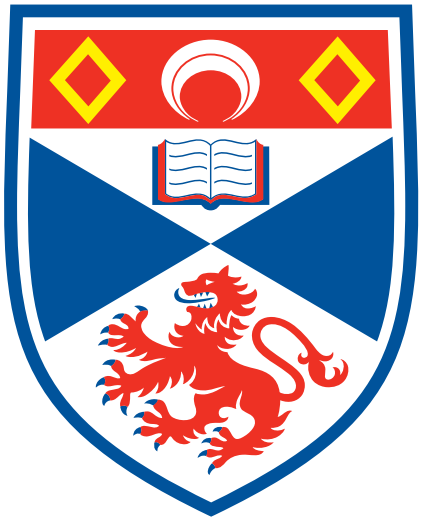
Tracking People with Multiple Kinects in Occluded Environments



University of
St Andrews

CS4099: Major Software Project

Author:
Chi-Jui Wu

Supervisor:
Dr. David
Harris-Birtill

April 9, 2015

Abstract

The current work is about tracking people with multiple Kinects in occluded environments. The final submission contains an interactive software for demonstrating the tracking system, a report describing the current work and a series of user studies intended to evaluate the system. Strengths and limitations of the system are discussed. An outline of future work is presented.

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is NN,NNN words long, including project specification and plan.

Contents

List of Figures

CHAPTER 1

Introduction

People detection and tracking is the process of identifying and following people in an environment.

The problem of detection and tracking is a challenging issue in surveillance, interactive systems, medical imaging, and humanoid robotics. why is it important

human traffic Counting people in crowds with a real-time network of simple image sensors Counting Crowded Moving Objects

human behaviour analysis A survey of advances in vision-based human motion capture and analysis

The current work proposes an algorithm for tracking people with multiple Kinects which resolves the problem of occlusion.

1.1 Problem Statement

Tracking depends on initial detection results. The current work leverages Kinect sensor's ability to identify people from a depth map. This allows the researcher to focus on the problem of tracking.

Tracking moving objects is non-trivial. There are many sources of tracking errors, including raw sensor data noise, illumination levels, changing backgrounds, and occlusion. Tracking in real-life scenarios are even harder. The environment is unpredictive and complex, often consisting of multiple people. A crowded environment with complex human interactions gives rise the the problem of occlusion.

Occlusion occurs when the tracked target is masked by other objects in the scene. The masked target would not exist in the field of view of one or more cameras. If a person were occluded, his precise joint positions and movements would be unknown. Resolving the problem of occlusion would provide any tracking system with more spatial and physiological information about the tracked people.

There are two types of occlusions: static and dynamic. They are defined as:

Static occlusion Occlusion caused by stationary objects in the environment

Dynamic occlusion Occlusion caused by people interactions in the environment

A simple instance of the problem is illustrated in Figure 1.1. In the figure, both skeletons are invisible to the front Kinect but visible to the side Kinect. They are occluded by the red obstacle. When they step out of the obstacle into the views of both Kinects, the system should merge the skeletons of the same person from different perspectives. The main objective of the project is to avoid occlusion by extending the field of view of the system. The proposed algorithm would combine depth sensor information from multiple Kinects to achieve this goal.

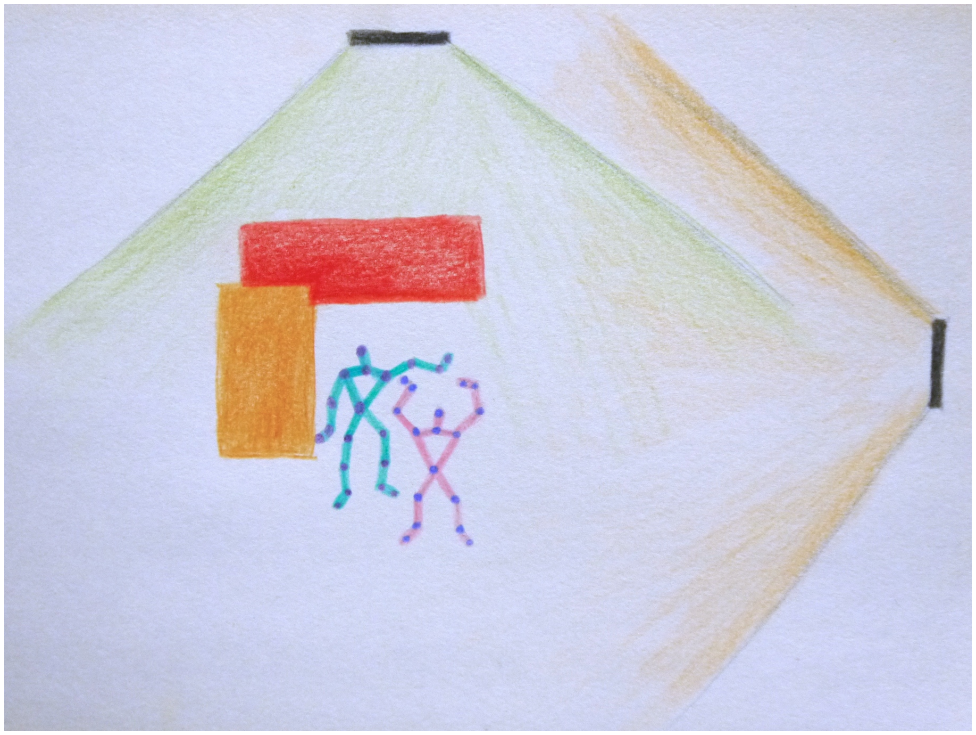


Figure 1.1: The occlusion problem

The current work addresses the occlusion problem by extending the field of view with multiple Kinects. The field of view, also known as FOV, is the extent to which the scene is observable through the camera. By extending the field of view, the system will have a more complete view of the environment in 3D space that was not available with a single camera.

1.2 Contributions

The contributions of the current work are...

1. Replicate, validate, and extend current research
2. A Kinect BodyFrame serialization library
3. A Kinect client-server framework
4. Track people with multiple Kinects
5. Integrate joints information from multiple Kinects to resolve the occlusion problem
6. User studies showing the strengths and weaknesses of the current system

1.3 Kinect

Kinect is a commodity depth sensor for body motion capturing and tracking. It provides depth and RGB streams at 30 frames per second [**?**]. In addition, it also provides infrared and audio streams at lower frame rates.

The current work uses the Kinect BodyFrame stream, which includes the skeletal information of the people in the sensor's field of view. The complete API reference for the Kinect v2 SDK is accessible at `https://msdn.microsoft.com/en-us/library/windowspreview.kinect.aspx`.

The system manipulates coordinates in the Kinect Camera Space. These coordinates are 3D points, consisted of the x, y, and z components, in meters. The origin of the coordinate system at (0, 0, 0) is "located at the center of the IR sensor on Kinect" [1]. The x axis grows to the left of the sensor, the y axis grows upward from the sensor, and the z axis grows outward from the direction the sensor (See Figure 1.2).

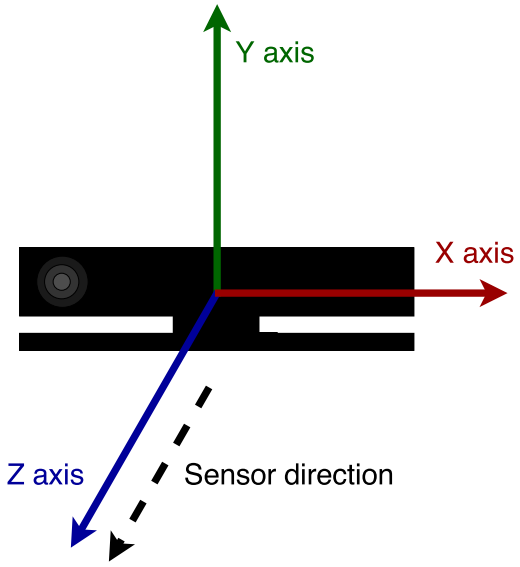


Figure 1.2: The Kinect Camera Space

CHAPTER 2

Background

This chapter will review the the state of the art people detection and tracking techniques.

The problem of people detection and tracking has been researched extensively in surveillance video, using techniques such as particle filters [2], background subtraction techniques and color histograms [3], 3D pose estimation [4], occlusion-aware people dectors [5], and clustering the motion paths of local features [6].

With the advent of time-of-flight cameras like the Kinect,

2.1 Tracking by detection

Tracking by detection is the primary approach to people tracking. The approach leverages the technology of reliable, inexpensive RGB-D sensors, such as the Microsoft Kinect. Person detection has been performed using depth information [12, 13, 14], and with Kinect depth data [18, 19]. Others use only RGB data [1, 15, 16]. Methods using a combination of appearance and depth information also show promising results [5, 17].

Tracking by detection is often applied in RGB-depth based tracking. In general, color information is great at distinguishing people who are far apart from each other and look somewhat different. Depth information provides more clues to the spatial position of each person within a large group.

RGB-based tracking invovles.

Depth-based tracking.

Kalman filter

2.2 Tracking with Kinect

This section describes a number of different methodologies of tracking people using the Kinect in recent research.

2.2.1 Subclustering and appearance classifiers

Munaro et al. proposes a people tracking algorithm for mobile robots using depth sensor information [7]. The proposed algoritm performs clustering on the scene for initial detection, then uses Kalman filter on the motion and appearance features to track multiple people within groups. The method assumes people are on the ground plane.

Initially, the method reduces the size of the Kinect depth point cloud through voxel grid filtering. The process divides the depth map into voxels, where the value of each voxel is the averaged depth value over an area. Then, depth clustering is used to segment the scene in to potential people with different heights. Sub clusters are created by finding the local maxima inside the entire cluster, where each sub-cluster is a bounding box enclosing a person's body. A HOG confidence is calculated for every sub-cluster, which will be compared against the HOG distribution when the person is occluded. The output will become the initial people detection result.

The tracking module leverages the AdaBoost, or Adaptive Boosting, learning algorithm to learn the color appearance of every tracked person, which is computed from the RGB histogram. The machine learning technique will improve the mode over time based on minimizing the errors in the previous models. The tracking module also estimates the motion for each person from their current position using an unscented Kalman Filter with a constant velocity model. Lastly, the system uses a people detector that helps the robot distinguish different people when they are close to each other using color information, as well as keeps it on tracked targets without moving towards obstacles when their colors are similar to those of the targets.

2.2.2 Point ensemble image (PEI), histogram of height difference (HOHD), joint histogram of color and height (JHCH)

Liu et a. proposes a number of new techniques for tracking in realtime with a single Kinect sensor [8], namely the point ensemble image (PEI), histogram of height difference (HOHD), and joint histogram of color and height (JHCH). The proposed method is divided into four stages stages. It transforms the RGB-D data into point ensemble images, then uses a detector to find positions of the human body, and finally a feature classifier that uses both histogram of height difference and joint histogram of color and height for fined-grained characterization of the human shape and apperance. Data association of the detection results over time with Kalman Filter is used to generate 3D trajectories for tracking.

A PEI representation combines the person's point cloud with height information. Firstly, the original point cloud of a person is transformed into the plan-view perspective, which is a 2D view of the model's depth data from the top-down perspective. A height map is generated from the point cloud in plan-view perspective. The height map color codes each cell with the highest point value. The final PEI image overlaps the original 3D point cloud with the height map.

In the second stage, the method detects human bodies from the generated PEI in two steps. Similarly to Munaro et al.ś appraoch, they find the local maxima points representing the head and draw a cylindrical boundary with radius $\omega/2$, where ω is the average width of the human torso. The points are selected with a constraint specifying the minimum and maximum height, thus filtering out a large amount of unfeasible points. Later on, the method tests whether the potential bodies are indeed humans using a shape and appearance classifier with data extracted from the neighboring points of the potential head position.

