FPGA-Based Real Time Hand Gesture and AR Marker Recognition and Tracking for Multi Augmented Reality Applications

A.O. Alghabri (1), F. H. Osman (1), and N. Y. Ahmed (2)

(1) Electrical Communication and Electronics Systems Dept., Faculty of Engineering, October University for Modern Science & Arts, Giza, Egypt
(2) Radiation Engineering Dept., National Center for Radiation Research and Technology, EAEA, Cairo, Egypt

Received: 10/5/2016 Accepted: 20/10/2016

ABSTRACT

Most Human Computer Interaction (HCI) systems, generally, and Augmented Reality (AR) systems, specifically, are designed based on general purpose processors. Consequently, the power consumption is considerably high as systems work at Gigahertz rates. In this paper, the recognition and tracking processes of hand gestures, and marker based interactive multi applications AR system, are implemented on a low power FPGA to reduce the overall power consumption, by working at lower operating frequencies. Recognition is performed based on shape features, whereas the depth feature, of gestures and markers, was estimated using an ordinary 2D webcam to reduce the power consumption and cost. The most suitable five hand gestures for 3 to 10 year-olds were determined and the FPGA implemented system was, practically, applied on a 100 children. Implementation results revealed that the system can work at up to 102.8 MHz, whereas only 25 MHz are sufficient to achieve real time performance at 30 fps. This, significantly, reduces the power consumption of the implemented system that was compared to other systems. The recognition rate achieved 93.2 %, on average.

Keywords: Augmented Reality / hand gestures / AR markers / FPGA / Human Computer Interaction (HCI).

I. INTRODUCTION

Augmented Reality (AR) is a direct or indirect combination between real and virtual worlds, by using computer generated graphics, sounds, or videos. AR is used in various applications such as medicine, entertainment, and marketing as a kind of Human Computer Interaction (HCI). AR is also used in educational fields, as it can improve the pedagogical methodology by enhancing the students’ concentration (1,2,3,4,5). However, such systems do not offer a direct virtual interaction between the students’ body and the virtual object, which is necessary to keep them attracted and enriches their imagination. Most interactive AR systems use markers that are represented by either barcodes (6), or certain objects, attached to data gloves (7). Interaction can, also, be achieved by means of bare hand gestures (8,9), or by hand gestures together with markers (10,11). The main challenge in such interactive AR systems is detecting, recognizing and tracking markers and/or gestures, in real-time with high accuracy. Object detection is developed, in some systems, using hardware devices such as gloves, supported with sensors, to digitize the detected hand-gesture/marker (7,12,13).

Corresponding author E-mail: nyounis@msa.eun.eg
Though such systems are robust and provide high recognition rates, their power consumption is high. Also, gloves are not user friendly as they limit the user’s movement, in addition to the inflexibility of the gloves size that cannot fit all human hands’ sizes, specifically, children’s. Other systems use a depth camera, such as Kinect, which provides the HCI system with the hand skeleton and depth such that the remaining detecting steps become easier (14,15). However, it is expensive and power consuming, and is only compatible with Windows operating system. In addition, there is another part of power, consumed by the software program, which is developed to execute the remaining recognition, and tracking operations. Such power consumption is considerably high as the program is executed at gigahertz rates to achieve a real-time performance. A mobile AR system was proposed for teaching and learning, using a low power, low processing device, to control simple hand gestures that in turn controls slides presentation (10). Another effective method to reduce the power consumption is implementing the high computationally complicated functions on an FPGA platform, where a wearable backpacked computer and tracking gloves are used (16). However, the system does not provide a high recognition rate, since it is built in Handel-C language that was later converted to a Hardware Descriptive Language (HDL). Also, the system is not user friendly, as the user should wear gloves and a heavy computer while using it. Another FPGA-based hand gesture HCI is proposed in a previous study (17), where an Artificial Neural Network (ANN) was implemented on the FPGA to recognize hand gestures. Again, a data glove is used to detect the proposed gestures.

To reduce the power consumption, an FPGA-based interactive AR system is proposed, where the high computationally complicated hand-gesture/marker detection, recognition, and tracking processes are designed and implemented on the FPGA. The power consumption is then reduced by operating the implemented system at lower frequencies. This is achieved by gaining the benefit of the parallelism feature from the FPGA to execute some complicated functions in parallel, in a way that the required real-time performance is not affected. As a result, the proposed low power consuming design can be used portably anywhere that suits children, especially in Kindergarten and primary levels. The implemented applications have been applied on normal and autistic children to find how far it improves their concentration.

