Indoor Human Tracking Application Using Multiple Depth-Cameras

Muhamad Risqi Utama Saputra, Widyawan, Guntur Dharma Putra, Paulus Insap Santosa
Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada
Grafika No. 2 Yogyakarta, 55281, Indonesia
risqi@te.gadjahmada.edu, widyawan@ugm.ac.id, guntur.dharma@mail.ugm.ac.id, insap@mti.ugm.ac.id

***Abstract*—This research developed an application that could tracks and locates human's presence and position in indoor environment using multiple depth-cameras. Kinect as the most affordable device that equipped with depth-camera was used in this work. The application obtains stream data from Kinect and analyzes presence of human using skeletal tracking library on Kinect for Windows SDK v1. The final application also visualizes human location on 3D environment using Windows Presentation Foundation (WPF) 4.0. In order to visualize 3D object correctly, the application also took into account the coverage that may intersect when two Kinects were placed in adjacent position so that the final human location is combined. In the end, application was tested in 3 different scenarios and it's found that the average error in determining human location was 0.13589 meters.**

I. Introduction

IN this modern era, the needs of indoor human tracking application, an application that could tracks and locates human's position in indoor environment, are increasing. Some of the needs are (1) to monitor elderly people's condition when the family was not with them, (2) to develop security system, (3) to make context-aware application, an application that can adapts changing condition and environment, for example household automation and interactive environment, and (4) to create rehabilitation games for recovery of lost motor functions.

Some researches associated with indoor human tracking application have ever been done before, but it had some drawbacks. First, human had to use troublesome wearable device, such as cricket [1] and active badge [2]. Second, some researchers used computer vision technology to track human utilizing RGB camera [3],[4]. In some cases including monitoring elderly people, utilizing RGB camera was not expected because it could be considered as invasion of privacy. Third, the installation of the system was still too expensive because it used extremely large device, such as active floor [5] or pressure-sensitive floor (EMFi floor) [6].

As a solution, depth-camera could be used to track and locate human's position in indoor environment without the needs of using wearable device and utilizing RGB camera. One of the most widely used devices that equipped with depth-camera was Kinect, an additional device for Xbox 360 launched by Microsoft for less than £100 per unit [7]. Using inexpensive depth-camera as Kinect, it is possible to develop indoor human tracking application that could eliminate three flaws in previous research.

Nevertheless, Kinect also has a drawback. It has a very limited coverage because it was designed for indoor gaming device. Kinect could only detect human at distance 0.8 – 4 meters with 57^0 horizontal field of view (approximately 8.3 m^2) [8]. Therefore, this research was conducted to develop an indoor human tracking application by using multiple Kinects so that the coverage limitation of using one Kinect can be resolved. In this research, the number of Kinect to be used was two. Some researches focus on human detection method using depth image have ever been carried out before, as has been done by [9]. Instead of developing new method for detecting human on depth image, this research focuses on developing potential useful application using existing method and improve the capabilities using multiple devices.

The application obtains stream data from Kinect using official Software Development Kit (SDK) from Microsoft called Kinect for Windows SDK v1. With those data, application could tracks and locates human's position relative from Kinect and visualizes it in 3D environment using Windows Presentation Foundation (WPF) 4.0. The 3D model itself was created using Blender 3D Computer Graphics 2.48a. Application also took into account the coverage that may intersect when two Kinects were placed in adjacent position so that the final human location inside that place is combined. Finally, the application was tested with three different scenarios to find out error from the process of detecting human location.

II. Literature Review

A. Depth-camera

Depth-camera is a device that can directly senses range to nearest physical surface at each pixel location. Depth-camera is unique because it could enables affordable real time 3D modeling of surface geometry and makes some difficult computer vision problems easier, such as removing false background in a video conference application [10]. Some of technologies used to obtain depth information from depth-camera are time-of-flight infrared and structured light [11].

