EPH - International Journal Of Science And Engineering
ISSN (Online): 2454-2016
Volume 07 Issue 02 June 2021

DOI: https://doi.org/10.53555/eijse.v4i1.155

METHOD FOR INDOOR HUMAN POSITION TRACKING USING MULTIPLE DEPTH SENSORS

Algirdas Dobrovolskis[1*], Doc. Audronė Janavičiutė[2] , Prof. Doc. Egidijus Kazanavičius[3] , Doc. Agnius Liutkevičius[4]
*[1,2,3,4]*Real time computer systems centre, Kaunas University of Technology, Lithuania*

Corresponding Author:-
Email: algdobr@ktu.lt

Abstract:-
In this article a positioning method for covering room area was proposed. Multiple Kinect depth sensors were used to work around narrow field of view of one Kinect sensor and cover the room area to prevent blind spots. Affine transformation was used to convert coordinates of the Kinect sensors to the coordinates of the room. Indoor human tracking application was developed in research. During application testing, average aggregated error of 15cm was determined.

Keywords:- *indoor positioning, depth camera, Kinect sensor, method for multiple depth sensors*

1. INTRODUCTION

There is a wide range of applications used for indoor human tracking. Some of them include health care services [1], smart house control and security. The principle behind indoor human tracking is based on multiple technologies, including active and passive RFID [2], ultrasound [3], WLAN solutions [4], Bluetooth positioning defined by Bluetooth Special Interest Group and various Inertial navigation systems [5] that use accelerometers, gyroscopes and, in some cases, even magnetometers. Given technologies use various wearable devices that are not comfortable for a person to use. Moreover, some of them require power to operate thus creating an inconvenient need to charge them constantly.

Another approach is non-invasive systems, such as RGB cameras [6] or even multiple cameras in stereo vision setup [7]. This also has its drawbacks since these systems cannot work in poor lighting conditions. In order to solve that problem, depth cameras were developed, using combined RGB and IR cameras with IR laser for depth measurement. Most affordable of those were Kinect sensors, used primarily in gaming and in various human tracking applications as well [8]. Kinect SDK is provided for sensors, which is useful for reusing raw data processing for human tracking methods. Kinect sensor has its drawbacks too: its field of view is just up to 5 meters and horizontal angle is 57.8°. There have been attempts to solve this problem by using multiple Kinect sensors [9], but that experiment included only two sensors working in parallel. In addition, the location of the sensors was parallel so, inevitably, blind spots were made, also including some intersection of field views.

To cover more area without creating blind spots, different angle positioning of Kinect sensors was developed in this research. Multiple Kinect sensors can be used in different scenarios (limited only by Kinect SDK). Additionally, on-the-fly configuration was developed for extensive sensors repositioning. Using proposed method, area of up to 5 meters in radius can be tracked for human location.

1.1. Affine Transformation.

An affine transformation is a linear (or first-order) transformation that can relate two 2D coordinate systems, typically through a composition of rotations, translations, dilations, and shears [10].

Since each sensor has a separate coordinate system, those coordinates have to be converted to room coordinate system. In figure 1, room coordinate system is x, y whereas sensor coordinate system is x', y'.

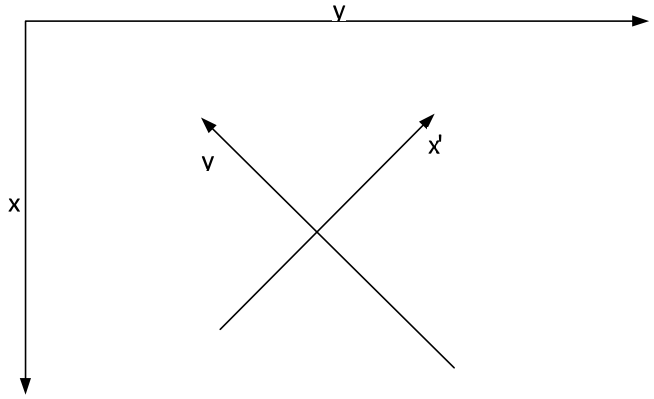


Figure 1 Room and Sensor Coordinate Systems

Affine transformations are expressed algebraically as follows:

$$x' = ax + by + c$$
$$y' = bx + ay + d$$

In matrix notation, this linear system can be written as:

$$\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 \\ -y_1 & x_1 & 0 & 1 \\ x_2 & y_2 & 1 & 0 \\ -y_2 & x_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \Rightarrow u = Mv$$

To determine *a* through *d*, we invert **M** and solve as follows:

$$u = Mv \Rightarrow v = M^{-1}u$$

1.2. Kinect Sensor.

Kinect sensor was developed as a motion controller for Xbox gaming computer. Data from Kinect can be accessed using a computer with appropriate driver through USB 2.0 port.