The remainder of the paper is organized as follows: Section II demonstrates the proposed hand-gesture/marker AR system, while its software and hardware architectures are explained in Sections III, and IV, respectively. In Section V, the Hardware-Software interface is described, whereas the practical implementation result is discussed in Section VI. The system efficiency w.r.t. some other gestures/markers based AR and HCI systems is discussed in Section VII, and the conclusion is presented in Section VIII.

II- Proposed System

The proposed system supports four different applications. The first two use hand gestures to control virtual objects, and the other two use markers. The first application is "Animal homeland", in which the user grabs a virtual animal using his/her hand and places it on its home land, whereas the second application uses hand gestures to explain the basic steps of planting. The third application uses different mono-color markers to assemble machinery objects, such as an airplane. The fourth application uses multi-colored markers to demonstrate the atom structure.

To minimize the computational complexity, caused by recognizing and classifying different objects of various applications, the system was designed such that the user, first, selects an application from the GUI, depicted in Fig. 1, which was developed using Unity (18). Then, the selected application number is sent to the FPGA, where classification and tracking are performed for only the application gestures/markers.
Fig. (1): GUI of the proposed multi-application interactive AR system

Fig. (2) illustrates the system setup, where the input of the system is successive frames, captured by a CMOS webcam, which is directly connected to the Personal Computer (PC). The scene includes a white background, on top of which, a hand gesture/AR marker moves. ‘A’ and ‘B’ represent the areas of the detected object at zero and maximum heights, respectively, which are used to get the object depth, as explained later.

The output of the FPGA, which represents the recognized gesture/marker and its 3D position, is sent to the laptop. Then, the camera monitor, after mixing it with a certain virtual object, depending on the FPGA received data.

III- Software-Based Architecture

Since the camera is, directly, connected to the PC to display the user’s hand on its screen, while interacting with virtual objects, then the frames are stored in its RAM. Hence, it is more efficient to carry out the first step of detection, which is color segmentation, by the PC, to minimize the size of the utilized FPGA RAMs. According to the selected application, the type of color segmentation is determined. For hand-gesture-based applications, a skin color filter, which range is selected using MATLAB, is applied. For AR Marker-based applications, a color filter is applied, based on the colors of the markers. Then, a binary conversion of the segmented frame is executed. Both color segmentation and binary conversion are developed by Unity.

Fig. (3) shows the skin color segmentation of the five hand gestures, used in the system, as it is found, by practice, that they are the most feasible gestures for children. A green ribbon is worn on the user’s wrist, if he/she is not wearing long sleeves to isolate the detected hand from any part of the arm that may exist in the received frame. The color segmentation of the applied markers is displayed in Fig. (4). The N×M binary frame is, then, sent to the FPGA for recognizing and tracking processes, where N×M represents the frame size. For marker-based applications, the markers colors of the selected application are also sent to the FPGA to ease classification, as children may use other markers of different applications, having the same features of those used in the selected application.
Since in the proposed system the application is first selected, a limited number of predetermined hand gestures or markers are detected. Hence, the shape-based features are used for classification to reduce the computational complexity; the extracted features are the object perimeter, area, solidity, Center of Gravity (CoG), and the object depth.

### A. Object Perimeter Estimation

Fig. (5) illustrates the FPGA architecture of the implemented system, where the received binary frame is stored in an N×M RAM, implemented in the FPGA. The perimeters of all segments in the frame are, first, extracted by dilating each shape in the original binary frame with the structuring element, as shown in Block A of Figure (3). The dilated frame is then XORed with the original binary frame, to give a new N×M array that includes only perimeters (19). Then, the structuring element, shown in block B of Figure (3), is used to go through all shapes in the frame, and the number of binary-one-pixels for each connected segment is accumulated, to get the perimeter length. The largest estimated perimeter represents the marker/gesture perimeter. Other segments are considered noise, and therefore deleted, except the ones enclosed inside the object perimeter, as they are used to detect the inner hole of the gesture, if any, as explained later. Also, the minimum and maximum X and Y coordinates of the object perimeter - (Xmin, Xmax) (Ymin, Ymax)- are saved to extract other features.
B. Parallel Extraction of Features