B. Kinect

Kinect is a motion sensing device by Microsoft that enables users to control and naturally interact with games or other programs without the need to physically touch with controller. Kinect performs this through natural user interface technologies by tracking user’s body movement, gestures, and spoken commands. As shown in Fig. 1, Kinect consists of 3D depth sensor (using an infrared projector and an infrared camera), RGB (Red-Green-Blue) camera, multi-array microphones, and a motor for tilting mechanism. Depth sensor interprets depth information from projection of infrared emitted by infrared projector. Multi-array microphones contribute on speech recognition and localization based on beam angel of sound. Kinect also uses accelerometer to detect tilt and stabilize image [7].

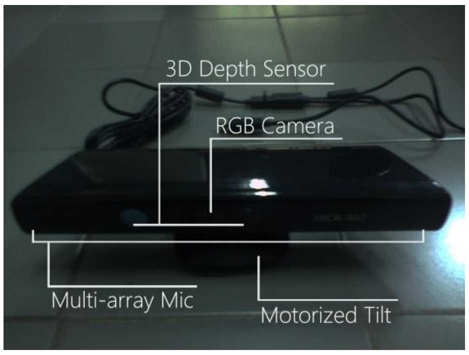


Fig. 1. The main parts of Kinect are 3D depth sensor, RGB camera, motorized tilt, and multi-array microphone.

To measure depth information, the inventors described that it was used triangulation method. Infrared projector emits randomly speckles pattern of infrared and it is captured back by infrared camera. The captured pattern is correlated against a reference pattern afterward. The reference pattern itself is obtained by capturing a plane at a known distance from sensor and is stored in a memory. When a speckle is projected on an object whose distance to the sensor is smaller or larger than the reference plane, the position of speckle in infrared image will be shifted in the direction of the baseline between the infrared projector and infrared camera. These shifts are measured using image correlation algorithm which yields a disparity image. With this disparity image, the distance information in every pixel can be retrieved using triangulation formula [12].

C. Kinect for Windows SDK v1

Kinect for Windows SDK is a tool and Application Programming Interface (API) which helps developer to create Kinect enabling application on Windows operating system, whether it uses native code (C++) or managed code (C# or Vb .NET). Kinect for Windows SDK provides library to consume stream data from Kinect, such as images data, audio input, and skeletal data. There are 3 images data provided by Kinect for Windows SDK. They are color image data, depth image data, and player segmentation data. Color image data generates image as appears on RGB camera. Depth image data provides frames which every pixel contains distance in millimeters (mm) from camera to nearest object on x and y coordinates in Kinect field of view. Whereas player segmentation data processes stream data to identify up to 6 humans in front of Kinect and creates segmentation map. This segmentation map is a bitmap, which every pixel indicates player index from nearest detected human [8].

Moreover, Kinect for Windows SDK also have Natural User Interface (NUI) API feature called skeletal tracking. Skeletal tracking provides skeleton information and position of human in front of Kinect. There are two kind of skeletal tracking, i.e. active skeletal tracking and passive skeletal tracking. Active skeletal tracking provides full skeleton data (skeleton joints and human’s location), whereas passive skeletal tracking provides only location of human. The position of skeleton and each of the skeleton joints are stored as (X, Y, Z) coordinates. Skeleton space coordinates are expressed in meters. The (X, Y, Z) axes are the body axes of the depth sensor as shown in Fig. 2 [8].

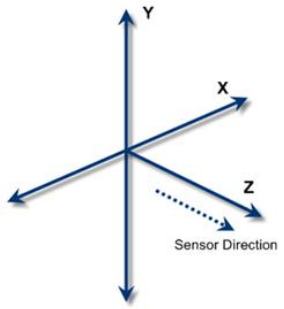


Fig. 2. Coordinates space of skeleton data provided by Kinect for Windows SDK [8].

III. Architecture Design

The architecture of indoor human tracking application is divided into 3 main layers as shown in Fig. 3. They are hardware, Kinect SDK, and application. First, hardware layer shows that each Kinect has to be connected with different USB controller bus because of large bandwidth usage. One USB controller bus can’t handle 2 incoming stream data from 2 Kinects, especially depth data. Then, Kinect SDK layer consists of Kinect drivers and Natural User Interface (NUI) library which is provided by Kinect for Windows SDK. This is a bridge between hardware layer and application layer that provides depth data and skeletal tracking data which is used later by Application layer.