Kinect components consist of IR laser and IR depth sensor forming a depth camera, RGB camera, tilt motor and microphone array [11].

Kinect depth and RGB cameras can operate at maximum 30 frame rate. Microphone array can be used for speech recognition.

1.3. Kinect for Windows SDK 1.8.

Kinect for Windows SDK 1.8 is the latest API to process Kinect data on a computer, providing various useful functions, such as natural user interface (NUI) skeletal user tracking, face recognition, voice recognition or raw data streams [12]. It uses native code (C++) or managed code (C# or Vb .NET). In addition, accelerometers inside can be used to determine the tilt of a Kinect sensor. Minimal requirements for a PC to run Kinect for Windows SDK are as follows:

o Dual core 2.66GHz 32-bit (*x86*) or 64-bit (*x64*) processor;
o USB 2.0;
o 2 GB RAM;
o Video card with DirectX 9.0c support.

Skeleton position is provided in X, Y and Z coordinates, where Z is the depth from sensor in millimeters.

2. Method for Implementation

Method consists of 3 steps:

1. Calculating the number of sensors needed;
2. Sensor positioning;
3. Sensor calibration.

User has two options to calculate the number of sensors needed:

1. Optimal circle placement based on Lagrange proof, giving 90.6% coverage, shown in Figure 4a;
2. Full coverage circle placement, giving 82.7% sensor area efficiency, shown in Figure 4b.

First option calculates the number of sensors

$$Q = \frac{360xy}{\pi r^2 \alpha}$$

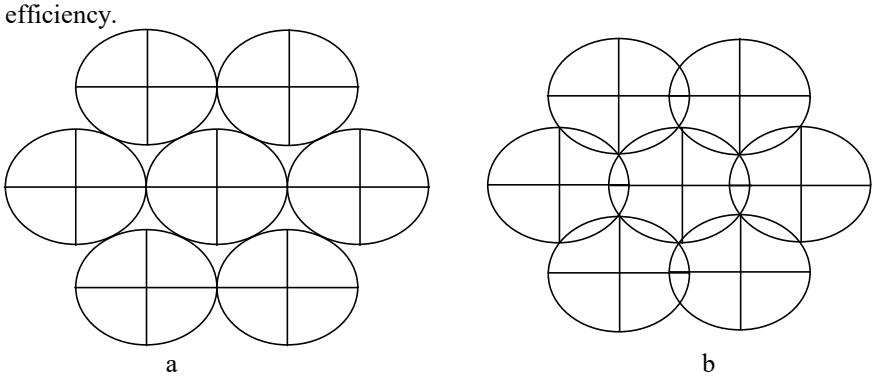
Where x is length of room, y – width of room, r – sensor view length, and α – view angle of sensor.
Second option calculates the number of sensors

$$Q = \frac{240nxy}{r^2\sqrt{3}}$$

Where x – length of room, y – width of room, r – sensor view length, and n – number of sensors needed to form a circle.
In order to position sensors, user applies 3 sensor arrays forming half-circle field of view, as shown in Figure 3.

Figure 2 Sensor Array for Room Coverage

Next step is to place as many arrays as to cover area of the room, forming full circle figures. Figure 4a has optimal sensor efficiency and is recommended for standard usage if 90.6% coverage is sufficient. Figure 4b has full coverage, but only 82.7% sensor field efficiency.



a b

Figure 3 Array Positioning Techniques

Let's say that depth sensor view range is r, and view angle is 90 degrees, room length is 6r and room width is 4r.
Then sensor amount A is:

$$K(a) = \frac{360 * 4 * 6 * r^2}{\pi r^2 * 90} = 30.6$$

Sensor amount B is:

$$K(b) = \frac{240 * 4 * 6 * r^2}{90 * r^2\sqrt{3}} = 37$$

The guidelines for sensor array placement requires for each array to cover as much area as possible, not leaving blind spots bigger than 5 cm in any place of the room. User has to pick room coordinates for each of Kinect sensors calibration and add them into application's configuration file. Coordinates must be in the Kinect sensors' field of view, as shown in Figure 5. Here, zone A is not visible by Kinect sensor and field B is in Kinect's field of view, therefore point 1 is in the B zone, and coordinates X and Y are the ones user needs to add to configuration file. This has to be done for each Kinect sensor in the setup.