The features used in the classifier are HOHD and JHCH. HOHD leverages the information that the height of the head crown is larger than other points on the head and points around the shoulder. The height difference between the head and shoulder is less than a quarter of the average human height. JHCH analyzes the color and height distribution of the human head. The method examines the probability distribution of the skin and hair color with respect to height for neighboring points around the head whose height values are within the average length of the human head. Normally, the probability of the skin color occurrence decreases as the probability of the hair color increases.

The tracking algrorithm takes the surving points in PEI from the current frame as input and returns

the association result with the current tracks as output. The color similarity is computed from the JHCH, and the spatial position similarity is computed from difference between the actual position and predicted position using Kalman Filter. The algorithm labels detection results which have have no corresponding tracks as new tracked targets in the scene.

2.3 Coordinate transformation

[9] [10] [11]

CHAPTER 3

Objectives

3.1 Primary

3.2 Secondary

CHAPTER 4

Current Approach

This chapter identifies the main source of comparison in literature, then it proceeds to describe the tracking algorithm.

The current work is an extension of Wei et al.ś study [11]. It applies the same algorithms for doing calibration and coordinate transformation on the skeletal joints. The system will use the transformation technique to convert a skeleton's spatial position in one Kinect to another Kinect's field of view. Subsequently, the system can merge skeletons of the same person from different Kinects fields of view to achieve persistent tracking. The technology allows the system to continuously track people when the targets are obstructed by obstacles in the scene, as long as they remain in the extended field of view.

Both work uses only two Kinects, but the current work does a wider range of experiments. Wei et al. places the Kinects side by side. The current system is evaluated with Kinects that are not only adjacent to each other but further away from each other and have larger angle gaps. The former study also uses only one participant, whereas the current work uses up to two people in evaluation and can theoretically track up to six people, where the number of tracked person is limited by the maximum number of skeletons a single Kinect can recognize. Wei et al. uses simple movements to measure the performance of the tracking system. On the other hand, the current work introduces more complex movements. Occlusion is not discussed in Wei et al.ś study.

4.1 Running the application

KinectMultiTrack is available at `https://github.com/cjw-charleswu/KinectMultiTrack`. The application should run on the Microsoft Windows operating system. It is developed, tested, and maintained on Windows 8.

The application consists of a server and client. There must be at least one server and one client when running the system. It is recommended that there be at least two client. One machine can run both as the server and client.

Run the server:

contents

Run the client:

contents

4.2 Overview

The complete system architecture will be explained in section 5.3. In short, the clients send Kinect BodyFrames to the server. Later, the server process the data, performs calibration and tracking. Wei et al. designed a system where each individual machine running a Kinect will perform its own calibration then transmit the calibrated, transformed coordinates to the server [11]. The current approach puts the burden of processing on the server machine, and it requires less computational resources on the client side. Essentially, the clients are like Kinect hotspots sending skeletal information to the server. The tradeoffs between the two different approaches are not investigated.

Calibration allows the current system to precisely convert the skeletal joints coordinates from the original kinect's camera space into the new coordinate system, also known as the World coordinate system. After calibration, the system will combine multiple skeletons of each tracked person from different Kinects fields of view. This constitutes the initial tracking result. The system matches the skeletons based on their proximity of spatial positions in the World View, or the field of view of the World coordinate system.

When the server receives a new BodyFrame from the client, which is effectively a process running a Kinect sensor, it will update the spatial positions of the skeletons in the particular field of view. The person whose skeletons comprised of that skeleton will now have an updated position in the World View. The server handles multiple clients, hence multiple Kinects, in parallel to perform tracking.

The tracking process relies heavily on the initial result created after calibration. Any imperfect transformation process will produce errors, so as the current system. In this current context, error refers to the coordinates distances between multiple skeletons when converged from different Kinects fields of view into the same field of view. For instance, the system will try its best at merging skeletons of the same person from different Kinects fields of view into the same positions in the World View. Since the transformation process is not perfect, the underlying algorithm will produce two skeletons in the World View that have similar, but not exactly the same, positions. The system will minimize the skeletal joint differences by constructing an average skeleton.

The following chapter will explain all the steps leading to the transformed skeletons in a single viewing perspective.

4.3 Serialization

The Kinect BodyFrame is the wrapper around skeleton infromation at each frame. The same data structure was called SkeletonFrame in the Kinect v1 SDK. The BodyFrames are assembled by the Kinect sensor internally from its depth data; they provide a high level API for programming with skeletons. Each BodyFrame contains at most six Kinect Bodies, where each Body represents a person's skeleton from the Kinect's field of view. A Body encloses the skeletal joints coordinates and other related metadata. The current system uses this preprocessed information to track individuals in the scene.

The current Kinect v2 SDK does not support running multiple Kinects on a single machine. Therefore, the researcher writes a simple TCP server and client framework to pass Kinect BodyFrames from the clients to the server. TCP sockets deal with data in bytes, therefore the server and clients must exchange serialized data. Unfortunately, the Kinect v2 SDK does not support serialization of the Kinect BodyFrame. To resolve such inconvenience, the researcher also develops a Kinect BodyFrame serialization library. See Appendix 14.1 for a complete list of serialized data. The most important pieces of information are the skeletal joints and their tracking states, because the system requires people's spatial positions and Kinect's confidence level about those information. The system will give more weights to actively tracked skeletal joints when creating the average skeleton.

4.4 Calibration

The calibration procedure requies each skeleton's initial center position and angle between itself and the Kinect. The number of skeletons to be calibrated increases with the number of Kinects. For example, if there is only one person in the scene with two Kinects, there is a total of two skeletons, one from each of the Kinect's field of view, and so on.

Wei et al. defined the initial center position and angle as follows [11]:

Initial center position A skeleton's intial center position is its average of all joints coordinates over the duration of calibration. The center position is represented as $C(x_c, y_c, z_c)$, where x_c, y_c, and z_c are the x, y, and z coordinates, respectively.

Initial angle A skeleton's initial angle is the angle between the skeleton and Kinect in the last frame. The initial angle is represented as θ.

N denotes the number of joints per skeleton. N equals 25 in the current Kinect SDK. T denotes the number of frames used for calibration. T equals 120 in both the Wei et al. and the current study. $S(x_s, y_s, z_s)$ denotes the the sum of the joints coordinates in all calibration frames. $A(x_a, y_a, z_a)$ denotes the average of the joints coordinates in all calibration frames. $C(x_c, y_c, z_c)$ can be derived from the above information. The derviation is shown in equation. 4.1

$$\begin{aligned} S(x_s, y_s, z_s) &= \sum_{t=1}^{T}(\sum_{j=1}^{N}(x_{t,j}, y_{t,j}, z_{t,j})) \\ A(x_a, y_a, z_a) &= A(x_a, y_a, z_a)/N \\ C(x_c, y_c, z_c) &= C(x_c, y_c, z_c)/T \end{aligned} \tag{4.1}$$

The initial angle is defined as the the angle between the Kinect and the skeleton.

$$\begin{aligned} \text{Let } Z_r &= \text{Right shoulder z coordinate} \\ \text{Let } Z_l &= \text{Left shoulder z coordinate} \\ D &= Z_r - Z_l \end{aligned} \tag{4.2}$$

$$\begin{aligned} \text{Let } X_r &= \text{Right shoulder x coordinate} \\ \text{Let } X_l &= \text{Left shoulder x coordinate} \\ W &= X_r - X_l \end{aligned} \tag{4.3}$$

$$\theta = \arctan(D/W) \tag{4.4}$$

The system uses 120 Kinect BodyFrames for calibration. The Kinect provides BodyFrame at 30 frames per second, meaning the calibration process will take at least four seconds. The system will initiate the calibration process once it has received sufficient frames from all connected clients. If more frames were available from a connected Kinect, the system would use the latest 120 frames. The calibration procedure uses the coordinates in the Kinect Camera Space for all calculations.

4.4.1 Detecting interference

The system will automatically restart the calibration process if people move their joints over ten centimeters during calibration. The current implementation checks movements at the person's head, left hand, and right hand.

Algorithm 1 REMAINSTATIONARY(jt, c, p, msg)

Input:
jt: Joint type
c: Serialized body in the current frame
p: Serialized body in the previous frame
msg: Error message
Output:
Whether the person has moved the joint during calibration

```
1: if (c = null) ∨ (¬CONTAINS(c, jt)) ∨ (¬CONTAINS(p, jt)) then
2:     msg ← "Missing" + jt
3:     return false
4: c_jt ← Joint(c, jt)
5: p_jt ← Joint(p, jt)
6: d ← Distance(c_jt, p_jt)
7: if d > 0.1 then
8:     msg ← "" + jt + " remain stationary"
9:     return false
10: return true
```

4.4.2 Coordinate transformation

After the system completes calibration, it can transform any calibrated skeleton into the World coordinate system. Skeletal joints in the new coordinate system can also be transformed back to any Kinect's field of view. The transformation process is the very first step of filling the missing joints of a skeleton during occlusion. The average skeleton represents a person' complete joint coordinates, joined from multiple Kinects fields of view. Unsurprisingly, the accuracy of the system depends on how well the transformation algorithm is implemented.

Given the initial center position $C(X_C, Y_C, Z_C)$, initial angle θ, and the number of joints per skeleton, a skeletal joint in the World coordinate system can be expressed as follows:

$$J_t(X_{Jt}, Y_{Jt}, Z_{Jt}) = (X_j - X_C, Y_j - Y_C, Z_j - Z_C) \tag{4.5}$$

$$J_w(X_{Jw}, Y_{Jw}, Z_{Jw}) = (X_{Jt}\cos\theta + Z_{Jt}\sin\theta, Y_{Jt}, Z_{Jt}\cos\theta - X_{Jt}\sin\theta) \tag{4.6}$$

4.5 Tracking by detection

After calibration, all the system sees is a collection of skeletons with their initial position and angle. As aforementioned, the system can represent these skeletons in both the Kinect Camera Space and the world coordinate system. This information is not useful on its own, and the more people there are in the views of the Kinects, the larger this collection of skeletons would be. The tracking system should know which skeletons belong to which person, thus knowing about people's absolute position relative to any Kinect field of view.

The system construct models of tracked people by finding skeletons in different fields of view that have high proximity in the world coordinate system. The system performs the detection algorithm using joints in the world coordinate system, because using these coordinates would allow the system to compare skeletons' spatial positions regardless of perspectives. It assumes that skeletons from different Kinects field of view that have the highest proximity must belong to the same person. The system continues this process until it can no longer put skeletons in pair. The pseudocode is shown below **(todo: insert pseudocode)**.

The current implementation assumes that every person is visible to all Kinects. The system works fine

when there is only one person who is occluded from all Kinects, because the person would only have one skeleton, leaving it to the last to be matched. The system would not detect people correctly in scenarios where several people are occluded from all Kinects in the calibration phase, because it would try to match skeletons from different Kinect fields of view even though they are far apart from each other. **(todo: add illustrations of a working scenario and a non-working scenario)**.

Every calibration process follows a people detection result. It entails models of currently tracked people, where each model consists a number of potential skeletons, all from different Kinect perspectives. A potential skeleton represents one replica of a person from a Kinect field of view. The average skeleton of a person is the average body across all potential skeletons in the person's model. The average skeleton of a person can be seen as the system's view of that person. The result is permanent until the system recalibrates. That is, the same skeletons are always associated with the same person whom they were initially identified with.

The system performs tracking by updating every potential skeleton in the current result, propagating the changes to the skeleton visualization.

4.5.1 Detecting occlusion

When a person obstructs another person causing a Kinect sensor to partially or completely lose the sight of the masked person, the system would fill in the joints from other actively tracked potential skeletons. The average skeleton would be calculated using only the actively tracked joints from all potential skeletons. The effect would be visible on the application front-end; the visualization would display the latest average skeleton.

