Depth Based Hand Shape Recognition using Contour Analisys

João Delgado[1], Tiago Cardoso[1,2], Pedro Mota[1]
[1] Dept. of Electrical Engineering (DEE), Universidade Nova de Lisboa (UNL)
[2] Centre for Technologies and Systems (CTS) – UNINOVA
Campus FCT/UNL, 2829-516, Caparica, Portugal
j.delgado@campus.fct.unl.pt, tomfc@uninova.pt, p110243@campus.fct.unl.pt

Abstract — **Human-Computer Interaction (HCI) faces a great challenges in what concerns the creation of a fully natural interface that is able to recognize a wide range of hand gestures in real-time. Lately efforts have been made by the research community catalyzed by the wide appearance of depth sensors such as the one from the Microsoft Kinect sensor. Nevertheless the available solutions struggle to robustly recognize hand gestures since, compared with the rest of the body, the hand is a small region with many joints, difficult to recognize and ultimately track. This paper proposes a hand shape recognition method that takes advantage of the depth sensor and the joint tracking ability of Microsoft Kinect sensor. The proposed method relies on the analysis of the hand contour or outline and compares it with previously stored templates achieving a good recognition ratio and proving it eligible for real-time scenarios.**

Keywords—Hand Shape, Hand Gesture, Kinect, Contour, Template Matching.

I. INTRODUCTION

The world of digital games has always been at the edge of technological advances in many of computer science fields of research, all to provide a better experience to the user. This was mainly the case in graphical solutions and artificial intelligence algorithms that were developed to make the game feel more real to the consumer. Lately the gaming industry has expanded itself to yet another research area, the Natural User Interfaces, that allow for a digital solution to be controlled as it was a real word object. This, besides providing the games with a more realistic input method has another (and perhaps the most important) advantage, the machine can now evaluate the way the input is being made, allowing for very interesting and useful applications in the field of serious games.

The field of Human-Computer Interaction (HCI) is a multidisciplinary one that is intimately connected with computer and social sciences. The goal of HCI is to create new interaction paradigms between persons and machines, developing descriptive models for these interfaces and developing methods to evaluate them.

Since HCI studies both the behavior of humans and machines [1], it gathers knowledge from both sides:

- On the machine side, skills like computer graphics, operating systems and programing languages are covered or included;
- On the human side the relevant skills are communication theory, linguistics, cognitive psychology and social sciences, among others.

Nowadays one of the new emerging paradigms is the so called Natural User Interfaces (NUI). The overall idea is that rather than the user having to rely on additional hardware as means of input or output, she or he can interact with the computer systems in the same way as the interaction with real-life objects or artefacts. In terms of input means, the target of this community have tackled the fields of input through touch, voice, body gestures or face recognition. Another input perspective is the mechanisms or languages of commands that machines understand. In this field, ambiguities such as the ones existing in the natural languages are not covered by the computer input means. For example, if the command “run” is supported by some system, usually instructions like “go” or “start” are not understood by such machine.

Putting the focus on gesture recognition, the most effective solutions make use of electro-mechanical devices such as data gloves [2] since they provide the most complete set of information of the hand. However, besides the fact that they are very expensive for recreational use and that they require complex calibration procedures, they do not provide a fully natural interface, since the user is required to wear the glove. A commercially available solution that follows this approach is MYO [3], an armband that uses the electrical impulses of the arm muscles to wirelessly control digital components.

Vision based solutions have been presented as a viable alternative as they don’t require additional hardware, besides some sort of camera. Nevertheless, these solutions usually require high processing power for image recognition algorithms to provide reliable results.

The appearance of these solutions has lately been catalyzed by the availability of depth sensors that obviate the impact of the background and lighting conditions in the system performance.

Amongst these sensors the most prominent are the Microsoft Kinect sensor, Asus Xtion and Leap from Leap Motion.

