Tracking People with Multiple Kinects

Chi-Jui Wu
Computer Science, University of St Andrews
United Kingdom
cjw21@st-andrews.ac.uk

ABSTRACT
This paper is my undergraduate thesis, completed in School of Computer Science, University of St Andrews, in 2015. The project is about tracking people with multiple Kinects. The goal is to reliably track people in uncertain, occluded real-world environments. The final submission contains an interactive application that demonstrates the tracking system as well as a series of user studies reporting the success of the system. The strengths and limitations of the system are discussed. An outline of future work to make the current system deployable in the real life is described.

Author Keywords
Tracking; Occlusion; Kinect; Calibration; HCI

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
People detection and tracking in realtime are essential in surveillance, interactive systems, medical imaging, and humanoid robotics.

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

Word Count
5,385

PROBLEM STATEMENT
The task of detecting and tracking moving targets is non-trivial. There are many sources of tracking errors, including raw sensor data noise, illumination levels, changing backgrounds, and occlusion. Real-world environments are unpredictive and complex, thus making the task much harder. The system attempts to solve the problme of occlusion with multiple Kinects.

University of St Andrews, April 10th, 2015, United Kingdom.

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 occlusion refers to occlusion caused by stationary objects, and dynamic occlusion refers to occlusion arised from interactions of many targets in the environment. The aim is to resolve both types of occlusion.

A simple instance of the problem is illustrated in Figure 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.

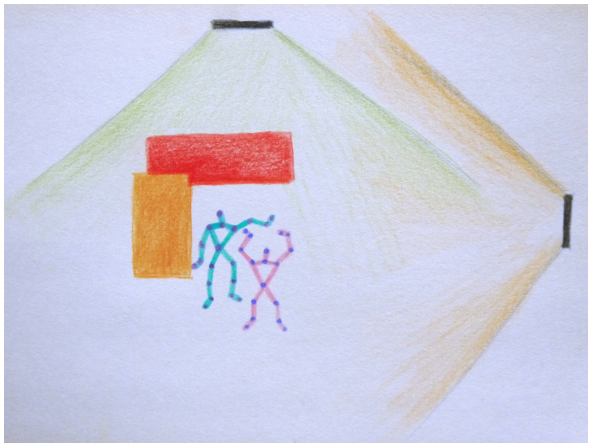

Figure 1: The occlusion problem

Aims and Objectives
TODO

CONTRIBUTIONS
The contributions of the current work are:

1. Replicate and validate current research
2. Tracking people in occluded environment with multiple Kinects
3. A Kinect BodyFrame serialization library
4. ...

PREVIOUS WORK
TODO

Existing people detection and tracking techniques. Tracking people in surveillance video and in real time. Tracking using mobile and wearable devices. Motion sensors and wireless. Time-of-flight and structured-light cameras.

1. Particle filter to track multiple people for visual surveillance
2. Tracking People in Video Sequences by Clustering Feature Motion Paths
3. Evaluation of real time people tracking for indoor environments using ubiquitous motion sensors and limited wireless network infrastructure
4. Tracking people under heavy occlusions by layered data association
5. Detection and Tracking of Occluded People

Tracking people
TODO

Coordinate transformation
TODO

[1] [2]

Tracking using depth data
TODO

Tracking using color data
TODO

1. Tracking people within groups using RGB-D data
2. Detecting and tracking people in real time with RGB-D camera
3. Applications for a people detection and tracking algorithm using a time-of-flight camera
4. Real-time Human Motion Tracking using Multiple Depth Cameras
5. Human Detection Using Depth Information by Kinect

KINECT
TODO

The specification and components. Include image Larger field of views. Give examples. Define field of view.

The complete API reference for Kinect v2 is accessible at https://msdn.microsoft.com/en-us/library/windowspreview.kinect.aspx.

Features
TODO

Coordinate System
Kinect Camera Space. The coordinates are 3D points (x, y, z), expressed in meters. The origin of the coordinate system at (0, 0, 0) is at the center of the Kinect's IR sensor. The x axis grows to the left of the Kinect, the y axis grows upward, and the z axis grows outward from the direction of the sensor.

[3]

Multiple Kinects
TODO

CURRENT APPROACH
This section describes the people tracking algorithm.

The current work is an extension of the Wei et al study [4]. It applies the same algorithms for doing calibration and coordinate transformation on skeleton joints. The differences are that the current system accommodates multiple Kinects (not limited to parallel with other) and multiple people (six people). The number of tracked people is limited by the maximum number of skeletons one Kinect can recognize through its BodyFrame stream.