4.5.2 Detecting new and missing people

The system makes the assumption that people during calibration will be the only people in the scene throughout the lifetime of the system. This is because the system does not have any information about the new people entering the scene that would otherwise be obtained during calibration, such as their precise initial position and angle, which are needed to perform coordinate transformation. When the system detects intruders or zero people in the scene, it would automatically initiate calibration. **(todo: finish coding)** Scenarios where a number of skeletons, but not all, is missing from the system's available Kinects are unimportant, because the people possessing those skeletons may only be temporarily occluded.

Since the positions of all potential skeletons are updated every frame, the system would know when the skeletons are missing. The scene is empty if and only if every potential skeleton in the scene is empty.

CHAPTER 5

Design

5.1 Requirements

The application should be intuitive and easy to use. Since it is a prototype demonstrating the capability of the tracking system, it puts large emphasis on the skeleton visualization, showing that the combined skeletons match the expected outcome of the tracking process. The application displays the skeletons before and apply applying coordinate transformation and skeleton matching. The combined skeletons should render at the same speed as the server receives BodyFrames from multiple sources.

The application provides the end users essential functionalities for running the application, including to start and stop the server, recalibrate, view tracked skeletons from different views, and send the average skeleton stream to other applications. The logger should also store the tracking data on demand.

The researcher has discarded the following requriments for the software, due to time constraints and the scope of the project:

- The security of the application.
- The privacy of users' tracking information
- The scalability and robustness of the client and server.

5.2 Software stack

The current work uses Kinects for XBox One.

The system is written in C# with the 4.5 .NET framework, and the user interface is created using the .NET WPF framework. The choices are made because the official Microsoft Kinect SDK is in C# , and the latest examples use the WPF framework for the user interface.

5.3 System architecture

The main components of the system consist of a server, a tracker, a user interface, and a logger. The server passes on the Kinect BodyFrames received from the clients to the tracker. The tracker then processes the data and signals the user interface when the latest result is available. The user interface displays the tracking result on the skeleton visualization. When required by the end user, the logger would write tracking result to files.

The system topology consists of one or more machines in a client-server model. The latest Kinect v2 SDK at the time of writing (version 2.0.1410.19000) still does not support running multiple Kinects on a single machine, as a result, the system leverages the TCP/IP protocol for communicating between multiple Kinects. In the current system architecture, each client is running one Kinect (Figure 5.1). There is only one server, and any client machine can also run the server. All clients send Kinect Body frames to the server. The server is the workhorse of the system. It serves incoming client connections, establishes network streams with the clients, runs the user interface and exchanges information with the tracker (whom in runs the tracking algorithm), and lastly, informs the logger to write tracking data to files.

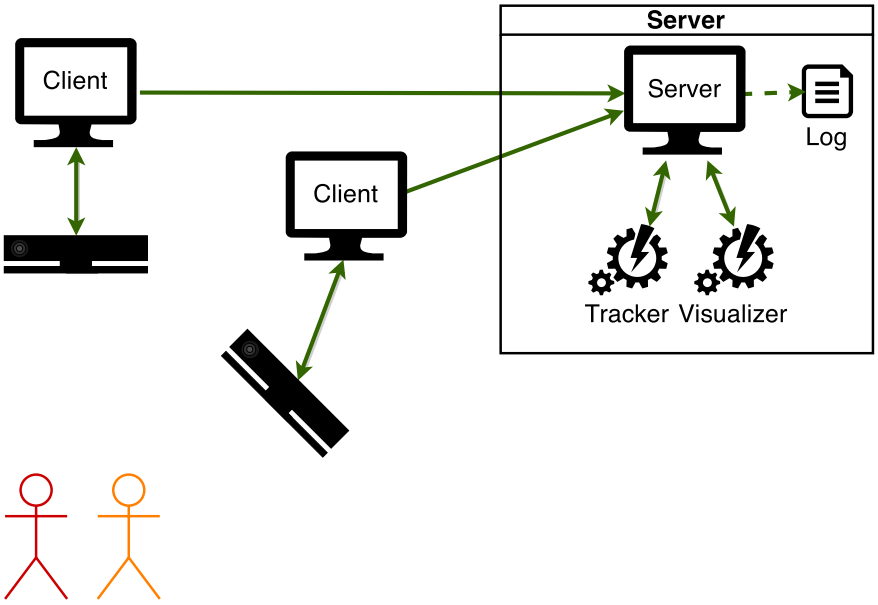


Figure 5.1: The system architecture

CHAPTER 6

Implementation

6.1 Client

The clients and servers communicate through TCP connections. The server opens the a port on the TPC network, and clients request connections to the server via sockets. When a client starts running, it will also start the Kinect **(change code: only start the Kinect after it's connected to the server)**. The client will continuously make connection requests until the server responds. If a connection is terminated by the server before the client stops running, the client will keep trying to reconnect to the server.

After the client establishes a connection with the server, it will start sending Kinect Body frames to the server. The low level networking is handled by the Microsoft .Net framework. The client serializes this data before transmitting it to the server, and the server will deserialize it. The server will then passes on the data to the tracker.

6.2 Server

The application leverages the C# events and delegates model. The application components subscribe to the event queues of other components. When new data is available from the subscribing component, the subscribed component consumes the data, does something with it, and fires events to all of its subscribed components. The components on the receiving end do the same, and so on. The server assembles the overall communication via events.

The application is started with one parameter, the server port number. It then creates a server to be run at that port number. The server will only start running when it receives such command from the user. After the server is created, it creates the user interface thread as a Single Threaded Apartment running in the background. The user interface will appear now.

The server will receive the following events from the user interface:

- Setup parameters for the server
- Start the server

- Stop the server
- When the user interface has displayed the tracking result (then the server will notify the logger)

The user will have control over the server, hence the system.

The server will pass the following events to the user interface:

- Clients (Kinects) have been connected to the server
- Clients (Kinects) have been removed from the server

The user interface can know which Kinects are connected to the server, giving the user feedback and later allowing him to choose from which Kinect perspective to view tracked people's skeletons.

The server will bind the following events from the tracker to the user interface:

- The tracker is waiting for Kinects to be connected
- The tracker is calibrating (and how many frames remaining)
- The tracker needs recalibration (and for what reason)
- The tracker has synchronized the latest BodyFrame with the tracking result

The user interface would show more feedback, including the latest result, from the tracker.

The application listens for TCP client connections. This work is done in a separate thread, called the ServerWorkerThread. When the TCP socket listener receives a new connection, the server will handle it in a new, separate thread. In the socket thread, the server will create a network stream between the client and the server. After the connection is established, the server will fire a "OnKinectConnected" event to the user interface. Later on, the server will receive Kinect BodyFrames from the client through the network stream it had created. Upon receiving some data, the server would deserialize it into a BodyFrame object, then it would fire another event called "Track" to the tracker with information about the sender and the BodyFrame itself. Lastly, the server will send a response (a string) back to the client. The response is trivial; it is used to tell the client that the data has been received. The client is also implemented so that it In the current implementation, the server returns "Okay". The server will continue to process additional BodyFrames received on its end of the network stream, and the above procedure repeats.

(todo: fix the server so that it can stop and report the additions)

6.3 User Interface

The application has one window. It has a number of small buttons as controls on the top of the interface. Below the controls the window is split into halves. The left hand side shows the visualization for merged skeletons after applying coordinate transformation, and the right hand side shows the visualization of skeletons in their original fields of view. Displaying the two different views at once demonstrates the system's tracking algorithm.

The design decisions on the look of the user interface are not discussed, because aesthetic features were not taken into consideration when the user interface was developed.

Available user controls are shown as buttons. The user interface responds to click events on each button. Buttons representing functionalities are that not meant to be used are disabled. For instance, when

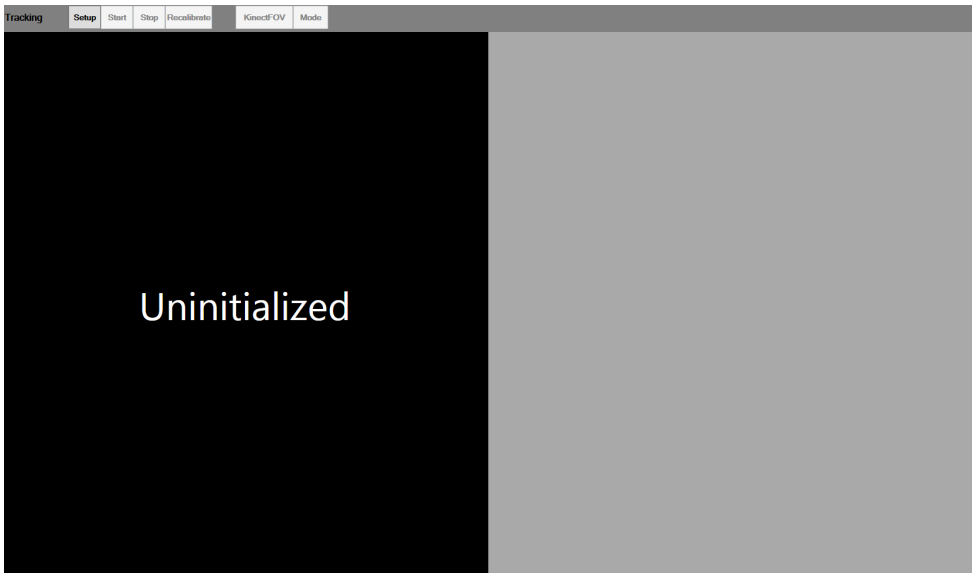


Figure 6.1: The user interface

the start button is pressed, signaling the server to start running, the setup button, which parameterized the server, should be disabled.

When the user interface receives events from the server about new and old connections with clients, the user interface adds and removes option to transform the kinect Bodies into the particular Kinect's field of view, respectively. The user interface would also display texts, centered on the left hand side visualization, about the progress of calibration or any interrupted action causing calibration to fail. How the user interface displays the BodyFrames is discussed in the next subsections "Tracking View" and "Disjoined View".