The presented solution will make use of the Kinect sensor since, comparing to Asus Xtion it has several advantages such as an official SDK and the joint tracking algorithm materialized in the so called Skeleton frame. As for Leap although is provides a better resolution of the hand and a complete skeleton of the hand, it only focuses on the user's hands which is insufficient for complete sign language recognition since most of the gestures require reference points on the user's face and body to be successfully recognized. Overall Microsoft Kinect sensor is both the one that provide the better set of tools and the one that will most benefit from a vision based hand gesture recognition platform.

Despite this the low resolution provided by the Kinect's depth frame (typically 640 x 480), is sufficient for large object tracking, such as the human body, but renders the segmentation of the hand inaccurate, since it is a much smaller region with some of the most complex articulations in the human body, which can affect the recognition process.

II. Related Work

Generally a hand gesture may be separated into its dynamic and static components meaning the hand movement and the hand shape.

For the dynamic component literature is divided in two main categories:

- Machine Learning approaches - based on probability ratio rather than exact values defining the gesture as the output of a stochastic process. The main approach is the use of Hidden Markov Models (HMM) to classify hand gestures [4] [5].

- Algorithmic Approaches - define the gesture as a set of manually encoded conditions and restraints. For example Galveia [6] defines the dynamic component of hand gestures as a set of 3rd degree polynomial equations. This author proposes the creation of a gestures library, represented by the equations and, afterwards, the corresponding recognition, reduced to the complexity of the equation's comparison handling.

For the recognition of the static component of the hand gesture the approaches in literature can be differentiated according to the extracted features as follows:

- 3D reconstruction – these approaches attempt to compute a 3D model providing a complete definition of the hand (this eliminates the self-occlusion problem inherent to 2D projections). However a robust 3D reconstruction is difficult to achieve, according to [7] [8]. This fact is mainly based on the need for better resolution on the input mechanisms, on one hand side, and the high computational costs needed for rendering, on the other hand side. This last fact is incompatible with real-life requirements.

- High-Level features – these approaches aim to infer the position and orientation of key anchor points of the hand [9] [10] [11]. They usually rely on a high contrast 2D hand image which makes it very sensitive to the segmentation process. These high-level features result on a set of rules and conditions, for the vectors and joints of the hand that define the gestures.

- Low-Level features – these approaches extract features that are fairly robust to noise and can be extracted quickly. For example Zou [12] defines the hand shape as a cluster based signature through a novel distance metric called Finger-Earth Mover's Distance that recognizes the hand shape.

From the point of view of the processes used to recognize the static gestures' components, the research community proposes:

- Algorithmic approaches - based on a set of conditions and restraints between key anchor points and their orientation.

- Template Matching approaches - define the gesture as a signature immune to the three geometric transformations (T, S, R - Translation, Scaling, Rotation) and then match it against a shape library towards finding the best match.

These approaches, both concerning the static and dynamic components of gesture recognition, correspond to the state-of-the-art on this research area. More detailed presentations can be found in [13] [14].

III. Hand Contour Recognition

The proposed approach for hand recognition is based on gathering the contour or outline hand and comparing it to existing or pre-defined gesture templates. Figure 1 shows the main workflow of actions of the proposed solution.

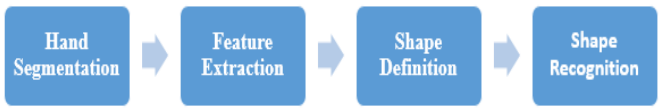

Figure 1 - Solution Workflow

The four main steps illustrated above will be detailed in the following sub-sections.

A. *Hand Segmenation*

The first step is the isolation of the hand image from the rest of the body and other "trash" that might be gathered by the input device. For that purpose, the depth camera is used. The alternative method would be the usage of RGB. The main problem would be that one could not distinguish the hand from the rest of the body elements. In other words, using the depth camera, the statistic success of the hand isolation process improves. This option also eliminates interferences that might be caused by color of the background as well as changes in luminosity.

Taking advantage of the joint tracking feature of the Kinect sensor one can achieve a more robust segmentation process. As a result, the proposed solution is based on this image segmentation in what concerns the user's hand isolated part. The process works as follows:

1) Get the position of both hand central joints, provided by Kinect's Skeleton frame.