The complete system architecture is explained in section "System". In short, the clients send Kinect BodyFrames to the server. The server will perform both the calibration and tracking. On the other hand, Wei et al designed the system so that each indivdual machine running a Kinect will perform its calibration then transmit the transformed coordinates to the server [4] **TODO: DOUBLE CHECK**. The current approach puts weights on the server machine and requires less computational resources on the client side. The clients are essentially Kinect hotspots, simply redirecting depth information to the server. However, the tradeoffs of two different approaches are not studied.

The calibration process allows the system to precisely transform the skeleton joints coordinates into a new coordinate system, also known as the World View. The system can now compare skeletons in different Kinects fields of view based on their spatial positions in the World coordinate system. Furthermore, the system can transform any skeleton in World View back to the perspective of any connected Kinect, because it knows the initial position and angle of the skeleton in that particular Kinect's camera space.

After calibration, the system would combine skeletons of each tracked person in multiple Kinects. This is the initial tracking result. When the server receives a new BodyFrame from a client, or effectiviely a Kinect, the system would update each skeleton's spatial position in that particular field of view. Therefore, the person represented by the skeleton will also have his position updated.

Any imperfect transformation process will produce errors **TODO: ERRORS???**. For instance, for skeletons of the same person in different Kinects fields of view, the algorithm

may produce two skeletons that are in similar, not same, spatial locations in the World View. To overcome this inherent handicap, the system will minimize the skeleton joint differences by creating an average skeleton for every person. The average skeleton is the average of all skeletons of a person.

Kinect BodyFrame

The Kinect BodyFrame contains real-time skeletal information of people who are trakced by a Kinect sensor. It was called SkeletonFrame in the Kinect v1 SDK. The BodyFrames are computed by the Kinect sensor internally from its depth data; they provide a higher level API for programming with skeletons. The current system uses the preprocessed information to track individuals.

Each frame contains at most six Kinect Body data, where each Body represents a skeleton from the Kinect's field of view. A Body contains a skeleton's joints coordinates and metadata.

Serialization

TCP network clients and server exchange data in bytes, thus requiring any content passed between the two nodes to be serialized. Unfortunately, the Kinect v2 SDK does not support serialization of Kinect BodyFrame. To resolve the inconvenience, the current work builds a Kinect BodyFrame serialization library. The most important pieces of information are the skeleton joints, their coordinates in the Kinect Camera Space, types and tracking states. See Appendix for a complete list of serialized properties.

Calibration

To perform coordinate transformation on a skeleton from its current Kinect field of view into the World coordinate system, the system requires its initial center position and initial angle between itself and the Kinect.

The following definitions are from the Wei et al study [4].

Initial center position is defined as the average of a skeleton's joints coordinates over the duration of calibration.

Let J = number of joints per skeleton (20). Let T = number of frames used for calibration (120).

Let S = joints sum. Let A = joints average. Let C = joints center (over time).

$$S(X_S, Y_S, Z_S) = \sum_{j=1}^{J} (X_j, Y_j, Z_j) \tag{1}$$

$$A(X_A, Y_A, Z_A) = S(X_S, Y_S, Z_S)/J \tag{2}$$

$$C(X_C, Y_C, Z_C) = A(X_A, Y_A, Z_A)/T \tag{3}$$

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

$$\begin{gathered} \text{Let } Z_r = \text{Right shoulder z coordinate} \\ \text{Let } Z_l = \text{Left shoulder z coordinate} \\ D = Z_r - Z_l \end{gathered} \tag{4}$$

$$\begin{gathered} \text{Let } X_r = \text{Right shoulder x coordinate} \\ \text{Let } X_l = \text{Left shoulder x coordinate} \\ W = X_r - X_l \end{gathered} \tag{5}$$

$$\theta = \arctan(D/W) \tag{6}$$

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.

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.

(TODO: PSEUDOCODE).

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:

$$J_t(X_{Jt}, Y_{Jt}, Z_{Jt}) = (X_j - X_C, Y_j - Y_C, Z_j - Z_C) \tag{7}$$

$$J_w(X_{Jw}, Y_{Jw}, Z_{Jw}) = (X_{Jt}\cos\theta + Z_{Jt}\sin\theta, Y_{Jt}, Z_{Jt}\cos\theta + X_{Jt}\sin\theta) \tag{8}$$

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.

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.

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.

Strengths

TODO

Limitations

TODO

Improvements

TODO

SYSTEM

The final application demonstrates that the system works as intended. 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 is a standalone application written in C# (.NET Framework 4.5), but it can also integrate with other applications, such as a heart rate estimation application using Kinect color data. More details on the design and implementation of the application will be discussed in section "**??**".