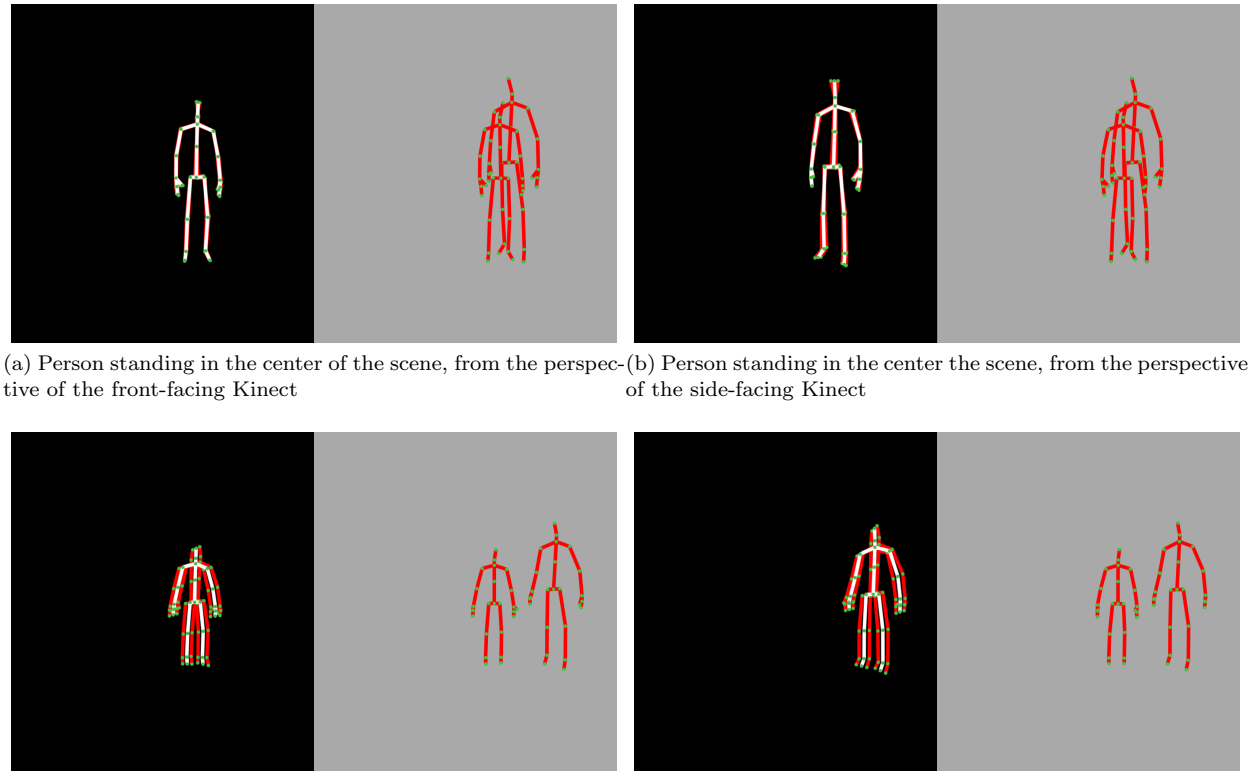
6.3.1 Tracking View

The Tracking View shows the skeleton visualization of the tracking result. The merged skeletons are drawn from the perspective of the selected Kinect, specified by the user or is defaulted to the local client's Kinect (The local client is the client that is also running on the server machine). The average skeleton is calculated at this stage. The potential skeletons of a person share the same color, and the average skeleton is always colored white. There are six available colors, because the system sets the cap of number of tracked people to six.

The skeleton visualization in both Tracking View and Disjoined View has the same implementation. The bones of the skeletons are drawn first, then the joints. The simplified list of human bones using the Kinect joints are taken from the Microsoft Kinect Developer examples. The inferred bones are displayed thinner than the tracked bones.

6.3.2 Disjoined View

The Disjoined View shows the skeleton visualization of the tracked skeletons in their original Kinect coordinate system. The skeletons are colored with respect to their Kinect origin. In other words, skeletons coming from the same Kinect would share the same color. The number of available colors is also limited to six.



(a) Person standing in the center of the scene, from the perspective of the front-facing Kinect

(b) Person standing in the center the scene, from the perspective of the side-facing Kinect

(c) Person standing at the back of the scene, from the perspective of the front-facing Kinect

(d) Person standing at the back of scene, from the perspective of the side-facing Kinect

Figure 6.2: The user interface showing both the tracking and disjoined views when the same person stands in different positions, from both the front-facing and side-facing Kinects' perspectives

6.4 Tracker

The tracker runs the tracking algorithm on the server. Much of the algorithm has been explained in section **??**. This section will explain how the algorithm has been implemented.

The tracker contains a dictionary of clients' IP address (as key) and a generic data structure (as value), called kinectClient, storing information about the Kinect connected to the client and all the frames received from it.

In calibration phase, all frames are stored in a stack inside each KinectClient. This allows the tracker to quickly find the latest 120 frames for calibration. Each KinectClient also keeps track of a list of active skeletons, or TrackingSkeletons. Throughout the lifetime of a tracking process, the tracker updates the position of the TrackingSkeletons.

6.5 Logger

The logger takes a tracking result and writes it to the file. The complete list of items logged at each time interval can be found in appendix **??**.

During the experiments, after the user interface displays the tracking result, it will fire an event to the server. The server will write every other tracking result to a file on the local disk. The user interface will not signal the server about the result if the experiment is paused between tasks.

The logger receives the joint coordinates in World coordinate system (as stored in the result), therefore, it converts them into coordinates in the local Kinect's camera space. The local Kinect is the one which

is connected to the server machine via a TCP client. The logger flushes the buffer after it completes the action.

CHAPTER 7

Testing

The system running multiple Kinects is verified by looking at the skeleton visualization on the user interface for a number of different users and in various interaction scenarios. The researcher is interested in whether the application has accomplished the following tasks:

1. The skeletons from different Kinect fields of view are matched correctly to the corresponding persons in the scene.
2. The transformed skeletons of the same person are close together, showing minimal differences in the joint coordinates.
3. The skeletons can be transformed to different Kinects' Camera Space (3D coordinate system) for viewing
4. All of the above statements hold when the users move freely.

7.1 Tracking

Curabitur porttitor quam ut ante condimentum, in sollicitudin urna pulvinar. Maecenas tincidunt sed enim ac posuere. Donec ligula odio, dignissim eu nisl at, placerat rutrum enim. Nullam tincidunt condimentum leo, sit amet tristique libero aliquam ac. Curabitur ornare elit tortor, non dapibus sapien sollicitudin sit amet. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam et lorem a risus cursus aliquet sit amet ut urna. Nunc sodales felis nec tortor efficitur venenatis. Curabitur et suscipit sem. Nunc rhoncus, ligula ut sagittis congue, ante dolor blandit ipsum, non accumsan elit augue at neque. Aenean volutpat sed turpis vel ultricies.

7.2 Occlusion

The main goal of the project is to show persistent tracking results in occluded environments and scenarios where complex human interactions are in play. The researcher has verified this requirement by partially and fully obstructing users in the scene. In the simplest case, a person may be self-occluded if he stands in a position such that one Kinect cannot fully see all the joints but two Kinects combined can have a

complete view of the person. **(todo: show an illustration)** The research stands back-facing the main Kinect, while showing his right arm only to the second Kinect. The system would form the average skeleton using the actively tracked information from both Kinects; the average skeleton would contain joint coordinates that both Kinects are most sure about.

CHAPTER 8

Studies

8.1 Motivation

A series of user studies are designed to evaluate the system's accuracy at tracking people in different scenarios. The accuracy of the tracking algorithm, or essentially the coordinate transformation algorithm, is measured by the differences in the joint coordinate between multiple potential skeletons of the same person. The studies will require participants to move around in front of multiple Kinects alone and with other participants. The software will log participants' positions from tracking, and these data will provide a quantitative feedback on the accuracy of the algorithm in different Kinect configurations and user scenarios.

To reiterate, a potential skeleton is a skeleton from a single Kinect field of view. One person may have multiple potential skeletons when they are visible to many Kinects. The application is most useful for its ability to transform any potential skeleton into any Kinect's camera space. The potential skeleton in the current Kinect field of view would be unaffected, but the other potential skeletons that were in other Kinects fields of view would have slight deviations in their joint coordinates. The user studies attempt to capture such deviations in all possible cases.

8.2 Hypotheses

The null hypotheses are:

1. The differences in each joint coordinate among all potential skeletons of a person are consistent across time and are within five centimeters
2. The differences in each joint coordinate among all potential skeletons of a person are consistent with different Kinect configurations
3. The application would fill in the missing joint coordinates of a person from information about all potential skeletons

8.3 Apparatus

The current studies use two machines and two Kinects. Each machine is connected to one Kinect and runs a client sending Kinect BodyFrames to the server. The server is running on one of the client machines.

The server machine is running Microsoft Windows 8 on a i5-3470S CPU at 2.90 GHz and 8 Gb RAM . The other client machine is also running Microsoft Windows 8, on a i7-3610QM CPU at 2.30 GHz and 8 GB RAM.

The sensors are the v2 Kinects for Xbox One. The SDK running those Kinects is version 2.0.1410.19000.

8.4 Participants

Participants are multinational university students and staff. There are participants in both genders and with a wide range of heights and weights. They are compensated with chocolates for participation.

8.5 Setting

The studies take place in a semi-controlled environment (See Figure 8.1). The two Kinects are placed at either three pre-defined locations, where they are approximately parallel, 45 and 90 degrees apart. One Kinect is always placed at the front position; it is the Kinect on the left in the image. ((**todo: measure the exact angles and distances between them - they are marked by duct tapes so shouldn't be too hard**)) Duct tapes are used to mark the precise locations of the Kinects. The boundary within which the participants will be moving is also marked with duct tapes. The sides of the block are found empirically to be near the minimum distance of the Kinect viewing range of the full body skeleton. The dimension of the boundary is x by x meters ((**todo: find the dimensions in meters**)). Each potential step is marked with duct tapes colored in a hue (either black or red) different from that on the duct tapes in the previous step. The starting position has a distinct color (green) from all other steps.

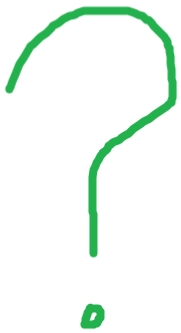


Figure 8.1: The setting where the user studies took place

8.6 Method

Firstly, participants are introduced to the project aims and objectives. They are given sufficient time to ask questions and decide whether to participate in the experiment before signing the consent form. Participants are free to withdraw from the studies at any time without any explanation. Secondly, participants are told beforehand what instructions they would expect during the experiments. This is because the studies are designed to measure how well the system tracks people, not how participants react to some situations. In user studies mode, the application would show instructions on the right hand side of the screen, telling the participants where to put their feet next. For instance, it will tell the participant to go around the obstacle (See Figure 8.2). The application will try to log as least amount of stationary movements as possible for tasks where they require participants to be moving. Not only because testing for differences in joint coordinates when the participant remains stationary is a standalone study, but also the researcher is interested in how the tracking algorithm performs when tracked people are constantly moving. To achieve this goal, the application introduces pauses between tasks. The researcher has control over the starting time of the next task. During the pauses, the researcher would give additional details about the studies to the participants.

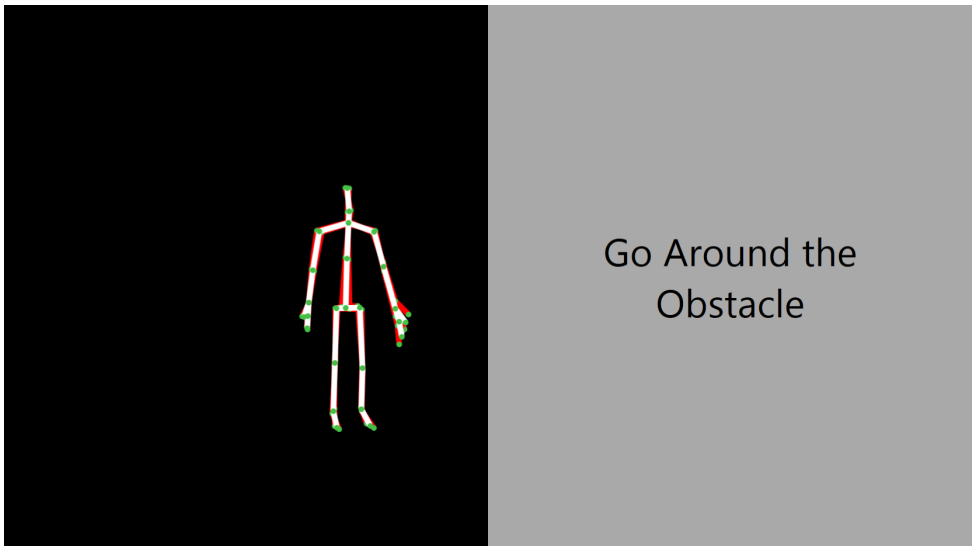


Figure 8.2: The instruction in the Obstacle task.

8.7 Ethics

There are no legitimate ethical concerns about participating in the studies. The skeleton data are anonymized and will be stored up to a maximum of three years. Any participant who feels uncomfortable with the guideline is welcome to speak to the researcher and his supervisor.