To gain the benefit of the parallelism feature from the FPGA, the calculation of the area, bounding box, CoG, as well as the inner hole detection of the gestures, shown in Figure (3 d and e), are processed in parallel. This reduces the power consumption and processing time, considerably. To increase the recognition rate of the gestures of Figure (3 d and e), their inner hole is detected after extracting the object perimeter. This is done by comparing the perimeter of shapes that are bordered by \((X_{\text{max}}, X_{\text{min}})\) and \((Y_{\text{max}}, Y_{\text{min}})\). The segment represents an inner hole of the gesture if the ratio between the perimeters of the candidate hole and the object is greater than 0.01%. Otherwise, it is considered noise. Figure( 6) shows the inner hole, enclosed inside the hand gesture of Figure( 3 d).

![Fig. (6): The edges of the three fingers-gesture and its inner hole](image)

The Object area is calculated by (1), where max \(x_y\) and min \(x_y\) represent the maximum and minimum X coordinates, respectively, which exist on the object edge at a specific \(y\). In parallel, the coordinates of the CoG, \((X_C, Y_C)\), are calculated by (2) \(^{(20)}\). Also, the width, \(W\), and length, \(L\), of the bounding box are calculated using (3) and (4), respectively.

Another feature, to be extracted, is the solidity. It is used since it is not affected by the distance between the object and the camera, as it represents the ratio between the object area and the convex area. In the implemented system, the bounding box is used instead of the convex area to reduce the computational complexity. The solidity, \(S\), is then estimated by (5).

\[
A_d = \sum_{y \in Y_{\text{min}}}^{Y_{\text{max}}}(\text{max } x_y - \text{min } x_y) \\
X_C = \frac{X_{\text{max}} - X_{\text{min}}}{2}, Y_C = \frac{Y_{\text{max}} - Y_{\text{min}}}{2} \\
W = X_{\text{max}} - X_{\text{min}} \\
L = Y_{\text{max}} - Y_{\text{min}} \\
S = \frac{A_d}{(W \times L)}
\]

C. Object Recognition and 3D Pose Localizations

The perimeter, area, and solidity of the five gestures were pre-calculated for 140 different hands to define the upper and lower limits of their thresholds. In Figure( 7), the five gestures from "Open" to "With hole" represent the five gestures of Figure 3, from Figure 3(a) to Figure 3(e). Figure (7) shows that the (open) and (3 Fingers) gestures can be recognized based on any of the three features, whereas the (Vertical) gesture can be recognized based on the area and the perimeter features. Though the (With hole) and (Horizontal) gestures are overlapped in all features, the inner hole detection is used to distinguish them. Similarly, for marker recognition, the colors and the normalized values of the area and solidity of all utilized markers were calculated, and listed in Table(1). For 3D pose localization, more than a single 2D cameras, or depth sensors are usually used \(^{(14,15)}\). This is mandatory if the utilized gestures/markers rotate or twist during interaction. In the proposed system, neither rotation nor twist is required. Thus, the object depth can be calculated using only one 2D camera, based on the
change of the object area at different heights, with respect to the camera, as shown in Figure (2). For any different user, both A and B areas shown in Figure (2), are calibrated once, for all gestures, such that after object recognition they are used to estimate its depth, Z, by applying (6). A is the biggest possible hand area in pixels, whereas B is the smallest possible hand area in pixels. H, represents the height at which the camera is fixed, with respect to the background, and ‘∆’ is the difference between the areas A and B that is independent of A_d. However, ‘∆’ varies if the distance between the camera and the background is changed.

\[ Z = (A_d - A) \times H / \Delta \]  \hspace{1cm} (6)

Fig. (7): Normalized values of (a) Area, (b) Solidity, and (c) Perimeter of 140 different hands for the applied five hand gestures, in the system