2) Extract a segmentation output image - the smallest box that surely contains the region to be analyzed – the hand. In this case it was defined that the hand would be contained in a 120 x 120 pixels box, positioning the hand joint at the center, the box dimensions were dimensioned by threshold adjustment and it will be different according to the resolution of the depth sensor and the distance at which the solution is being used.

3) Hand-Pixels assessment - Pixel-by-pixel analysis carried out to determine if a pixel is a hand pixel. This process analyses the intensity of each pixel. The advantage of this approach is that the cost of running through the pixels has already been minimized as we only need to focus on the 120 x 120 region around the hand joint. In this case a hand pixel was define as one that obeys the following rules:

   - The pixel belongs to the player, this takes advantage of the player labeled pixels, provided by the Kinect depth frame, to separate the player from the background 2.a).

   - The pixel is within a certain depth threshold of the hand joint. This will eliminate any player pixels that are in the hand region but does not belong to the hand, such as chest pixels 2.b).

   - The pixel is in the semi-plane defined by the wrist joint of the corresponding hand and the vector going from the wrist to the hand joint. This will "cut the hand by the wrist" and eliminates arm pixels producing better results in some feature extraction algorithms 2.c).

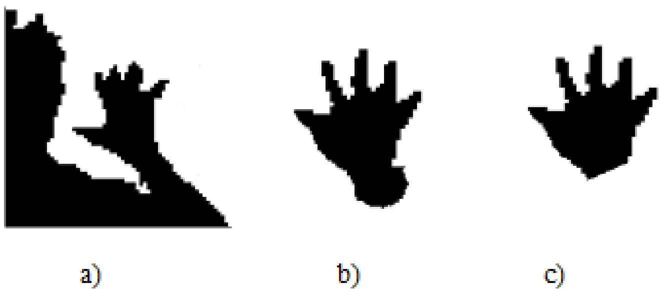

Figure 2 - Hand Segmentation Process: a) Segmentation by Player Index; b) Depth threshold segmentation; c) Final Hand Output

The output produced will be a 2D projection of the hand in a grayscale image to reduce memory cost as much as possible.

B. *Feature Extraction*

For the proposed solution the hand shape contour is extracted in this step.

This can be done practically by implementing an algorithm that produces the set of contour points by analyzing the intensity value of neighbor pixels, such as a Moore-Neighbor Tracing Algorithm [15], which can be described as follows:

Below is presented a list of variables meaningful to this algorithm as well as their meaning:

$$\left\{\begin{array}{l} N(\alpha) \rightarrow \text{Moore Neighborhood of pixel } \alpha; \\ P \rightarrow \text{Current contour pixel;} \\ Q \rightarrow \text{Starting pixel of current Neighborhood checking;} \\ C \rightarrow \text{Set of detected contour points, initialized to be empty;} \end{array}\right.$$

For a box line we consider that the Moore Neighborhood is the Eight-point Neighborhood of a certain pixel, namely pixels P_1 to P_8, as is shown in the figure below:

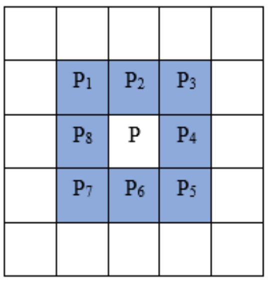

Figure 3 - Eight-point Neighborhood Representation

The algorithm is now described as follows:

1) Start by scanning all pixels from top to bottom, left to right until a hand point is found, let this pixel be the start pixel S.

2) Set the current contour pixel P to be S and the pixel of the current neighborhood checking Q to be the pixel north of S.

3) Insert P into C and compute N(P).

4) Start from Q and go clockwise around N(P) until a hand pixel is found, let it be R.

5) Backtrack i.e. set Q to be P and P to be the new contour pixel R.

6) Repeat from step 3 until S is found again.

C. *Shape Definition*

The contour extracted in the previous process is defined as the set of pixels that separate the object from the background, making it susceptible to geometrical transformations (translations, rotations and scales) since it is coded relatively to the image's origin point.