8.8 Study 1: Stationary

In the first study, participants are required to remain stationary for ten seconds in the center of the block. The study is done with all three Kinect configurations (Parallel, 45 degrees-apart and 90 degrees-apart)

8.9 Study 2: Steps (Basic movements)

The second study requires the participants to move in the same way as explained in the Wei et al study. These are basic movements such as moving forward, backward, left, and right. The study is done with all three Kinect configurations.

8.10 Study 3: Walk (Continuous Movements)

The third study requires the participants to walk around the perimeter of the block and walk diagonally to each of four corners. Like the previous two studies, study 3 is done with all three Kinect configurations. Studies 1, 2, 3 are conducted in succession for every participant.

8.11 Study 4: Obstacle

Participants are asked to walk around a large obstacle, which is a large poster in the current study. The obstacle divides the field of view of two Kinects at 90 degrees apart (See Figure 8.3). The participant starts on the right hand side of the obstacle, where he is visible to both Kinects. As the participant walks around the obstacle, from the back, then to the left side of the obstacle, the Kinect that was looking at the side of the participant will slowly lose the sight of the person. When the participant is on the other side of the obstacle, only the front-facing Kinect will have sight of the person. The study should demonstrate that the system would still be able to track the person despite that one of the Kinect loses the person's sight temporarily. The study is only done with Kinects placed 90 degrees apart.

Figure 8.3: A ITS 2013 poster used as an obstruction in the Obstacle task.

8.12 Study 5: Interaction

The interaction study involves two people. They stand next to each other. The person on the left will walk to the front of the other person, then back to his initial position. Then he will walk around the person, from the front to the back, then return to his starting position. The other person does the same. In the end, both people exchange positions.

CHAPTER 9

Results

This chapter summarizes the results in each user study and makes comparisons between a number of different studies. The results should provide insights into the reliability and accuracy of the tracking algorithm in different settings. In additiona, it would show the coordinates and joints that yield the most stable results during normal human activities.

9.1 Accessing the data

The complete dataset and plots are publicly available at `https://github.com/cjw-charleswu/KinectMultiTrackDataset`

9.2 Analysis

The analysis was done in Matlab 2015a. The analysis scripts are available at `https://github.com/cjw-charleswu/KinectMultiTrack/tree/master/Analysis`.

9.2.1 Cleaning the data

The logging data are post-processed for ease of plotting the results. The initial logging data, for all aforementioned studies, contain 244,527 rows and 87 columns in total. The final data for evaluation contain 243,550 rows and 87 columns. 977 rows are deleted for various reasons documented below.

There is also an error in code where a scenario id is logged incorrectly. In the second part of scenario 8, when the second participant is asked to walk around the first participant, the scenario id is falsely written as 4. This logging error is corrected by replacing all occurrences of scenario id 4 that are immediately after scenario id 8 and before scenario id 5, which is the next task in line for the participants.

The tracker time is stored as the server's current time in milliseconds. The times for each user task (scenario) are reset. The current timestamps refer to the amount of time passed in each scenario, for a particular Kinect configuration and experiment.

The timesalso converted from milliseconds to seconds. The joint coordinates are converted from meters to centimeters.

The studies are interested in the amount of coordinate differences between multiple skeletons after transformation during tracking. Thus, evaluation requires the joint positions of more than one skeletons (from different Kinect fields of view) at any given timestamp.

9.3 Definitions

Δx, Δy, Δz, Δd, Avg., and Std. are defined as:

Δx The distance, or difference, between the x coordinates of a joint of multiple skeletons representing the same person from different Kinects fields of view, expressed in centimeters.

Δy The distance, or difference, between the y coordinates of a joint of multiple skeletons representing the same person from different Kinects fields of view, expressed in centimeters.

Δz The distance, or difference, between the z coordinates of a joint of multiple skeletons representing the same person from different Kinects fields of view, expressed in centimeters.

Δd The distance, or difference, between the x, y, and z coordinates of a joint of multiple skeletons representing the same person from different Kinects fields of view, expressed in centimeters.

Avg. Average (mean)

Std. Standard deviation

These values quantify the amount of differences produced by the tracking algorithm when transforming multiple skeletons of the same person to a single Kinect field of view.

Coordinates and joints distances are defined as:

Coordinates distances The Δx, Δy, Δz, and Δd distances averaged over a person's entire set of joints, expressed in centimeters.

Joints distances The Δx, Δy, Δz, and Δd distances for each of a person's joints, expressed in centimeters.

9.4 Structure

There are three main results sections, one for each of the primary tasks, namely the Stationary, Steps, and Walk tasks. Each of these sections contains three basic visualizations showing the effects of the particular scenario on coordinates transformation errors when tracked with Kinects at different locations.

Each main section will begin with a figure showing changes in the average coordinates and joints distances with Parallel, 45° and 90° apart Kinects. The x asis will be the position of Kinect placement. The y axis will be the average coordinates distances, in terms of Δx, Δy, Δz, and Δd (as defined in section 9.3). The figures will also entail the average coordinates distances over all of the Kinect placements.

The second figure will give more details about the average case shown in the previous figure. It will show changes in the average coordinates distances for each of the joint types. The x asis will be the joint type. The y axis will be the average joints distances, also in terms of Δx, Δy, Δz, and Δd.

Then, each section will finish with a table summarizing the actual values of the average coordinates distances for the presented task with each of the Kinect placements, as well as averaged over all different

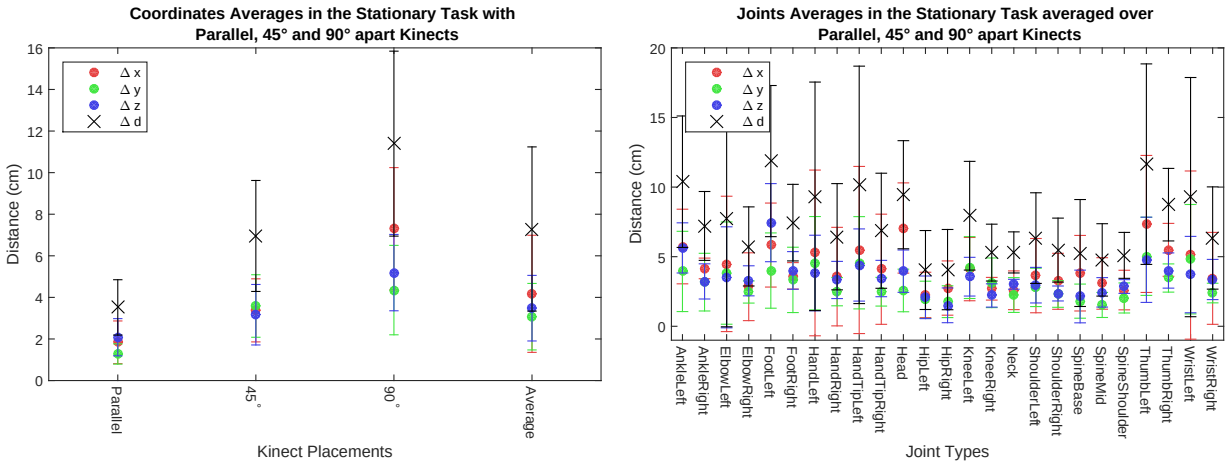
placements. The figures show how Kinect placements affect the average coordinates and joints distances during the Steps task.

At last, the results obtained from different scenarios and Kinect placements are compared. Figures will show the average coordinates and joints differences over time.

9.5 Stationary

This section reports average coordinates and joints distances in the Stationary task with parallel, 45° and 90° apart Kinects.

The Δd distance in the Stationary task with parallel Kinects is 3.52 cm. The Δd distance in the Stationary task with 45° apart Kinects is 3.95 cm. The Δd distance in the Stationary task with 90° apart Kinects is 11.39 cm. The difference between the best and worst cases is 7.87 cm. The average Δd distance in the Stationary task over all Kinect placements is 7.9,cm, with a standard deviation of 3.95 cm.



(a) Average coordinates distances in the Stationary task with Parallel, 45° and 90° apart Kinects.

(b) Average joints distances in the Stationary task averaged over Parallel, 45° and 90° apart Kinects.

Figure 9.1: Plots showing the results in the Stationary task with different Kinect placements.

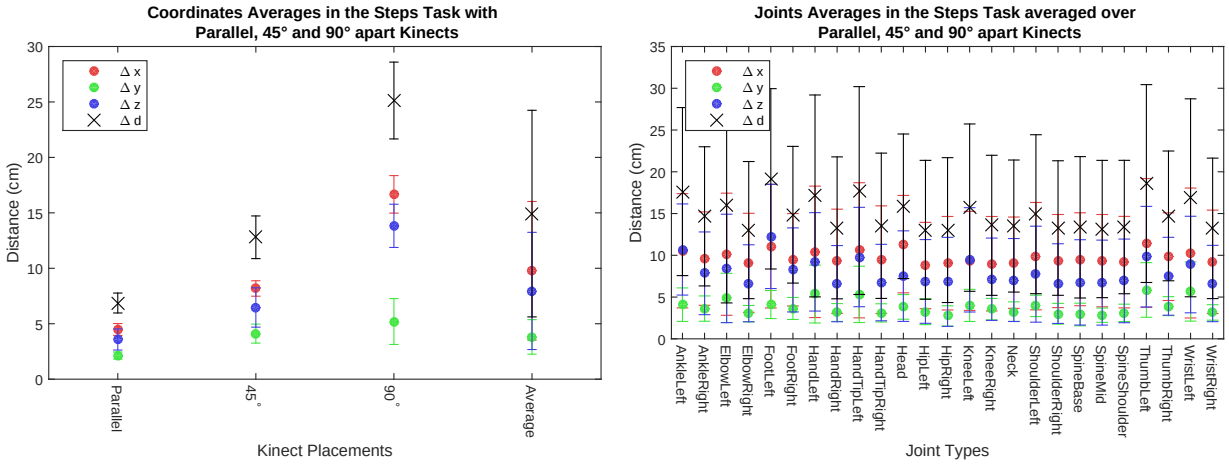
Distances	Parallel	45°	90°	Average
Avg. Δx	1.84	3.38	7.30	4.17
Std. Δx	1.03	1.52	2.94	2.82
Avg. Δy	1.28	3.59	4.35	3.07
Std. Δy	0.49	1.50	2.15	1.60
Avg. Δz	2.08	3.17	5.1917	3.48
Std. Δz	0.89	1.45	1.84	1.58
Avg. Δd	3.52	6.95	11.39	7.29
Std. Δd	1.33	2.67	4.45	3.95

Table 9.1: Average coordinates distances in the Stationary task with Parallel, 45° and 90° Kinects, as well as the average case. The means and standard deviations for Δx, Δy, Δz, and Δd are reported.

9.6 Steps

This section reports average coordinates and joints distances in the Steps task with parallel, 45° and 90° apart Kinects.

The Δd distance in the Steps task with parallel Kinects is 6.87. The Δd distance in the Steps task with 45° apart Kinects is 12.80. The Δd distance in the Steps task with 90° apart Kinects is 25.13. The average Δd distance in the Steps task over all Kinect placements is 14.93, with a standard deviation of 9.32.



(a) Average coordinates distances in the Steps task with Parallel, 45° and 90° apart Kinects.

(b) Average joints distances in the Steps task averaged over Parallel, 45° and 90° apart Kinects.

Figure 9.2: Plots showing the results in the Steps task with different Kinect placements.