Application layer is the main layer that completely developed by this research. This layer describes that indoor human tracking application is divided into 3 sub applications. It was done because to track and retrieve human's location, application uses skeletal tracking data from Kinect for Windows SDK which work only on 1 process or 1 application. Those 3 sub applications are 2 client applications for accessing skeletal tracking data from each Kinect and 1 server application that obtains those data, processes it, and visualizes it in 3D environment.

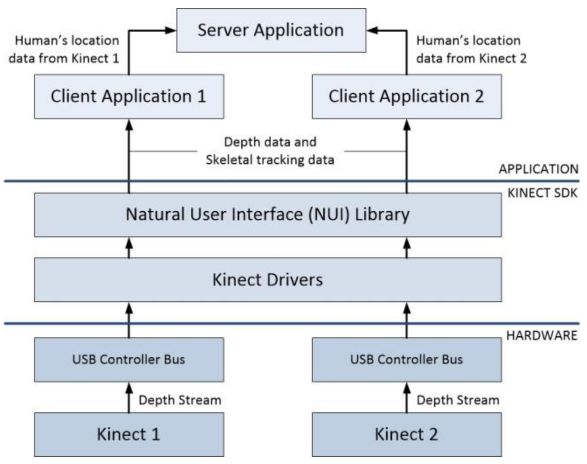


Fig. 3. It's architecture of indoor human tracking application. It's divided into 3 main layers: hardware, Kinect SDK, and application.

IV. Implementation

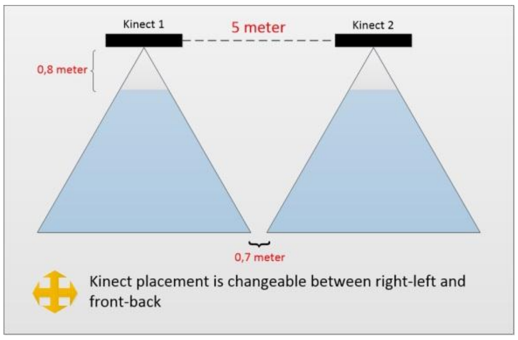
A. System Requirements

For the purpose of development and testing, the application needs:

1. Hardware
   a. Kinect for Xbox 360 (2 devices).
   b. A computer with minimum 2 GHz processor, 2 GB RAM, 30 GB hard disk space, and DirectX 9 graphic device with WDDM 1.0.
   c. Express card to USB 2.0/USB 3.0.
2. Software
   a. Windows 7 Basic Edition operating system.
   b. Microsoft Visual C# 2010 Express.
   c. Blender 3D Computer Graphic 2.48a.
   d. Kinect for Windows SDK v1.
   e. XAML Exporter for Blender.

B. Kinect Placement

In order to visualize human location correctly in 3D environment, user should place two Kinects in the right position. To ease user on Kinect placement, application provides Kinect placement template. The template is two Kinects should be placed parallel each other and the coverage should be in same direction as shown in Fig. 4. Fig. 4 also describes the distance of two Kinects is 5 meters. It is just initial distance which is created to ease placement of 3D object that represents Kinect in application's 3D environment. User can change distance, move the Kinect right-left or front-back as desired.



Kinect 1
5 meter
Kinect 2
0,8 meter
0,7 meter
Kinect placement is changeable between right-left and front-back

Fig. 4. This is Kinect placement template which is provided by application. User should place two Kinects parallel each other and each coverage area should be in the same direction.

C. User Interaction

The user of indoor human tracking application can adjust the setting of application before it's used to track human. This is the list of tasks which user can do (see Fig. 5 for to view user interface):

1. Import 3D indoor environment object, a 3D object that represents real indoor environment such as family room, laboratory, etc. The format of 3D object is XAML. User can create this object using Blender and export it to XAML file using XAML Exporter for Blender.
2. Move 3D indoor environment object to desirable specific coordinate in 3D coordinates system.
3. Identify each Kinect, which one is connected to client application 1, which one is connected to client application 2. It can be done by entering Kinect ID in each client applications.
4. Choose Kinect placement template and place 3D Kinect object on 3D environment.
5. Move 3D Kinect object to specific coordinate in 3D coordinates system. It is tailored with the need of user and Kinect placement in actual condition.

