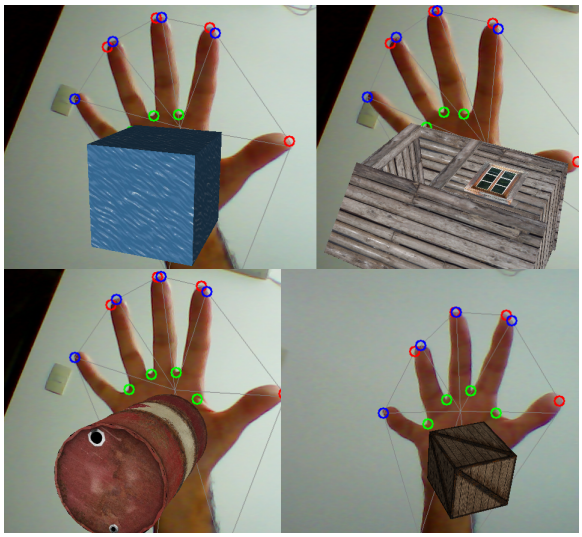

Figure 8. Some 3D models of our system

IV. Results

The presented system was developed in C++ along with Qt library [12] (for the graphical user interface and event management), OpenGL (for augmentation and 3D models), the standard C++ library and OpenCV library, which provides a large number of implementations of the algorithms most widely used in image analysis and processing. Experiments were performed on an HP ENVY 2.2GHz Intel Core i7 computer with 8GB DDR3 SDRAM at a rate of 15 fps with an RGB camera with resolution of 640x480 pixels. The source code is available in [13].

Table I
Accuracy of hand and finger gestures

Gesture	Rotation	Accuracy
Outstretched hand	Frontal	100%
”	$\approx 15°$ in y axis	95%
”	$\approx 30°$ in y axis	50%
”	$\approx 45°$ in y axis	10%
Fingertips forward	$\approx 15°$ in x axis	100%
”	$\approx 30°$ in x axis	70%
”	$\approx 45°$ in x axis	20%
Fingertips back	$\approx 15°$ in x axis	90%
”	$\approx 30°$ in x axis	70%
”	$\approx 45°$ in x axis	50%
Two joined fingers	Frontal	100%
”	$>30°$ in y axis	$<40\%$
Fingertips forward	$>20°$ in x axis	$<40\%$
Fingertips back	$>20°$ in x axis	$<40\%$

The accuracy of our approach is shown in Table I. The first results with outstretched hand frontally and fingers upward were highly accurate. Then, with rotation on the vertical axis we obtained good results even with a rotation of 15 degrees. Exceeding 30 degrees, the results are not very accurate. This limitation is due to the inefficiency of shape analysis to detect the finger valleys when fingers cover up themselves.

Similar results are obtained with clockwise rotation and leftward. With rotations on the horizontal axis, e.g. finger forward or backward, the results are more favorable. Rotations until 30 degrees are acceptable. The interaction joining two fingers has drawbacks when exceeding 30 degrees rotation in either axis (experiments were performed joined index finger and middle finger).

Concerning the computational complexity, the proposed approach is not particularly CPU demanding. The system displays the video images to a rate close to 15 fps during detecting hand and drawing the virtual elements.

V. Discussion

Our experiments indicate the hand segmentation is somewhat sensitive to changes in illumination when the background (e.g. desk) is high in white. This makes the hand gets white saturated. However when the background is black, the detection accuracy is optimal.

Since we are using fingertips and finger valleys as correspondence points for camera pose estimation, we have limitations due to self occlusions and it is significant even for a few degrees of movement as shown in Figure 9. As shown in Table I, when the rotation exceeds 30 degrees, the accuracy is low.

When our system loses tracking of the fingertips and finger valleys due to occlusions, tries to detect them again. The use

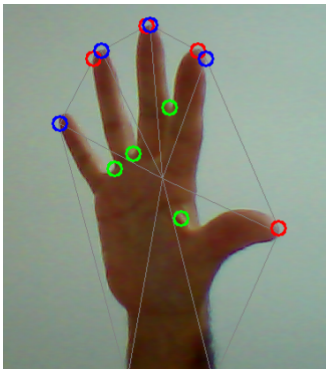

Figure 9. Occlusion due to rotation

of more features on the hand and silhouette-based approaches may be appropriate in order to deal with more hand poses and self occlusions. Using statistical models such as Active Shape Models [14] can be a good option to improve the tracking of the hand.