Distances	Parallel	45°	90°	Average
Avg. Δx	4.48	8.18	16.7	9.78
Std. Δx	0.53	0.70	1.70	6.25
Avg. Δy	2.13	4.11	5.20	3.81
Std. Δy	0.32	0.86	2.07	1.56
Avg. Δz	3.58	6.47	13.83	7.96
Std. Δz	0.95	1.77	1.95	5.28
Avg. Δd	6.87	12.80	25.13	14.93
Std. Δd	0.90	1.92	3.46	9.32

Table 9.2: Average coordinates distances in the Steps task with Parallel, 45° and 90° Kinects, as well as the average case. The means and standard deviations for Δx, Δy, Δz, and Δd are reported.

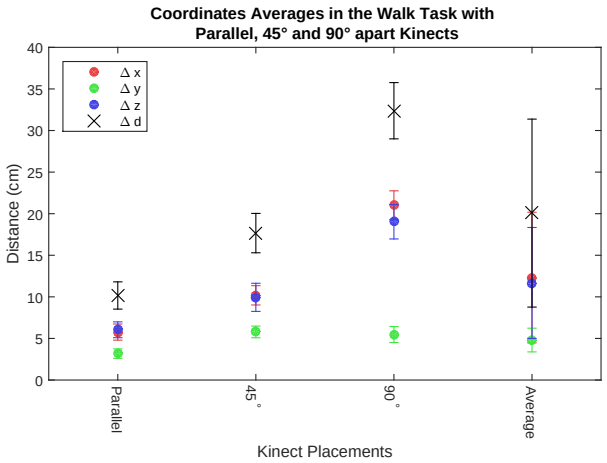
9.7 Walk

This section reports average coordinates and joints distances in the Walk task with parallel, 45° and 90° apart Kinects.

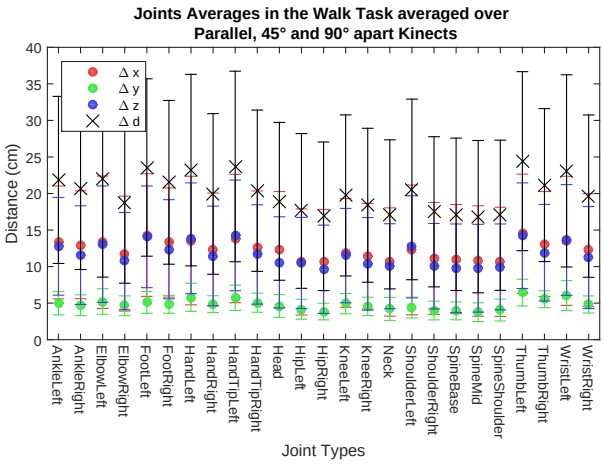
The Δd distance in the Walk task with parallel Kinects is 10.17. The Δd distance in the Walk task with 45° apart Kinects is 17.67. The Δd distance in the Walk task with 90° apart Kinects is 32.38. The average Δd distance in the Walk task over all Kinect placements is 20.07, with a standard deviation of 11.30.

9.8 Stationary, Steps, Walk

This section summarizes the results in the Stationary, Steps, and Walk tasks with Parallel, 45° and 90° apart Kinects.



(a) Average coordinates distances in the Walk task with Parallel, 45° and 90° apart Kinects.



(b) Average joints distances in the Walk task with Parallel, 45° and 90° apart Kinects.

Figure 9.3: Plots showing the results in the Walk task with different Kinect placements.

Distances	**Parallel**	**45°**	**90°**	**Average**
Avg. Δx	5.76	10.18	21.02	12.32
Std. Δx	0.97	1.16	1.73	7.85
Avg. Δy	3.17	5.78	5.47	4.81
Std. Δy	0.57	0.70	0.96	1.42
Avg. Δz	6.04	9.94	19.03	11.67
Std. Δz	0.95	1.69	2.07	6.67
Avg. Δd	10.17	17.67	32.38	20.07
Std. Δd	1.64	2.37	3.38	11.30

Table 9.3: Average coordinates distances in the Walk task with Parallel, 45° and 90° Kinects, as well as the average case. The means and standard deviations for Δx, Δy, Δz, and Δd are reported.

Figure 9.4 shows two different plots. Firstly, it shows the average coordinates and joints distances in the Stationary, Steps, and Walk tasks averaged over different Kinect placements (Parallel, 45° and 90° apart Kinects). The reverse is also shown. The figure also shows the average coordinates and joints distances with Parallel, 45° and 90° apart Kinects averaged over different tasks (Stationary, , Steps, and Walk). The figure shows how the complexity of tasks and the placement of the Kinects, respectively, affect the accuracy of the tracking algorithm.

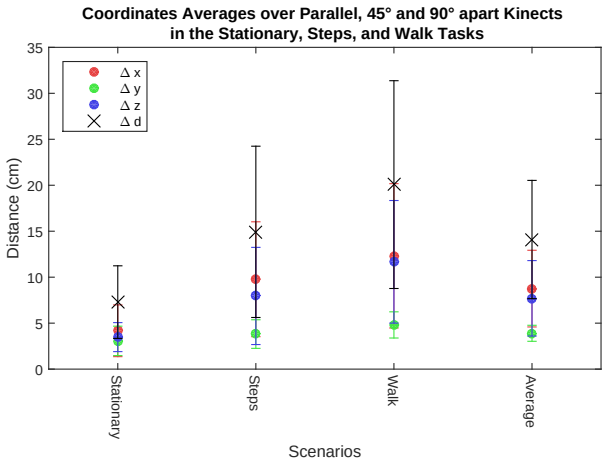
When the Kinect placements are averaged, the best case is 7.29 cm (Stationary), the worst case is 20.07 cm (Walk), and the average case is 14.10 cm.

When the tasks are averaged, the best case is 6.85 cm (Parallel), the worst case is 22.97 cm (90°), and the average case is 10.57 cm.

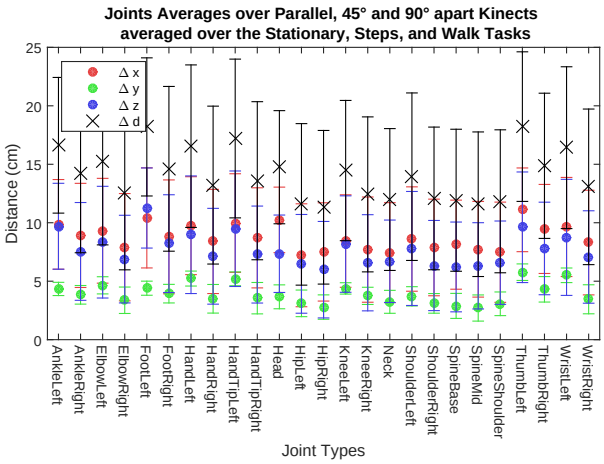
Figure 9.5 shows the average joints distances over time in the Stationary, Steps, and Walk tasks with Parallel, 45° and 90° apart Kinects. These figures demonstrate the stability of the tracking algorithm for different joints over time when performing different tasks in different Kinect placements. The figures are taken from participant 18.

9.9 Obstacle

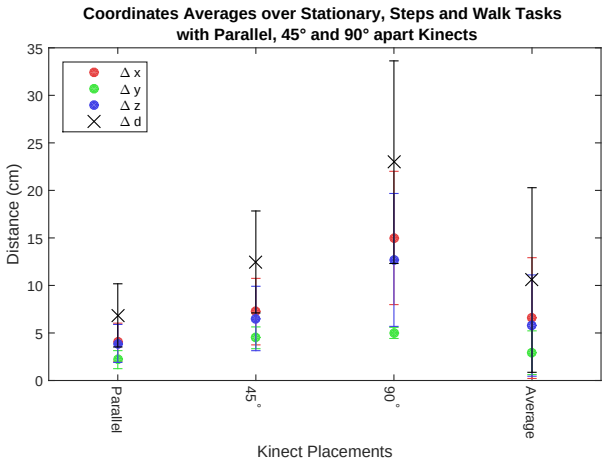
TODO



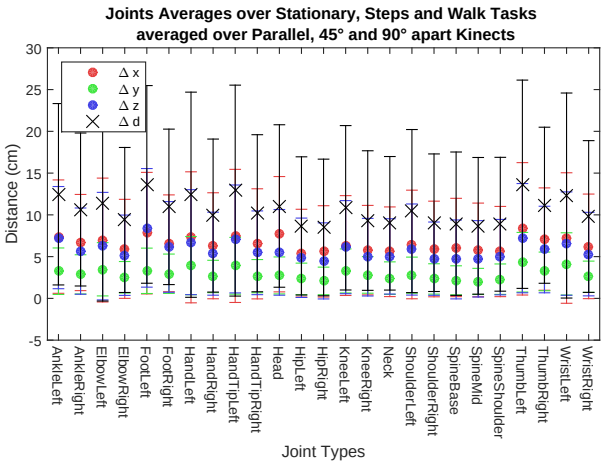
(a) Average coordinates distances in the Stationary, Steps, and Walk tasks averaged over Parallel, 45° and 90° apart Kinects.



(b) Average joints distances in the Stationary, Steps, and Walk tasks averaged over Parallel, 45° and 90° apart Kinects.



(c) Average coordinates distances with Parallel, 45° and 90° apart Kinects averaged over Stationary, Steps, and Walk tasks.



(d) Average joints distances with Parallel, 45° and 90° apart Kinects averaged over Stationary, Steps, and Walk tasks.

Figure 9.4: Plots showing the overall results in the Stationary, Steps, and Walk tasks with different Kinect placements.

9.10 Interactions

TODO

9.11 Overall

Distances	Stationary	Steps	Walk	Average
Avg. Δx	4.17	9.78	12.32	8.76
Std. Δx	2.82	6.25	7.85	4.17
Avg. Δy	3.07	3.81	4.81	3.90
Std. Δy	1.60	1.56	1.42	0.87
Avg. Δz	3.48	7.96	11.67	7.70
Std. Δz	1.57	5.28	6.67	4.10
Avg. Δd	7.29	14.93	20.07	14.10
Std. Δd	3.95	9.32	11.30	6.43

(a) caption

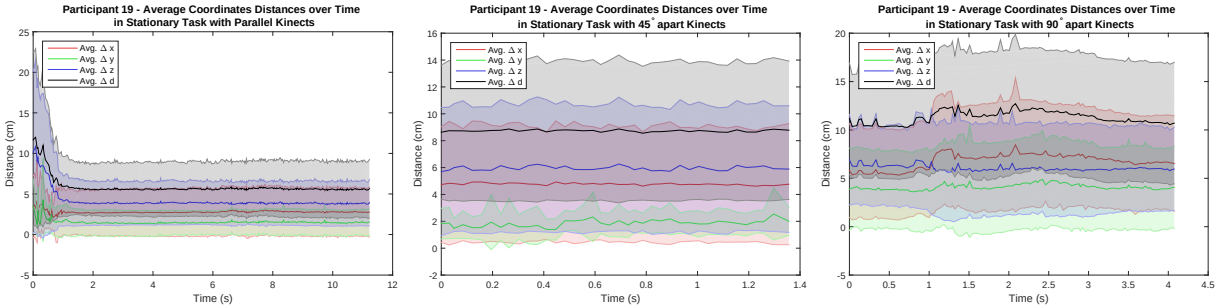
Distances	Parallel	45°	90°	Average
Avg. Δx	4.03	7.25	15.00	6.57
Std. Δx	2.00	3.50	7.00	6.35
Avg. Δy	2.19	4.49	5.01	2.92
Std. Δy	0.95	1.15	0.58	2.30
Avg. Δz	3.90	6.53	12.68	5.78
Std. Δz	2.00	3.39	6.99	5.33
Avg. Δd	6.85	12.47	22.97	10.57
Std. Δd	3.32	5.36	10.66	9.71

(b) caption

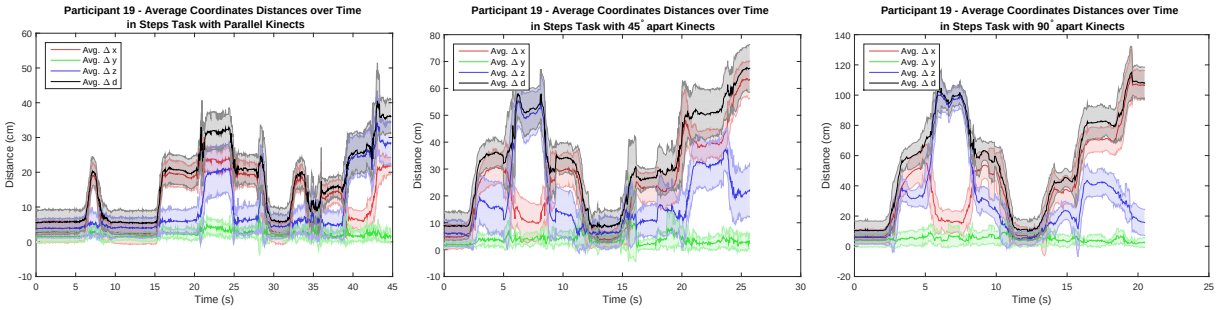
Table 9.4: Table showing coordinates distances in the Walk task with Parallel, 45° and 90° Kinects, as well as the average case. The means and standard deviations for Δx, Δy, Δz, and Δd are reported.