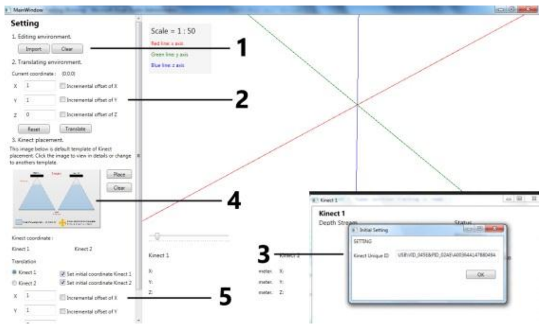


Fig. 5 On server application, user can (1) imports 3D indoor environment object, (2) moves it to desirable coordinates, (3) identifies each Kinect, (4) chooses Kinect placement template, and (5) moves 3D Kinect object to specific coordinates.

D. Tracking Process and Communication between Client and Server

The process of tracking human location begins with initializing Kinect's connectivity and prepares depth frames by client application, including setup frame resolution. For every incoming frame, application puts every transferred byte to buffer memory. Then, application converts each byte to monochrome for displaying it on monitor, with closer depth information will be colored white, while farther depth information will be colored black. By using player segmentation data on Kinect SDK, application enables to detect the presence of human in every frame and marks the pixels with gold color. Finally, using skeletal tracking feature on Kinect SDK, each client application retrieves human's location in x, y, and z coordinates, accord with Kinect's skeleton space.

For purpose of displaying human location in 3D environment, client application should transfer information about human's location to server application. Communication between client application and server application is carried out using shared-memory feature on Windows OS, which is called Pipes. First, server application should initiates pipes server, opens the connection, and waits for incoming data. Then, when client application is running, it automatically initiates pipes client and sends data to server application. If the data has been delivered successfully, server application closes the connection and prepares for the next communication.

In order to achieve understanding communication between client application and server application, the data exchange format should be defined clearly. This application is used string as type of data exchange. The format is:

"KinectUniqueID AppNumber DetectedStatus X Y Z"

The explanations of data exchange format above are:

1. KinectUniqueID: ID of Kinect. Every Kinects has different ID.
2. AppNumber: Identity number of client application which sends the data. The value is 1 or 2. It's useful to visualize 3D Kinect object.
3. DetectedStatus: the value is DETECTED when application detects presence of human.
4. X: Location of detected human in X axis based on skeleton space coordinates.
5. Y: Location of detected human in Y axis based on skeleton space coordinates.
6. Z: Location of detected human in Z axis based on skeleton space coordinates.

E. Calculation of Human Location in Intersection of Two Coverage Kinects

The first step to calculate detected human's location on intersection of two coverage Kinects is to detect whether the human is present on that area or not. This is done simply by checking incoming data from two client applications. If two client applications send skeletal tracking data on the same time (the value of DetectedStatus property is assigned with "DETECTED"), it means the human is present on intersection area. Then, the calculation itself is done by finding the middle point between two incoming location values from two client applications. It's done by computing average value of two human locations with location from Kinect 1 as reference point. Here details of the process:

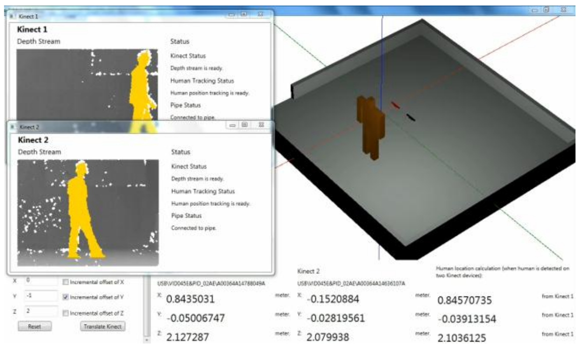
1. Calculate distance placement (x and y) of two Kinects object in 3D environment.
2. Determine the position of two Kinects based on Kinect object's coordinate in 3D environment.
3. Convert distance placement of two Kinect (x and y) to units of meter.
4. Calculate final human location based on the (1) human position and (2) Kinect's position. Table 1 shows all of formulas used to calculate final human location based on those conditions.