**Table (1): Normalized values of the markers features**

<table>
<thead>
<tr>
<th>Marker</th>
<th>Color</th>
<th>Area</th>
<th>Solidity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor</td>
<td>Green</td>
<td>0.30421875</td>
<td>0.65847</td>
</tr>
<tr>
<td>Tale</td>
<td>Green</td>
<td>0.22514329</td>
<td>0.84043</td>
</tr>
<tr>
<td>Wheel</td>
<td>Green</td>
<td>0.31708333</td>
<td>0.5021</td>
</tr>
<tr>
<td>Wing</td>
<td>Green</td>
<td>0.33115885</td>
<td>0.91021</td>
</tr>
<tr>
<td>Proton</td>
<td>Blue</td>
<td>0.32338541</td>
<td>0.66774</td>
</tr>
<tr>
<td>Electron</td>
<td>Red</td>
<td>0.36265625</td>
<td>0.74883</td>
</tr>
<tr>
<td>Neutron</td>
<td>Yellow</td>
<td>0.30085937</td>
<td>0.62123</td>
</tr>
</tbody>
</table>
V- Hardware - Software Interface

To interface the FPGA with the PC, the Ethernet and User Datagram Protocol (UDP) are used. The UDP is selected because the data are transferred directly without being divided into chunks. Also, the UDP does not depend on a certain Operating System (OS). The NXM binary frame is sent to the FPGA via Ethernet, followed by an $m$-bit data that represent the application number, and the marker color. For the proposed system, 4 bits are adequate to represent $m$. On the other side, a data vector is sent from the FPGA to the PC, via Ethernet, after executing the recognition and tracking processes. Such vector carries the required information of the recognized object and its 3D position. The first $Q$ bits of the data vector represent the recognized gesture/marker, where $Q$ is determined according to the maximum number of different objects, used in one application.

In the proposed system, the maximum number of objects in one application is five; hence, $Q$ equals three bits. The remaining bits of the data vector are 24 bits that represent the 3D position of the detected object, $(X_c, Y_c, Z)$, where each coordinate is represented with eight bits. The packets, sent via Ethernet to the FPGA, are monitored by the Wireshark analyzer that also monitors the packets received from the FPGA, and provides error check methods, which in turn, diagnose and correct errors that can be resulted from the VHDL code. A C# program was developed by Unity to capture the received data and check that they are passed by all network layers and the OS accepts it. Then the Unity is used to combine the user's hand/ Marker with a 3D virtual model, selected and located on the screen according to the received data vector.

Fig. (8) illustrates the functions, designed and developed by Unity, where the data base of the application, selected by the GUI, is passed to the UDP receiver that also receives the data vector from the FPGA at specified Internet Protocol (IP) and port addresses. The received data packet is then stored temporarily in the PC’s RAM, using the Buffer unit, where each new packet deletes the old one. A converter is used to convert the received Hex data to the American Standard Code for Information Interchange (ASCII), which represents the original data type. Afterwards, the data are analyzed according to the selected application, and the received 3D coordinates, where the center of the detected object is combined with that of the virtual 3D object in the Integration Unit. The continuous response of the virtual object is controlled by the Controller Unit, based on the data vector.

![Fig. (8): Software algorithm, executed by Unity game developing Platform](image)

VI Implementation Result

Several visits to schools were paid to determine the practical accuracy of the implemented system, and to see how well kids interact with it. The practical testing setup consisted of a webcam, with a 640×480 resolution, which represents the NXM size of the captured frames. The camera is fixed at height of 70 cm above a 50 cm × 25 cm white background, where the camera was directly pointed to the background. The software–based architecture runs on a 2.2 GHz processor. Table (2) lists the recognition rate of each hand gesture after testing it on 100 individuals. From Table (2) it can be deduced that the average recognition rate is 93.2%. On the other side, the accuracy of the utilized markers approaches 100 % because their shapes, features, and colors are constant, unlike gestures features.
Table (2): Recognition rates of the utilized hand gestures

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Open</th>
<th>Hori</th>
<th>Ver</th>
<th>3Fin</th>
<th>With Hole</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognition (%)</td>
<td>96</td>
<td>93</td>
<td>92</td>
<td>90</td>
<td>95</td>
</tr>
</tbody>
</table>

Figs (9 and 10) show kids, interacting with 3D objects by hand gestures and markers, respectively. Children were very excited, and they concentrated in the applications. Autistic children, however, were more interested in marker based applications, and they were hardly obeying their teachers in performing different hand gestures. However, after an adequate effort from teachers they, so fare, could interact by gestures.