Setup	Avg. Δx	Avg. Δy	Avg. Δz	Avg. Δd
Parallel, Stationery	1.84	3.38	7.30	4.17
Parallel, Steps	0.49	1.50	2.15	1.60
Parallel, Walk	0.49	1.50	2.15	1.60
45°, Stationery	1.03	1.52	2.94	2.82
45°, Steps	0.89	1.45	1.84	1.58
45°, Walk	0.89	1.45	1.84	1.58
45°, Interaction	0.89	1.45	1.84	1.58
90°, Stationery	1.28	3.59	4.35	3.07
90°, Steps	3.52	6.95	11.39	7.29
90°, Walk	3.52	6.95	11.39	7.29
90°, Obstacle	1.33	2.67	4.45	3.95

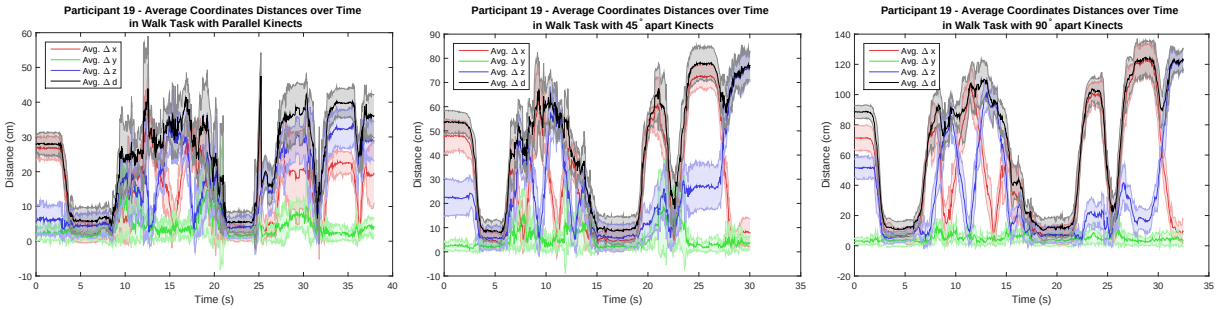
Table 9.5: Table showing the overall average coordinates distances



(a) Stationary

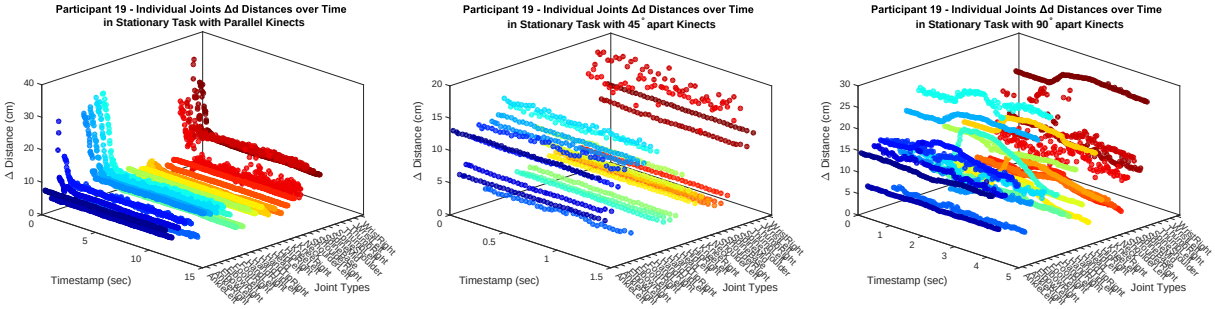


(b) Steps

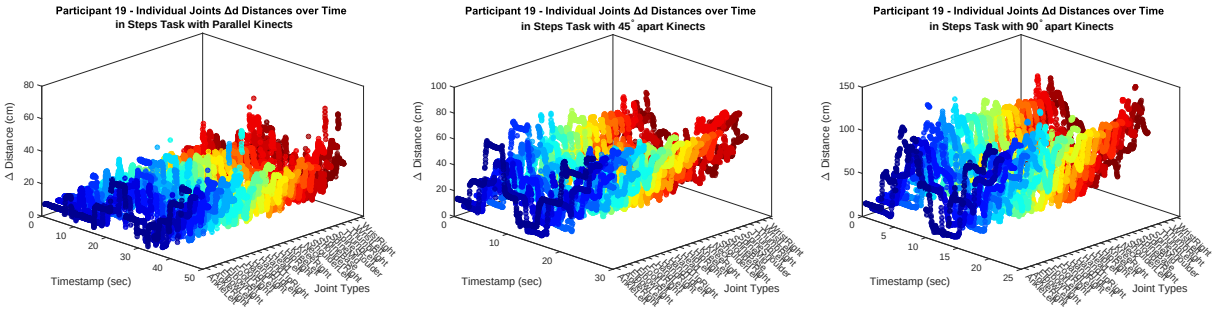


(c) Walk

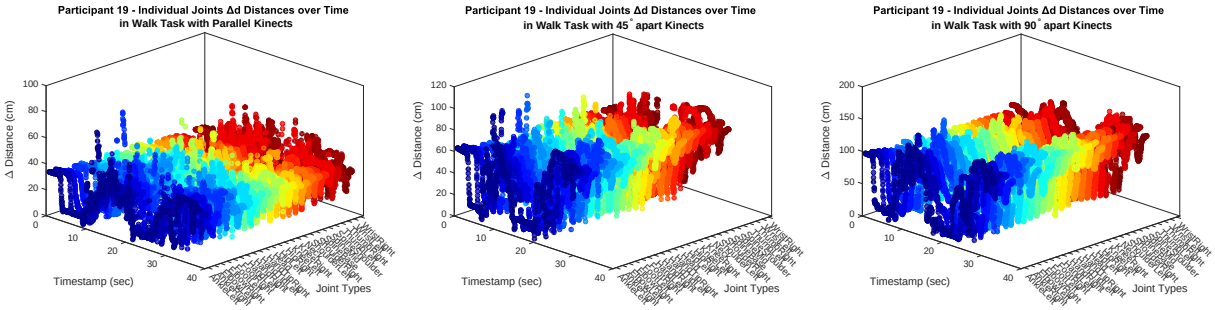
Figure 9.5: Plots showing average coordinates distances over time in the Stationary, Steps, and Walk tasks with Parallel, 45°, and 90° apart Kinects.



(a) Stationary



(b) Steps



(c) Walk

Figure 9.6: Plots showing average joints distances over time in the Stationary, Steps, and Walk tasks with Parallel, 45°, and 90° apart Kinects.

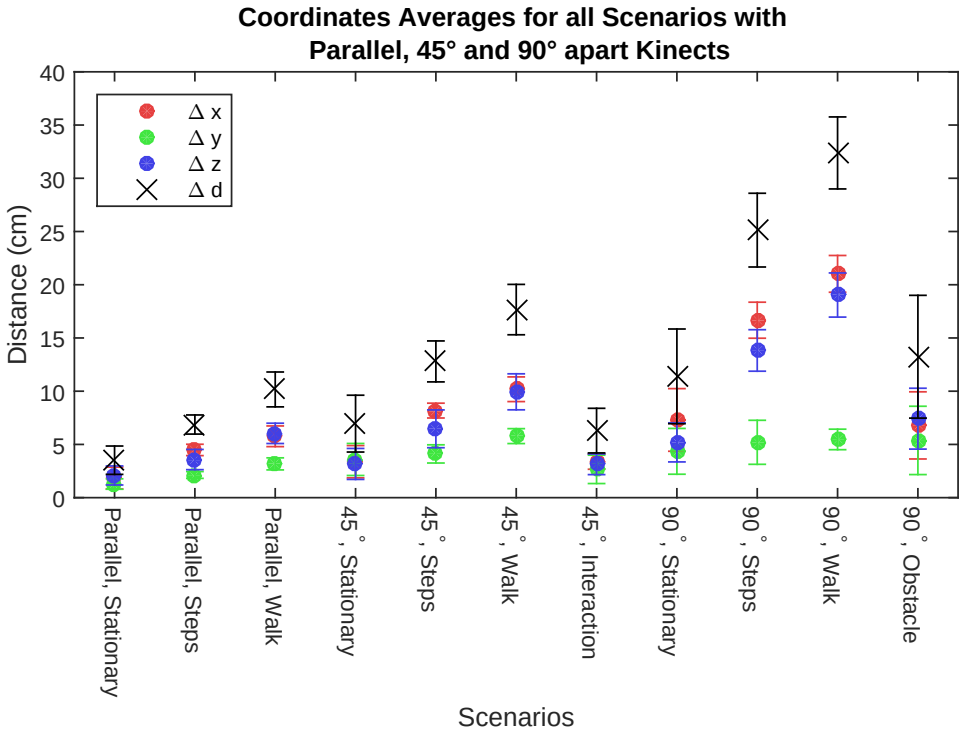


Figure 9.7: Plot showing the overall results in all scenarios

CHAPTER 10

Discussion

This chapter discusses the results stated in chapter 9, compare and contrast them with the Wei et al. study [11]. For each of the primary tasks, namely Stationary, Steps, and Walk, the discussion will consist of how the average coordinates and joints distances change with different Kinect placements. The spread of distances values over different joint types in each task will also be discussed.

It is worth noting that Wei et al only studied Stationary and Steps tasks, with near parallel and 45° apart Kinects. Tthe current work will compare results with those in Wei et al's study where appropriate [11].

10.1 Stationary

This section discusses results in the Stationary task.

The Stationary task shows best results when the Kinects are parallel to each other and worst when they are 90° apart. All measures of distances follow the same trend, from Δx to Δz (See Figure 9.1(a)). The results show that coordinates distances in the Stationary task increases with increasing angle between Kinects.

In general, the Δx values are the highest, then Δz and Δy. This shows that the tracking algorithm makes the largest errors in the coordinates transformation of the x axis and least errors in the y axis. A feasible explanation would be that the heights of both Kinects are the same, and the participants are not moving on the y axis.

The standard deviations of Δd in the average case is within the range of standard deviations across different Kinect placements. This shows that in the Stationary task the coordinates distances are consistent both within and between different Kinect placements.

The average joints distances are smallest for joints in the torso, namely the head, hips, knees, neck, shoulders, and spine (See Figure 9.1(b)). These joints appear to have similar coordinates distances. On the contrary, joints in the arm and foot regions have higher coordinates distances. The researcher speculates that the results are influenced by Kinect's internal performance; it produces more reliable and consistent results in the torso compared to other regions of the body.