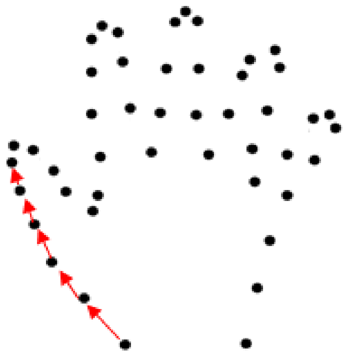

Figure 4 - Contour Definition: In black standard definition as the set of points, in red the vector representation

The proposed solution defines the contour as a start point and the set of complex vectors that define the contour as illustrated above. Each vector is defined by $(a + ib)$ with a being the offset in the x axis between two sequent points and b the correspondent offset in the y axis. With this definition an interesting set of properties may be added to the contour, such as:

- Scaling transformations of the object's contour are translated in the multiplication of each complex vector by a scale factor.

- The contour is unaffected by translations of the object since it is computed in relation to the starting point.

- Rotation operations on the source object will result in a change of the argument of each complex vector.

- The sum of the complex vectors of a contour is zero, this happens since contours are closed sets, meaning that the last complex vector will end at the starting point of the first one.

- A change of the starting point will not change the complex vectors sequence but only the vector by which it starts.

Considering the complex number $(a + ib)$ one can define it in its polar form as $r\ cis(\varphi)$, where $r = \sqrt{a^2 + b^2}$ and $\varphi = \tan^{-1}(\frac{b}{a})$, as illustrated below:

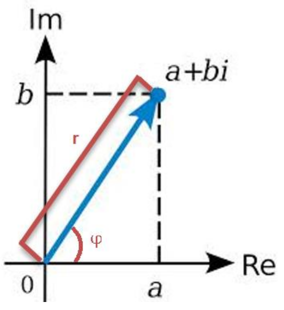

Figure 5 - Complex number representation

Now we define the scalar product of two complex numbers as:

$$(r_1\ cis(\varphi_1))(r_2\ cis(\varphi_2)) = r_1 r_2\ cis(\varphi_1 + \varphi_2)$$

With this the scalar product of two contours with size k, contours A and B, would be the sum of the scalar products of each pair of complex vectors, as follows:

$$SP(A,B) = \sum_{n=0}^{k-1} a_n b_n$$

Furthermore to normalize the scalar product of the contours so they become invariant to the size of the contours, we divide the scalar product by the multiplication of the contours norms:

$$NSP(A,B) = \frac{\sum_{n=0}^{k-1} a_n b_n}{|A||B|}$$

With the norms $|A|$ and $|B|$ being defined as $\sqrt{\sum_{n=0}^{k-1} |a_n|^2}$, in the polar form $|a_n|^2 = r_a$.

The normalized scalar product is defined as a complex number, with a Cartesian and a polar forms, it's norm relates to the degree of similarity between the two contours, reaching a maximum value of 1 if the contours are the same, although they might be rotated in some degree which is reported by its argument, φ.

So the normalize scalar product of contours is invariant to both scale, by being normalized by the contours norm, and rotation as the norm of the product remains unchanged. Furthermore we can retrieve the rotation angle between the contours by accessing its argument. Since the contour is, in its definition immune to translation geometrical transformation one could argue that the contour is already a good template, but there is still one edge to dull, the scalar product is susceptible to changes in the starting point.

To obviate this problem it has been defined an Intercorrelation function as being the set of scalar products of contour A, with all the possible configurations regarding the starting point for contour B, the solution can be defined as a contour since it is already a set of complex points:

$$IF(m) = NSP\left(A, B^{(m)}\right); \qquad m = 0,1,\dots,k-1$$

The Intercorrelation function contains the degree of similarity of contour A with all the different configurations of contour B, but only one is needed so, for recognition purposes we evaluate the inter-correlation by looking at its maximum value $\max(IF(m))$, where it's norm refers to the level of similarity and it's argument, the corresponding rotation angle between the contours, so now one can conclude that the maximum Intercorrelation function between two contours is invariant to the three geometrical transformations and the start point shifting operations.