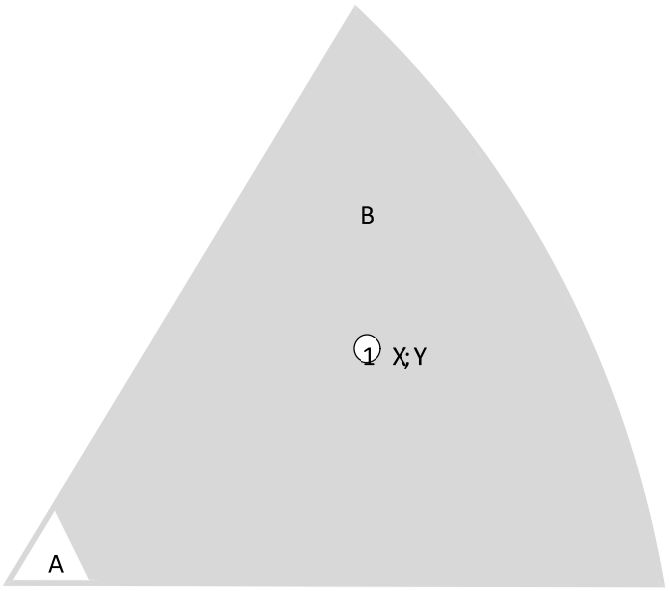


Figure 4 Sensor "Field of View"

When application starts, user can start configuring sensors. In order to do so, he must stand in designated point 1, as shown in Figure 6, and rapidly lift his left hand up, as shown in Figure 8a. This action issues configuration command in the application and initiates coordinates mapping for that sensor. This procedure must be completed for all of the sensors in use in order to get coordinate data from them.

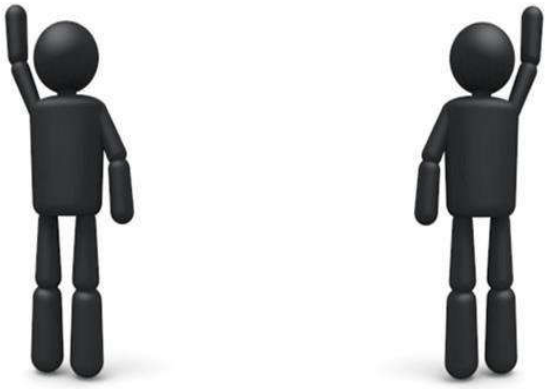


Figure 5 a) Left hand is up – configuration **b) Right hand is up – get coordinates**

After configuration is complete, application is ready for use. To get the current room coordinates the user is in, he has to lift his right arm to issue a command to the application, as shown in Figure 8b. Once the configuration is completed, the application gives X and Y coordinates for user's location, as shown in Figure 7.