All average coordinates distances are also higher for the left joints compared to right joints. In

summary, right hand side joints in the torso have the smallest coordinates disances, whereas left hand side joints in the arm and foot regions have the highest coordinates distances. The researcher also speculates that these results are due to the Kinect itself, which produces higher quality depth map in the torso.

Wei et al reported lower values compared to the current study [11]. In their Stationary task (Average Difference before Movement) with Parallel (4.25°) apart Kinects, the coordinates distances are 0.00, 1.00, and 2.00 cm for Δx, Δy, and Δz, respectively. The Δd will be 2.24 cm, which is lower than the 3.52 cm (See Table 9.1) found in the current study. In their same task with 45° (44.37°) apart Kinects, the coordinates distances are 1.00, 1.00, and 1.50 cm for Δx, Δy, and Δz, respectively. The Δd will be 2.06 cm, also lower than the 6.95 cm found in the current study.

10.2 Steps

This section discusses results in the Steps task.

The coordinates distances in the Steps task are higher compared to those in the Stationary task for every type of Kinect placements. The increase in coordinates distances is anticipated, because the task requires the participants to walk around in the environment, causing them to move closer or further away from the cameras. The movements will lead to increased liklihood of the tracking algorithm making more errors compared to when the participants are standing still. The following section will comment and explain the similarities and differences in the results between the Steps and Stationary tasks.

The Steps task also shows best results when the Kinects are parallel to each other and worst when they are 90° apart. All measures of distances follow the same trend, from Δx to Δz (See Figure 9.2(a)). Alike to the results in the Stationary task, the studies for the Steps task show that coordinates distances in also increase with increasing angle between Kinects.

For all Kinect placements, the Δx values are the highest, then Δz and Δy. The results support the findings in the Stationary task. This shows that the tracking algorithm produces consistent coordinates transformation across different tasks.

The standard deviation of Δd within each type of Kinect placement is lower than the same setup in the Stationary task. The standard deviations of Δd in the Stationary task are 1.33, 2.67, 4.45, and 3.95 cm, for Parallel, 45°, and 90° apart Kinects, respectively. On the other hand, the Steps task yields 0.90, 1.92, 3.46, and 9.32 for the same measures. These values suggest that the algorithm produces less variations within different Kinect configurations in the Steps task compared to the Stationary task. However, the Steps task has a higher standard deviation of Δd in the average case, compared to that of in the Stationary task. Even though the system shows smaller variations within different Kinect placements in the Steps task, it produces large variations between different conditions of Kinect placements. The system copes better to changes in Kinect placements in the Stationary task than in the Steps task.

There is a less visible difference in the coordinates distances between joints in the torso and other body regions (See Figure 9.2(b)), but it remains when examined closely. However, the pattern that left joints have higher coordinates distances is still very noticeable. Overall, the variations of the distances in each joint are still large, except for the Δy distances. The Δy distances and their standard deviations reamin small.

Wei et al also reported lower results compared to the current study. In their Steps task (Average Difference after Movement) with Parallel (4.25° apart) Kinects, the coordinates distances are 2.00, 1.28, and 3.78 cm for Δx, Δy, and Δz, respectively. The Δd will equal to 4.46 cm, which is lower than the 6.87 cm (See Table 9.2) found in the current study. In their same task with 45° (44.37°) apart Kinects, the coordinates distances are 4.28, 1.64, and 5.28 cm for Δx, Δy, and Δz, respectively. The Δd will equal to 6.99 cm, also lower than the 12.80 cm found in the current study.

10.3 Walk

This section discusses results in the Walk task.

The coordinates distances in the Walk task are higher compared to those in the Stationary and Steps tasks for every type of Kinect placements. The increase in coordinates distances in the Walk task has the same cause as previously mentioned in section 10.2 for the Steps task. Becuase walking movements are even larger than the stepping and stationary movements, the results in the Walk task will be higher compared to the other two tasks.

Similar to the previous two tasks, the results in the Walk task show smaller coordinates distances with parallel Kinects, and increasing distances with larger angles (See Figure 9.3(a)). Likewise, the Δx values are still the highest, follwed by Δz and Δy. Δy is still nearly invariant to changes in Kinect placement (See Figure 9.3(b)). Its standard deviation is the lowest compared to the standard deviations of all other distances measures, in all the tasks seen so far (Stationary, Steps, and Walk) with all different Kinect placements (Parallel, 45°, and 90° apart Kinects).

The average and standard deviation of Δy is the only value that hardly changes across different tasks and Kinect placements. In the Stationary task, the average Δy is 3.07 cm, with a standard deviation of 1.60 cm. In the Walk task, the average Δy only increases to 4.81 cm, with a standard deviation of 1.42 cm. The average Δy in the Steps task is in between these values. The results can be found in Table 9.1, Table 9.2, and Table 9.3. These support the argument that Δy is steady throughout the tracking process, regardless of tasks and Kinect placements.

In both the Steps and Walk tasks, the standard deviation of coordinates distances within each Kinect placement is small compared to the same value in the average case. The same relationship is not observable in the Stationary task. The researcher argues that increasing angle between multiple Kinects have a larger impact in non-stationary tasks.

There are common patterns in the joints distances between different tasks. Firstly, all three tasks show higher distances values for the left joints. Furthermore, there is a difference between joints in the torso and other boy regions, to varying degrees across tasks. In additional, all three tasks show that Δx and Δz are closer to each other than to Δy.

Wei et al did not run experiments containing longer, continuous walking tasks. The researcher has not found results for a similar task in the literature.

10.4 Stationary, Steps, Walk

The tasks Stationary, Steps, and Walk are ordered in increasing complexity, where the former requires zero movements, and the latter requires constant movements. The studies show that coordinates distances increase with increasing complexity of the tasks (See Figure 9.4(a)). The relationship can be attributed to increasing amount of joints movements and turning of the shoulder, of which the coordinates transformation uses for calculating the angle between the Kinects and the person. In addition, when averaged over different Kinect placements, the coordinates distances for different joints in Stationary, Steps, and Walk tasks are roughly the same, but there is a distinct difference between coordinates of the left and right joints (See Figure 9.4(b)).

The Kinect placements of parallel, 45° and 90° are ordered in increasing angle. The studies show that coordinates distances increase with increasing angle (See Figure 9.4(d)). The angle between Kinects correlates to the degree of rotation used in the transformation of multiple skeletons. The larger the angle is between Kinects, the skeletons will be rotated more, hence producing larger coordinates differences. The joints distances follow a similar pattern as shown in the plot with different tasks averaged over different Kinect placements. The left joints also have larger coordinates differences compared to the right joints, but there is less variation between joints, and the averages are slightly smaller (See Figure **??**).

When varying only either the complexity of the tasks or the angle of Kinect placement, the results

will show similar trends. In short, the more complex the task or the larger the angle between multiple Kinects, the worse results will be. The scenario where Δd is the smallest is the Stationary task with parallel Kinects (3.52 cm), and the scenario where Δd is the highest is the Walk task with 90° apart Kinects (32.38 cm). However, the average cases of varying only the task or Kinect placement show promising results, 14.10 cm and 10.57, respectively.

The worse average cases show that the Δd distances are around 20 cm (See Figure 9.4). This boundary is still within the personal space, or the space where only one person is most likely to exist. The results show preliminary success in tracking people using transformed 3D coordinates to find their spatial positions and joints.

Time

10.5 Obstacle

TODO

10.6 Interaction

TODO

10.7 Summary

The researcher summarizes the findings as follows:

1. Δx, Δy, Δz, and Δd increase with increasing complexity of tasks, from being stationary to walking
2. Δx, Δy, Δz, and Δd increase with increasing angle between Kinects
3. The torso (head, hip, knee, neck, shoulder, and spine) is more reliable than other body regions
4. The left joints are more reliable than the right joints
5. Δd is less than 15 cm on average, over different tasks and Kinect placements

10.8 Criticisms

The data collection (logging) procedure was flawed due to software failure. The logger did not log the same length of stationary movement for all participants and different Kinect placements. The lengths of the results in the Stationary task do not all equal to ten seconds as described in chapter 8.

The data cleaning process is also not rigourous. In both the Steps and Walk tasks, the participants remained in the same positions between instructions. The skeletons data for when the participants remained in the same positions should be discarded before doing any further analysis investigating the effects of task complexity and Kinect placements.

The analysis did not include significance testings on the hypotheses. The results are only descriptive, not inferrential.

10.9 Future Work

Future work can be divided into two domains, one for the tracking system, including the algorithms and application, and the other one for the user studies.

CHAPTER 11

Conclusion

CHAPTER 12

Ethics

CHAPTER 13

Acknowledgements

The researcher would like to thank Dr. David Harris-Birtill for his supervision, support, and optimism throughout the course of this project. The researcher would like to thank the School of Computer Science and the SACHI lab for an invaluable education and supportative environment. Lastly, the researcher would like to credit Aleksejs Sazonovs for providing the LaTeX template used in this report.

CHAPTER 14

Appendix

14.1 Kinect BodyFrame Serialization Library

Serialized BodyFrame:

- Timestamp
- List of serialized Bodies
- Depth frame width
- Depth frame height

Serialized Body:

- Is tracked
- Tracking id
- Dictionary of joint types and serialized Joints
- Clipped edges

Serialized Joint:

- Joint tracking state
- Joint type
- Joint orientation
- Camera space point
- Depth space point

14.2 Studies Log

Each tracking result contains:

- Study id (Participant id)
- Kinect configuration (The Kinect position, in angle, compared to the primary Kinect)
- Scenario (User task in the experiment)
- Tracker time
- Person id
- Skeleton id
- Skeleton time
- Skeleton initial angle
- Skeleton initial distance
- Kinect id
- Kinect tilt angle
- Kinect height
- Joint 1 (Ankle Left) X
- Joint 1 (Ankle Left) Y
- Joint 1 (Ankle Left) Z
- ...
- Last joint (Wrist Right) X
- Last joint (Wrist Right) Y
- Last joint (Wrist Right) Z

Bibliography

[1] Microsoft, "Coordinate mapping," 2015. `https://msdn.microsoft.com/en-us/library/dn785530.aspx`.

[2] J. Sherrah, B. Ristic, and N. Redding, "Particle filter to track multiple people for visual surveillance," *IET Computer Vision*, vol. 5, p. 192200, July 2012.

[3] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people," *Computer Vision and Image Understanding*, vol. 80, pp. 42–56, July 2000.

[4] M. Andriluka, S. Roth, and B. Schiele, "Monocular 3d pose estimation and tracking by detection," *In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 623–630, June 2010.

[5] S. Tang, M. Andriluka, and B. Schiele, "Detection and tracking of occluded people," *International Journal of Computer Vision*, vol. 110, no. 1, pp. 58–69, 2014.

[6] A. Gudy, J. Rosner, J. Segen, K. Wojciechowski, and M. Kulbacki, "Tracking people in video sequences by clustering feature motion paths," *In Computer Vision and Graphics*, pp. 236–245, June 2014.

[7] M. Munaro, F. Basso, and E. Menegatti, "Tracking people within groups with rgb-d data," *In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2101–2107, October 2012.

[8] J. Liu, Y. Liu, G. Zhang, P. Zhu, and Y. Q. Chen, "Detecting and tracking people in real time with rgb-d camera," *Pattern Recognition Letters*, vol. 53, pp. 16–23, 2015.

[9] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine Vision and Applications*, vol. 9, no. 5-6, pp. 272–290, 1997.

[10] B. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 5, no. 5, pp. 629–642, 1987.

[11] T. Wei, Y. Qiao, and B. Lee, "Kinect skeleton coordinate calibration for remote physical training," in *The Sixth International Conferences on Advances in Multimedia*, MMEDIA '14, pp. 66–71, IARIA, 2014.