Although the computational cost for calculating the Intercorrelation function is not very high, it can become a problem when working with template databases of a considerable size. In order to mitigate the issue of the recognition processing cost it has been defined an Autocorrelation function that is nothing more than the similarity level of a contour with itself at various shifts of the starting point, as follows:

$$AF(m) = NSP\left(A, A^{(m)}\right); \qquad m = 0,1,\dots,k-1$$

The Autocorrelation function, such as Intercorrelation, is again immutable to the three geographical transformations and the start point shifting operations, and thus provides a good signature for the contour.

Although Intercorrelation function provides a much more accurate evaluation of the similarity between contours, autocorrelation is useful for a rough selection in order to minimize the number of contours analyzed by the Intercorrelation function.

As Autocorrelation function depends only on the contour itself, it shifts the computational cost inherent to its computation to the step of the creation of the template, as it will be a part of it.

So, for each contour extracted a template is created that contains the redefined contour, the Autocorrelation function and, to further lessen the computational cost, a set of three descriptors for the initial comparison, since comparing three values will be much faster that comparing one contour against another, there are computed as follows:

- Maximum value for the Autocorrelation Norma.
- Medium value for the Autocorrelation Norma.
- Offset between maximum and medium values.

These three are not the most exclusive description of a contour but provide a good initial filtering providing for a better execution time for the solution.

D. *Shape Recognition*

The recognition process follows a template based approach. These approaches usually present better success ratio regarding shape recognition since one can actually check for a level of similarity between templates instead of basing the output in a set of predefined conditions.

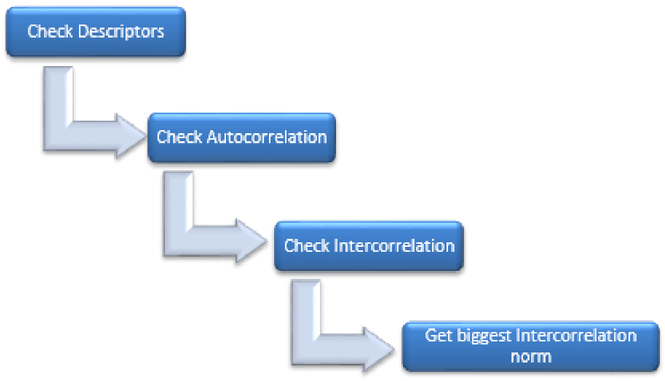

Figure 6 - Shape Matching Process

In the initial condition the templates are scanned for their Autocorrelation descriptions which are tested to be within a certain threshold of each other. This provides the first layer of examination that will exclude the vast majority of templates.

In a second level the Autocorrelation functions are compared against each other by computing their normalized scalar product, since Autocorrelation is represented as a complex vector as well, which is compared with the desired level of similarity. In this case we look for about 95% similarity between Autocorrelation functions.

In the third level only very few contours remain and they are checked for their Intercorrelation and the maximum norm of the resulting contour is compared against the desired level of Intercorrelation similarity, in this solution this value was defined to be 80%, the corresponding argument is also checked against a predefined maximum angle deviation factor, that in this case is $\frac{\pi}{4}$.

In the last layer of recognition, since only one contour can be recognized, their maximum Intercorrelation norm is checked against each other being selected the one with the biggest level of similarity.

IV. VALIDATION

The validation methods were developed for the Kinect SDK version 1.8. The device used was the Xbox Kinect Sensor which has some differences in relation to its Windows counterpart (Kinect SDK was fully tested with Kinect Sensor for Windows which, besides API improvements, also implements a near mode). The processor used for the validating of the proposed solution was an Intel Core I7 @ 2,40 GHz. For testing purposes the Kinect was placed at a 95 cm far from the ground and the hand recognition process was tested at a distance of 200cm.