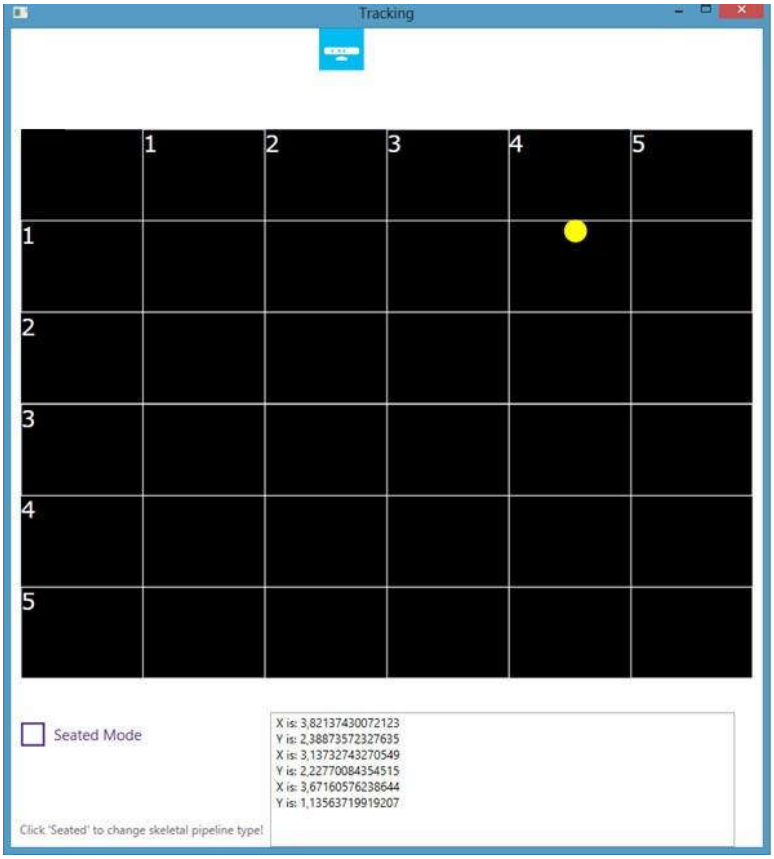


Figure 6 Application Window and Results after Configuration Is Completed

To check if calibration is correct, user can simply stand in the configuration point and issue "get coordinates" command with his right hand. If user gets coordinates that are close to what he put in the configuration file, configuration is correct, otherwise he can repeat the configuration procedure on that specific Kinect sensor.

3. Architecture Design

Application components are shown in Figure 2. Kinect sensors are displayed at the top. Sensors are connected to USB port. It is recommended that USB 3 ports handle high data loads from sensors, but only USB 2.0 is required if used on different physical USB fuses. Kinect SDK collects raw data from sensors and supplies application with tracking information. Application gets the coordinates of a person from Kinect sensor and converts them into room coordinates for further usage.

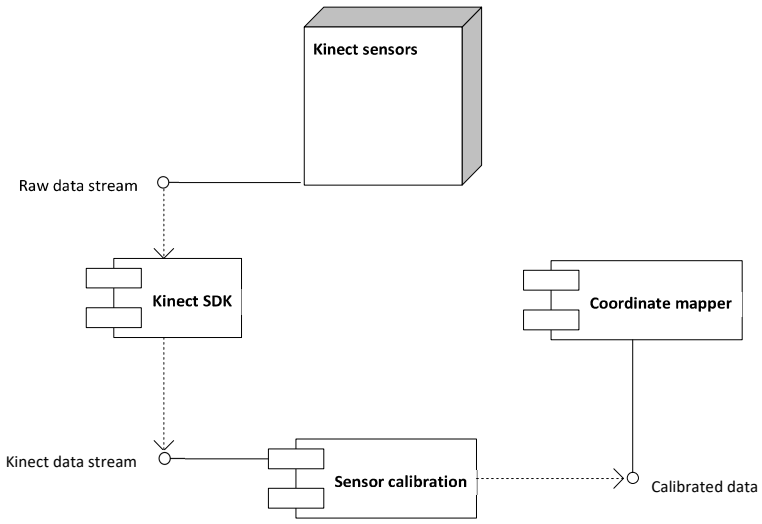


Figure 7 Application Components

4. Results

Three Kinect sensors were used for the experiment. Sensors were aligned to cover most of the room area without view field intersection, as shown in Figure 8. Sensors A, B and C are placed on the table in the middle top of the room. Such sensor positioning guarantees that there are no blind spots larger than 25cm in radius in any point of the room.