Table 1
Calculation Of Final Human Location Based On Human Position And Kinect Position

Final calculation of X coordinate	
Human's position is on the left side of two Kinects (each value of detection result is positive)	
Kinect 1 is on the left of Kinect 2 ($X_{Kinect1}$ is less than $X_{Kinect2}$)	$X_{Final} = \frac{X_{Kinect1}+X_{Kinect2}+X_{Actual}}{2} - X_{Actual}$
Kinect 1 is on the right of Kinect 2 ($X_{Kinect1}$ is greater than $X_{Kinect2}$)	$X_{Final} = \frac{X_{Kinect1}+X_{Kinect2}+X_{Actual}}{2}$
Human's position is on the middle of two Kinects (one value of detection result is positive, the other is negative)	
Kinect 1 is on the left of Kinect 2 ($X_{Kinect1}$ is negative and $X_{Kinect2}$ is positive)	$X_{Final} = -\left(\frac{\lvert X_{Kinect1}\rvert + (X_{Actual} - \lvert X_{Kinect2}\rvert)}{2}\right)$
Kinect 1 is on the right of Kinect 2 ($X_{Kinect1}$ is positive, $X_{Kinect2}$ is negative)	$X_{Final} = \frac{\lvert X_{Kinect1}\rvert + (X_{Actual} - \lvert X_{Kinect2}\rvert)}{2}$
Human's position is on the right side of two Kinects (each value of detection result is negative)	
Kinect 1 is on the left of Kinect 2 ($\lvert X_{Kinect1}\rvert$ is greater than $\lvert X_{Kinect2}\rvert$)	$X_{Final} = \frac{X_{Kinect1}+X_{Kinect2}-X_{Actual}}{2} + X_{Actual}$
Kinect 1 is on the right of Kinect 2 ($\lvert X_{Kinect1}\rvert$ is less than $\lvert X_{Kinect2}\rvert$)	$X_{Final} = \frac{X_{Kinect1}+X_{Kinect2}-X_{Actual}}{2}$
Final calculation of Z coordinate	
Kinect 1 is in front of Kinect 2	$Z_{Final} = \frac{Z_{Kinect1}+Z_{Kinect2}+Z_{Actual}}{2} - Z_{Actual}$
Kinect 1 is behind Kinect 2	$Z_{Final} = \frac{Z_{Kinect1}+Z_{Kinect2}+Z_{Actual}}{2}$

F. 3D Visualization

Visualization of tracking human's location in 3D environment is carried out based on final human location data. The final human location data should be converted into 3D coordinate values based on the scale used by application. Then, those values are summed with coordinate position of object that represents two Kinect devices in a 3D system. Finally, using the final coordinate values, 3D object that represents humans is displayed on the screen. Fig. 6 shows the final results of 3D visualization.



Kinect 1
Depth Stream
Status
Kinect Status
Depth stream is ready.
Human Tracking Status
Human position tracking is ready.
Pipe Status
Connected to pipe.

Kinect 2
Depth Stream
Status
Kinect Status
Depth stream is ready.
Human Tracking Status
Human position tracking is ready.
Pipe Status
Connected to pipe.

```
X 0    [ ] Incremental offset of X
Y -1   [✓] Incremental offset of Y
Z 2    [ ] Incremental offset of Z
Reset   Translate Kinect

USB\VID045E&PID_02AE\A00364A14788049A    Kinect 2
                                         USB\VID045E&PID_02AE\A00364A14636107A    Human location calculation (when human is detected on two Kinect devices)
X: 0.8435031    meter   X: -0.1520884    meter   0.84570735    from Kinect 1
Y: -0.05006747  meter   Y: -0.02819561   meter   -0.03913154   from Kinect 1
Z: 2.127287     meter   Z: 2.079938      meter   2.1036125     from Kinect 1
```

Fig. 6. Final human location is visualized in 3D environment.