Table (3) lists the FPGA utilized resources of the proposed system, estimated by Xilinx Integrated Software Environment (ISE) tool, after implementing it on the low power Spartan3, S1600e-4fg320 device. The FPGA chip was mounted on a MicroBlaze development Kit - Spartan3E-1600E that featured an Ethernet interface to the PC. It can be noticed that the implemented system uses less than 4% of the FPGA resources, which allows further modifications to improve the recognition rate and increase the number of gestures and markers. Also, the implementation reports showed that the maximum operating frequency at which the FPGA-implemented system can work is 102.817 MHz. However, the system, in real time, works at, only, 25 MHz for 30 fps rate that represents the maximum frame rate of the used webcam. On the contrary, using a higher rate camera is not effective, since the hand speed varies from medium to slow rates. Consequently, it is very fair using low frame rates, starting from 15 fps.

The power consumption of the implemented system was also estimated, using XPower analyzer, provided by Xilinx. It is found that the FPGA implemented system consumes a power of 5.65 mw at 25 MHz.

It should be mentioned that, using higher frame rates is not required, since hand speed varies from medium to slow rates. Hence, it is more efficient to use lower frame rates, such as 15 fps, to reduce the operating frequency required for a real-time performance. This, in turn, optimizes power consumption much better.

Fig. (9): Kids practice (a) the first and (b) the second applications.

Fig. (10): Kids practice (a) the third and (b) the fourth applications.
To determine whether the learning outcomes of the applications have been successfully received by the students and how far such interactive applications enhance their concentration, some matching quizzes were given to 25 normal children ranging from 4 to 8-year-olds. The quizzes results differed according to the child’s age and the application, however, the results were 75% on average, after the first time of practicing the applications.

VII- System Efficiency

Table (4) summarizes a comparison between the proposed system and some other hand gesture and marker based HCI systems. Where G and M are the number of the used gestures and markers, respectively, R is the recognition rate, FR is the frame rate, FOP is the applied operating frequency for real-time recognition and tracking, PHW is the amount of power, consumed by the hardware equipment, UR is the FPGA Utilized Resources, and U stands for undefined. From Table (4) it is noticed that for a real-time performance, the operating frequency and power consumption of the proposed system are lower than those of the other systems. Another important point is that, the markers, used in the proposed system, are handmade, unlike the complicated barcode markers, used in used in earlier studies (7,12,16). Using such simple and cheap markers makes children feel included as they participate in creating the 3D objects, and it makes the system more reliable, as its reusable resources are affordable.

Table (4): A comparison between the proposed system and other interactive AR an HCU systems

<table>
<thead>
<tr>
<th>System</th>
<th>G</th>
<th>M</th>
<th>R (%)</th>
<th>FR</th>
<th>FOP (MHz)</th>
<th>P_HW (mw)</th>
<th>Technology Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]</td>
<td>2</td>
<td>None</td>
<td>90</td>
<td>U</td>
<td>1600</td>
<td>&gt;12000 (for Kinect [21])</td>
<td>Intel Core i5 processor &amp; Kinect camera.</td>
</tr>
<tr>
<td>[10]</td>
<td>4</td>
<td>3 colored ribbons</td>
<td>93.3</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>Low Processor (U).</td>
</tr>
<tr>
<td>[16]</td>
<td>None</td>
<td>2 color balls</td>
<td>U</td>
<td>25</td>
<td>U</td>
<td>6000</td>
<td>Celoxica RC200 reconfigurable computer, Virtex II, Head worn PAL resolution camera, and Tracked glove.</td>
</tr>
<tr>
<td>Proposed system</td>
<td>5</td>
<td>7</td>
<td>93.2</td>
<td>30</td>
<td>25 at 30 fps</td>
<td>(5.65: for FPGA-based design)</td>
<td>General purpose processor &amp; FPGA (UR &lt; 4 %).</td>
</tr>
</tbody>
</table>

VIII Conclusion
A low power interactive AR learning system was proposed. The high computationally complicated recognition and tracking functions were implemented on an FPGA to minimize the operating frequency without violating real-time performance. This helps using the proposed applications portably. Implementation and testing results show that the recognition rate of the implemented system is 93.2% on average. Comparing the proposed implemented system to some other systems, it is found that the proposed system is more efficient in terms of power consumption, and reliability. For future work, a general form could be deduced to get the optimal FPGA operating frequency, at which the system can work in real-time, as a function of the frame rate. This will enable working at rates even lower than 25 Mhz. Also, more hand gestures and markers can be recognized by adding additional features as only less than 4 % of the FPGA resources are utilized.

REFERENCES


(18) Unity, <https://unity3d.com/> > 2016.11.16