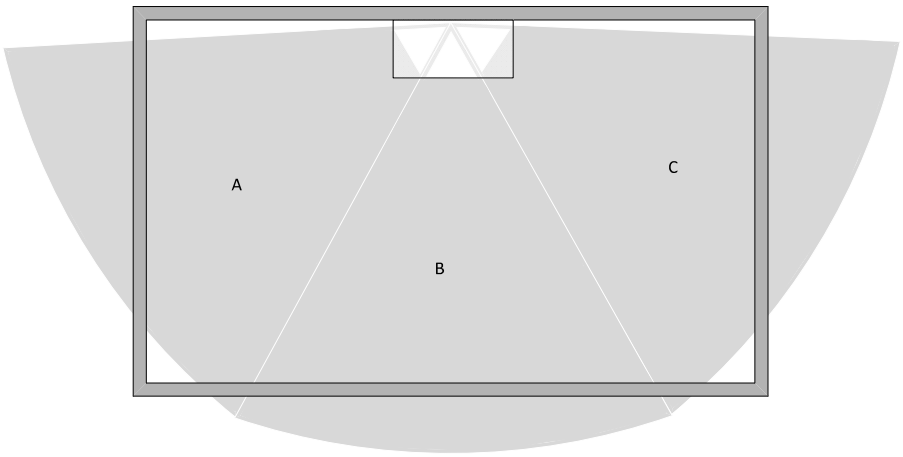


Figure 8 Kinect Sensor Placement Plan

Experiment was conducted picking random points in the room, as shown in Figure 9. Picture shows coordinates of each corner of the room, -as well as coordinates of each test point from 1 to 8.

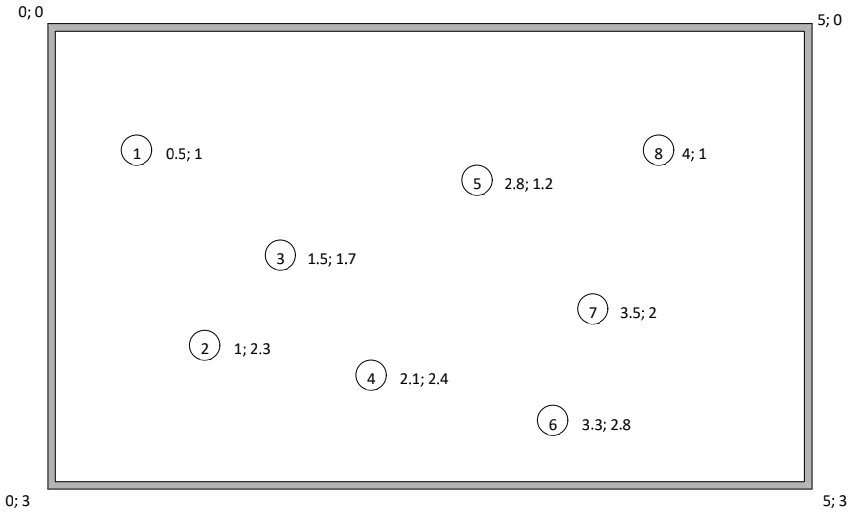


Figure 9 Random Test Points in the Room

Experiment results are provided in Table 1 where real measurements are compared with the coordinates from the test application. Since coordinates calculated from affine transformation are usually an infinite number, they were rounded to 3 decimal places.

Table 1 Error Calculation for Location Tracking

Point	Real coordinates (X; Y)	Application result (X; Y)	Absolute error
1	0.5; 1	0.567; 0.948	0.067; 0.052
2	1; 2.3	1.031; 2.342	0.031; 0.042
3	1.5; 1.7	1.585; 1.794	0.085; 0.094
4	2.1; 2.4	2.199; 2.306	0.099; 0.094
5	2.8; 1.2	2.894; 1.208	0.094; 0.008
6	3.3; 2.8	3.431; 2.824	0.133; 0.024
7	3.5; 2	3.638; 1.979	0.138; 0.021
8	4; 1	3.912; 1.094	0.088; 0.094

The average error for coordinate X was 0.092m, or about 9cm, whereas the average error for coordinate Y was 0.054, or about 5cm. Error distribution graph is shown in Figure 10. As can be seen from the graph, the maximum error for coordinate X was 0.138m, or about 14cm, and maximum error for coordinate Y was 0.094m, or about 9cm.