V. Experimental Result

Experiment was conducted with 3 different scenarios to get the average error from detection human location process. Scenario 1, two Kinects were placed in parallel right-left with 5 meters distance and there is no intersection coverage. This condition is suitable for human tracking implemented in a wide space, such as laboratory or stage presentation. Scenario 2, two Kinects were placed parallel with 1 meter distance and of course there is intersection area. This condition is suitable for the implementation of human tracking in a narrow space like the family room or the patient room. Scenario 3, two Kinects were placed front-back with 1.5 meters distance and there is intersection area. This condition is suitable for the implementation of human tracking in building hallways. Fig. 7 shows illustration of three scenarios above.

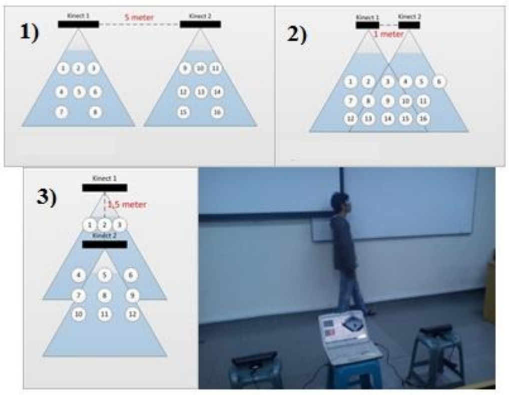


Fig. 7. Three different scenarios for experiment: 1) two Kinects are placed parallel right-left with 5 meter distance, 2) two Kinect are placed parallel right-left with 1 meter distance, and 3) two Kinect are placed front-back with 1.5 meter distance.

Table 2 shows error of detection human location from every scenario. Based on result in three different scenarios, average error of this indoor human tracking application is 0.13589 meter.

TABLE 2
ERROR CALCULATION OF HUMAN LOCATION FOR EVERY SCENARIO (METER)

Scenario	Kinect 1		Kinect 2		Calculation of human location in intersection area	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
1	0.11689	0.0217	0.11937	0.01993	-	-
2	0.10223	0.04797	0.10654	0.05083	0.1073	0.03829
3	0.18607	0.08795	0.14729	0.07004	0.2014	0.0335

Fig. 8, Fig. 9, and Fig. 10 show error for every scenario in chart bar. From those three charts, it's known that standard deviation of error is wider on scenario that has intersection area, that is Scenario 2 and 3. This is happened because infrared spackle from two Kinects overlapped each other so that it's difficult for Kinect's processing machine to determine which one infrared spackle to be calculated. Furthermore, the highest average error is on Scenario 3. This is happened because every projected infrared spackle from two Kinects on Scenario 3 is overlapped each other.

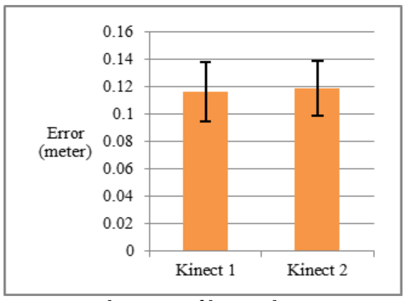


Fig. 8. Average error detection of human location in scenario 1.

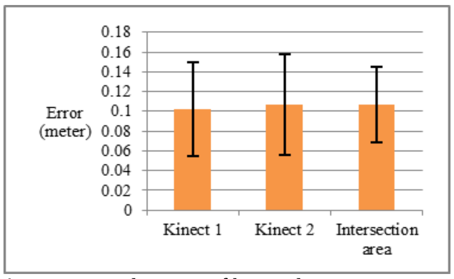


Fig. 9. Average error detection of human location in scenario 2.

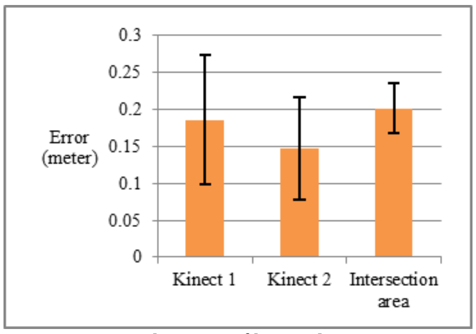


Fig. 10. Average error detection of human location in scenario 3.