This method is very sensitive to small changes in the position of the fingers that leads to a different projection than expected. This is because a requirement to achieve the correct projection is that the hand feature points are coplanar and non-collinear. In order not to depend on the position of the fingers, selecting only points in the palm can give more favorable results for these cases.

VI. Conclusion

Many prototypes implemented with sophisticated computer vision algorithms to robustly recognize hand gestures have demonstrated that gesture recognition is rather complex. The method proposed in this paper has low computational cost and with a simple shape analysis eliminates the finger valleys that are false positives, so we can avoid the bad effect of them. The pose estimation accuracy rate reaches high values when the hand is frontally.

For future work, we want to improve the detection of finger valleys and fingertips when the rotation of the hand exceeds the threshold for acceptable detection. We pretend to accomplish it by replacing the detection method through shape analysis and instead choose descriptor methods in order to achieve a greater invariance to changes in position and rotation. Methods such as HOG (Histogram of Oriented Gradients) or SIFT (Scale-Invariant Feature Transform) can be tested.

The possibility of interaction through joining two fingers allows a realistic manipulation of the virtual contents. This opens up a wider spectrum and introduces the concept of Natural Interaction for creating Natural User Interfaces (NUI). We envision a world where it is difficult to distinguish between what is real and what is not. With recent advances made by research groups around the world this idea will be part of the very near future.

References

[1] S. Garrido-Jurado, R. Munoz-Salinas, F. Madrid-Cuevas, and M. Marin-Jimenez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320314000235

[2] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A brief introduction to opencv," in *Proceedings of the 35th International Convention (MIPRO)*, May 2012, pp. 1725–1730.

[3] C. Osimani and E. Kohmann, "Augmented Reality in Universidad Blas Pascal," https://github.com/cosimani/ra-ubp2015, 2015.

[4] M. Sakkari, M. Zaied, and C. B. Amar, "Hands tracking for augmented reality applications," in *International Conference on Information Technology and e-Services (ICITeS)*, Mar. 2012, pp. 1–6.

[5] M. A. Morshidi and T. Tjahjadi, "Feature points selection for markerless hand pose estimation," in *International Conference on Smart Sensors and Application (ICSSA)*, May 2015, pp. 133–138.

[6] T. Lee and T. Hollerer, "Handy ar: Markerless inspection of augmented reality objects using fingertip tracking," in *11th IEEE International Symposium on Wearable Computers (ISWC)*, Oct. 2007, pp. 83–90.

[7] M. Anandhalli and V. P. Baligar, "Improvised approach using background subtraction for vehicle detection," in *IEEE International Advance Computing Conference (IACC)*, Jun. 2015, pp. 303–308.

[8] R. A. Elsayed, M. S. Sayed, and M. I. Abdalla, "Skin-based adaptive background subtraction for hand gesture segmentation," in *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dec. 2015, pp. 33–36.

[9] D. M. Viswanathan and S. M. Idicula, "Recognition of hand gestures of english alphabets using hog method," in *International Conference on Data Science & Engineering (ICDSE)*, Aug. 2014, pp. 219–223.

[10] C. H. Teng and J. Y. Chen, "An augmented reality environment for learning opengl programming," in *9th International Conference on Ubiquitous Intelligence & Computing and Autonomic & Trusted Computing (UIC/ATC)*, 2012, pp. 996–1001.

[11] Z. Wu, H. Wang, and H. Zhang, "Virtual scene modeling technology based on opengl and 3dsmax," in *Fourth International Symposium on Computational Intelligence and Design (ISCID)*, Oct. 2011, pp. 170–173.

[12] J. P. Gois and H. C. Batagelo, "Interactive graphics applications with opengl shading language and qt," in *25th Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, Aug. 2012, pp. 1–20.

[13] C. Osimani and E. Kohmann, "Markerless interaction with Augmented Reality in Universidad Blas Pascal," https://github.com/cosimani/handpose-ra, 2016.

[14] M. Esfandiarkhani, M. Delavari, A. H. Foruzan, and Y. W. Chen, "Improving active shape models performance in low-contrast images using a knn-based search algorithm," in *22nd Iranian Conference on Biomedical Engineering (ICBME)*, Nov. 2015, pp. 132–137.