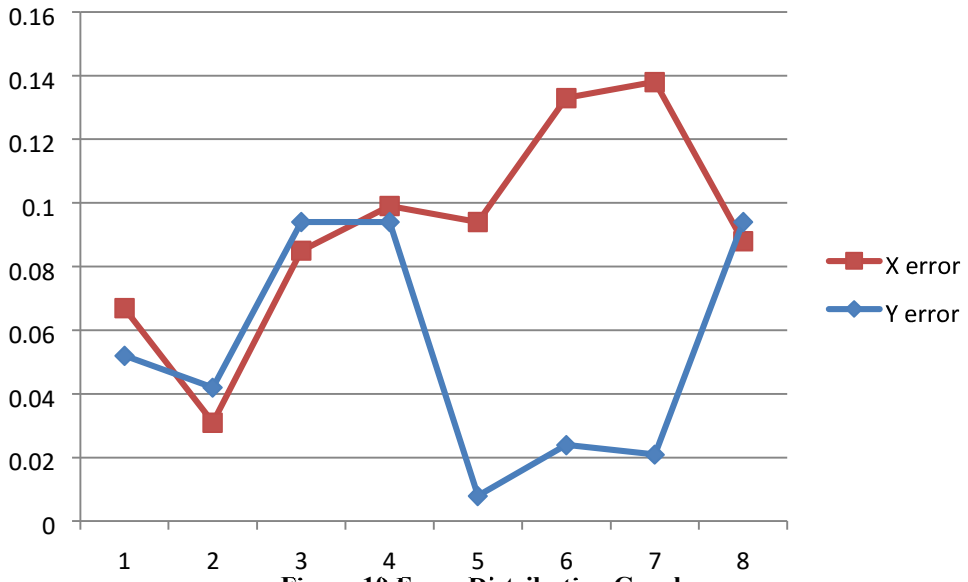


Figure 10 Error Distribution Graph

5. Summary

Method for indoor human tracking using multiple Kinect sensors was introduced in this article. Coordinate mapping application was created to test the method in actual room conditions using 3 Kinect sensors. On-the-fly configuration was developed for extensive repositioning of Kinect sensors. Experiment was conducted to determine the average error in human location tracking, it was 9cm for X coordinate and 5cm for Y coordinate. None of the results exceeded 0.2m of aggregated error for both coordinates, and the average aggregated error was 0.146m, or about 15cm. For future research, the application could be expanded to distribute human tracking on several computers to circumvent Kinect SDK limitations.

References

[1]. Toplan, E. ; Bilgisayar Muhendisligi Bolumu, Bogazici Univ., Istanbul, Turkey; Ersoy, C. „RFID based indoor location determination for elderly tracking“

[2]. F. Seco, C. Plagemann, A. R. Jiménez, and W. Burgard, “Improving RFID-based indoor positioning accuracy using Gaussian processes,” in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, 2010, pp. 1–8.

[3]. H. Schweinzer and M. Syafrudin, “LOSNUS: An ultrasonic system enabling high accuracy and secure TDoA locating of numerous devices,” in *2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2010, pp. 1–8.

[4]. R. Battiti, N. T. Le, and A. Villani, “Location-aware computing: a neural network model for determining location in wireless LANs,” 2002.

[5]. C. Lukianto and H. Sternberg, “STEPPING–Smartphone-Based Portable Pedestrian Indoor Navigation,” *Mmt*, 2011.

[6]. R. Bodor, B. Jackson, and N. Papanikolopoulos, “Visionbased human tracking and activity recognition”, in Proc. Of The 11th Mediterranean Conference on Control and Automation, 2003.

[7]. J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, “Multicamera multi-person tracking for easyliving,” in *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on*, 2000, pp. 3–10.

[8]. L. Xia, C.-C. Chen, and J.K. Aggarwal. (2011). “Human Detection Using Depth Information by Kinect", International Workshop on Human Activity Understanding from 3D Data in conjunction with CVPR (HAU3D), Colorado Springs, CO.

[9]. Muhamad Risqi Utama Saputra, Widyawan, Guntur Dharma Putra, Paulus Insap Santosa Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada „Indoor Human Tracking Application Using Multiple Depth-Cameras“.

[10]. “How to map points between 2D coordinate systems” [online] http://msdn.microsoft.com/en-us/library/ie/jj635757%28v=vs.85%29.aspx [Accessed: 05-Mar-2019].

[11]. “Kinect for Windows Features“. [Online]. https://blogs.microsoft.com/ai/seewhats-new-in-kinect-for-windows-sdk-1-8/ [Accessed: 05-Mar -2019].

[12]. “Kinect for Windows SDK“. [Online]. https://www.microsoft.com/enus/download/details.aspx?id=40278 [Accessed: 05-Mar -2019].