For the purposes of testing the rate of recognition of the hand shapes two tests were made both performed by four different persons in order to achieve a heterogeneous sample data. The shapes selected for testing purposes were the ones corresponding to the first five letters of the Portuguese Sign Language alphabet, as illustrated below:

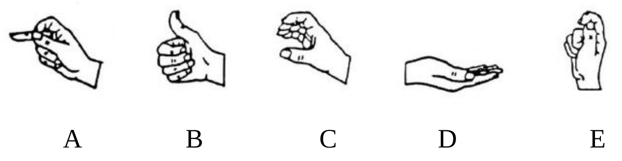

Figure 7 - Hand Shapes for the first five letters of the alphabet in Portuguese Sign Language

The first test was performed with only one shape stored in the shape library at a time to clearly demonstrate the recognition rate without any interference from "false positives" caused by having other shapes. This process was made towards testing the robustness of the recognition. This means that the persons that performed the shape tests introduced some degree of rotation and scaling to conclude if the conditions were too strict. Each shape was evaluated 10 times achieving a maximum of 40 in total recognition. The test results are presented in the table below:

Table 1 - Results of the test performed with only one shape stored in the database

	P1	P2	P3	P4	Total
A	9	9	8	9	35/40
B	9	10	10	9	38/40
C	8	7	9	8	33/40
D	10	10	8	9	37/40
E	8	9	9	9	35/40

The validation has proven that the prototype has a good recognition rate minimizing the non-matched hand shapes. This results arose the question of whether the solution would have provided false positives in the non-recognized shapes, so the test was repeated under the same base-conditions but with all five shapes stored in the template database. The first time the gesture was performed 10 times again. It was evaluated not only the successful recognitions but also the number of times the gesture was recognized as wrong gesture. The test results are presented in the Figure 8.

The results have proven that the solution performs relatively well in multi-shape database scenarios Please note that the shape A is the one providing more false positives and the vast majority of the times it is mistaken recognized as shape B. This happens due to the rotation threshold of the contours Intercorrelation argument not being strict enough.

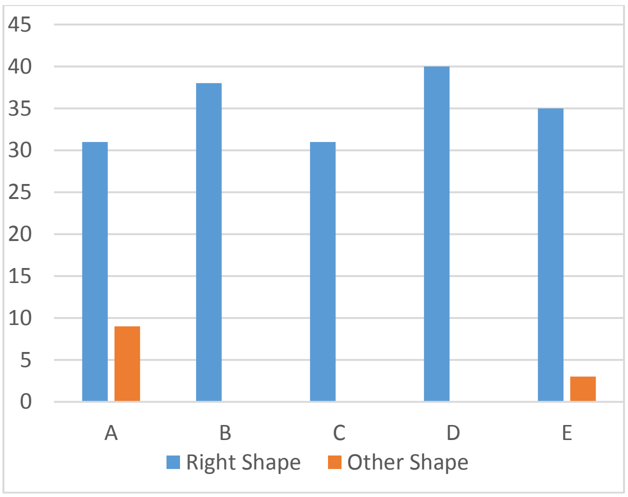

Figure 8 - Results of the test performed with all five shapes in the database

All testing was performed under a real-time scenario proving that the computational cost of the proposed solution renders it eligible for this type of applications.

V. Conclusion

Despite the efforts of the scientific community, the hand gesture recognition issue is still unresolved and attempts are still being made to provide a fully natural interface.

The proposed shape recognition solution proved to be viable for real-time implementations and has shown to have a good recognition success ratio despite the low resolution provided by the depth frame. The validation methods proved to be reliable in evaluating a template based approach and leaves no doubts of the good performance of the solution.

As future work more validations are required both testing for different hand shapes and a bigger library so that further conclusions can be made and the threshold parameters may be better tuned. Another challenge is the inclusion of the platform in a Kinect SDK extension. The result would provide programmers the means to develop applications tracking hand gestures as well as the existing body skeleton elements.

This solution recognizes static hand shapes, working with a gesture recognition platform to categorize the movement made by the users' hand. Applying a scenario such as the one proposed in [6] it would be able to recognize sign language in a fully natural environment. This could have many applications such as, for example, serious games devoted to help teach and learn Sign Language, like the Kinect Sign game, proposed in [16].

VI. References

[1] B. A. Myers, *A Brief History of Human Computer Interaction,* 1998.

[2] E. Foxlin, "Motion tracking requirements and technologies," in *Handbook of Computer Science,* 2002, pp. 163-210.

[3] T. Labs, 2013. [Online]. Available: https://www.thalmic.com/en/myo/. [Accessed September 2014].