Fig. 11 shows error pattern on Z and X coordinate against shift of measurement in actual condition. From the graph, it's shown that error of detection human location in Z coordinate changes linearly with shift of human in real condition. This means that error of detection human location in Z coordinates change proportionally against shift of measurement in real condition. If distance of human from Kinect in Z coordinates more far away, then the error of detection human location is bigger. But, it's not happened with error of detection human location in X coordinate. Error on X coordinate is not makes any particular pattern and not leveraged by shift of measurement in real condition.

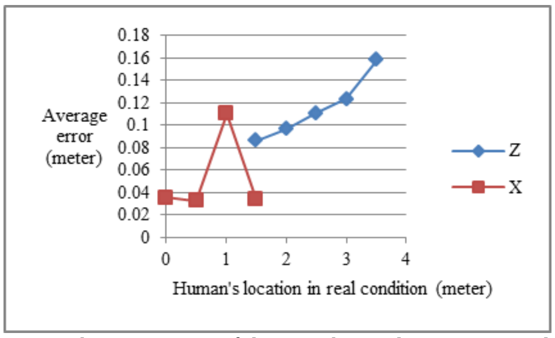


Fig. 11. The pattern error of detection human location in X and Z coordinates compared to actual condition.

VI. Conclusion

This paper presents development of indoor human tracking application using 2 depth-cameras. Tracking human's location was conducted real time using player segmentation data and skeletal tracking feature on Kinect for Windows SDK v1. Stream data from every Kinect was processed by client application and distribute it to server application using Pipes mechanism on Windows operating system.

The process of detection human location on intersection of two coverage Kinects was conducted by examine whether two Kinects sending skeletal tracking data simultaneously or not. To calculate the final location of detected human in intersection coverage was done by calculating the middle point of two location data from two Kinects. The final result was visualized using Windows Presentation Foundation 4.0 based on final calculation of human location and position of 3D Kinect object. From the experimental result, it's shown that average error of application in detecting human location is 0.13589 meters.

Indoor human tracking application that has been developed in this research still has a lot of potential improvement. Applications can be developed to detect a lot of people at once and be able to visualize it correctly for various placements of Kinect. For future work, the results of this research can be further used for development of a more advanced tracking system or context-aware application.

References

[1] N. B. Priyantha, "The cricket indoor location system", Doctoral Thesis, Departement of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2005.

[2] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system", *ACM Transactions on Information Systems*, vol. 10, pp. 91-102, 1992.

[3] R. Bodor, B. Jackson, and N. Papanikolopoulos, "Vision-based human tracking and activity recognition", in *Proc. Of The 11th Mediterranean Conference on Control and Automation*, 2003.

[4] J. Choi, Y. Cho, K. Cho, S. Bae, and H. S. Yang, "A view-based multiple objects tracking and human action recognition for interactive virtual environments", *International Journal of Virtual Reality (IJVR)*, pp. 71-76, 2008.

[5] M. D. Addlesee, A. H. Jones, F. Livesey, and F. S. Samaria, "The ORL active floor", *IEEE Personal Communications*, 5(4), pp. 35-41, 1997.

[6] S. Pirttikangas, J. Suutala, J. Riekki, and J. Roning, "Footstep Identification from pressure signals using hidden markov models", *Finnish Signal Processing Symposium (FINSIG'03)*, 2003.

[7] M. N. K. Boulos, B. J. Blanchard, C. Walker, J. Montero, A. Tripathy, and R. G. Osuna, "Web GIS in Practice X: a Microsoft Kinect Natural User Interface for Google Earth Navigation", *International Journal of Health Geographics*, 10:45, 2011.

[8] Microsoft Developer Network. (2012). Kinect for Windows SDK [Online]. Available: http://msdn.microsoft.com/en-us/library/hh855347.

[9] 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.

[10] A. D. Wilson and H. Benko, "Combining multiple depth cameras and projectors for interactions on, above, and between surfaces", *UIST '10 Proceedings of the 23nd annual ACM symposium on user interface software and technology*, 2010.

[11] A. Bogomjakov, Craig Gotsman, and Marcus Magnor, "Free-viewpoint video from depth cameras", *Proc. Of Vision, Modeling, and Visualization (VMV)*, pp. 89-96, November, 2006.

[12] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications", *Sensors*, 12, pp. 1437-1454, 2012.