[4] C. Y. Y. W. H. L. S. X. Xiaoyu Wu, "An Intelligent Interactive System Based on Hand Gesture Recognition Algorithm and Kinect," in *Fifth International Symposium on Computational Intelligence and Design*, 2012.

[5] C. Y. X. W. S. X. H. L. Youwen Wang, "Kinect Based Dynamic Hand Gesture Recognition Algorithm Research," in *4th International Conference on Intelligent Human-Machine Systems and Cybernetics,* 2012.

[6] B. Galveia, T. Cardoso and Y. Rybarczyk, *Adding Value to the Kinect SDK, Creating a Gesture Library,* 2014.

[7] B. P. R. M. a. R. C. Stenger, "Model-Based 3D T racking of an Articulated Hand," 2001.

[8] F. K. Y. E. K. L. A. Cem Keskin, "Real Time Hand Pose Estimation using Depth Sensors," in *IEEE International Conference on Computer Vision Workshops*, 2011.

[9] Y. Li, *Hand Gesture Recognition Using Kinect,* 2012.

[10] M. Panwar, "Hand Gesture Recognition based on Shape Parameters," in *Computing, Communication and Applications (ICCCA), 2012 International Conference*, 2012.

[11] M. P. L. L. A. P. Marco Maisto, "An Accurate Algorithm for the Identification of Fingertips Using an RGB-D Camera," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 272-283, 2013.

[12] J. Y. Z. Z. Zhou Ren, *Robust Hand Gesture Recognition Based on Finger-Earth Mover's Distantce with a Commodity Depth Camera,* 2011.

[13] R. S. J. G. R. S. Murthy, "A Review of Vision Based Hand Gestures Recognition," *International Journal of Information Technology and Knowledge Management,* vol. 2, pp. 405-410, 2009.

[14] V. G. M. O. G. Simion, "Vision Based Hand Gesture Recognition: A Review," *International Journal of Circuits, Systems and Signal Processing,* vol. 6, 2012.

[15] "Moore Neighborhood," Wolfram, [Online]. Available: http://mathworld.wolfram.com/MooreNeighborhood.html. [Accessed August 2014].

[16] J. Gameiro, T. Cardoso and Y. Rybarczyk, "Kinect-Sign, Teaching sign language to "listeners" through a game," in *Conference on Electronics, Telecommunications and Computers*, Lisboa, Portugal, 2013.

[17] J. M. J. Y. Zhou Ren, *Depth Camera Based Hand Gesture Recognition and its Applications in Human-Compiter-Interaction,* 2011.

[18] D. P. Valentino Frati, "Using Kinect for hand tracking and rendering in wearable haptics," in *IEEE World Haptics Conference 2011*, Istanbul, Turkey, 2011.

[19] D. D. E. G. Mark, "The Kinect Up Close: Adaptations for Short-Range Imaging," in *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI),* Hamburg, Germany, 2012.

[20] A. C. K. S. Jagdish L. Raheja, "Tracking of Fingertips and Centres of Palm using KINECT," in *Third International Conference on Computational Intelligence, Modelling & Simulation*, 2011.

[21] L. Y. X. W. S. X. Y. W. Hui Li, "Static Hand Gesture Recognition Based on HOG with Kinect," in *4th International Cinference on Intelligent Human-Machine System and Cybernetics*, 2012.

[22] S. M.-G. A. M. Á.-M. G. D.-S. G. C.-D. Daniela Ramírez-Giraldo, "Kernel Based Hand Gesture Recognition Using Kinect Sensor," in *XVII Simposio de Tratamiento de Señales, Imágenes y Visión Artificial*, 2012.

[23] B. C. C. M. P. S. V. Hewett, "ACM SIGCHI Curricula for Human-Computer Interaction.," 1992. [Online]. Available: http://old.sigchi.org/cgd/cgd2.html. [Accessed August 2014].

[24] A. F. M. C. T. S. M. F. R. M. A. K. A. B. Jamie Shotton, *Real-Time Human Pose Recognition in Parts from Single Depth Images,* 2013.