Running the application

TODO

Design

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 start and stop the server, recalibrate, view tracked skeletons from different views, and send the average skeleton stream to other applications. The user interface will fire a "OnDisplayResult" event when the rendering of the skeleton visualization is completed. This lets the server know when to signal the logger.

The overall control flow of the application is now clear. The server handles data from the clients. The tracker processes

the data. The user interface visaulizaes the data and provides additional control for the user. If required, the server then tells the logger to store the tracking done after the visualization is shown.

(todo: implement recalibration and stop button and the last feature)

The application does not take the following elements into design decision:

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

System architecture

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 2). 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.

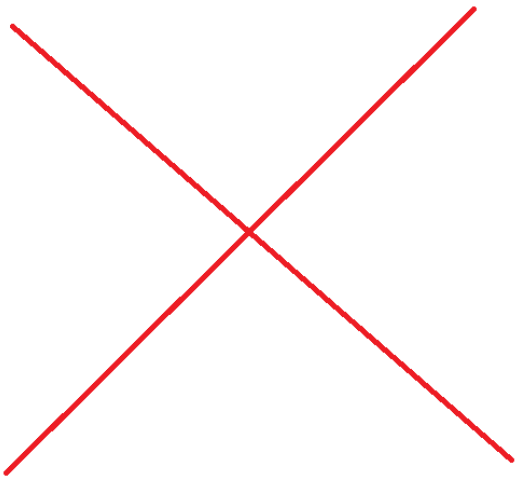

Figure 2: An overview of the system topology

Client Implementation

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.

Server Implementation

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

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 aseembles 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.

Communication Protocol

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 netwrok stream, and the above procedure repeats.

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

User Interface Implementation

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.

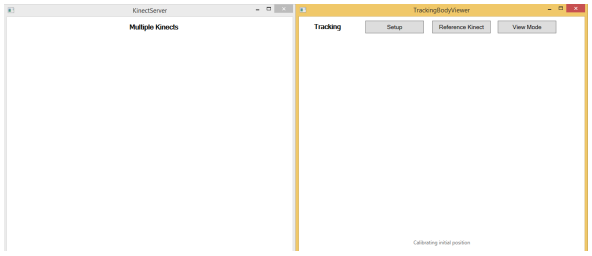

Figure 3: UI

Controls

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 the start button is pressed, signaling the server to start running, the setup button, which parameterizes the server, should be disabled. A full set of constraints on the availability of the buttons are specified below:

(todo: should I put this down?)

Events

When the user interface receives events from the server about new and old connections with clients, the user interface adds and remvoes 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".

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 avaialble colors, because the system sets the cap of number of tracked people to six.

The skeleton visaulization 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 Microsfot Kinect Developer examples. The inferred bones are displayed thinner than the tracked bones.

(todo: default to local Kinect)

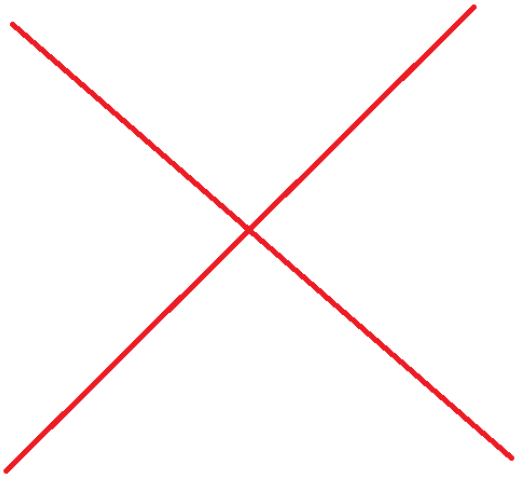

Figure 4: The Tracking View

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.

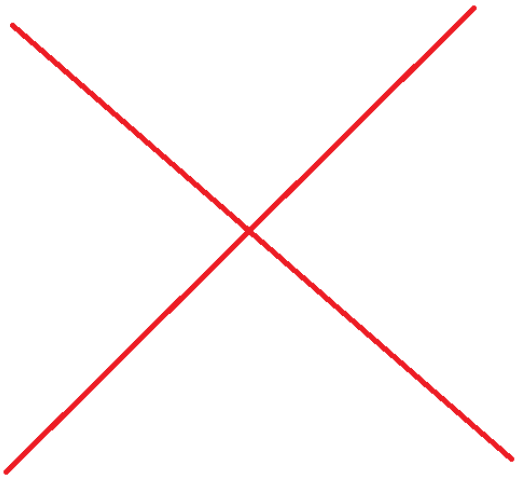

Figure 5: The Disjoined View

Tracker Implementation

The tracker runs the tracking algorithm on the Kinect BodyFrames from multiple Kinects. It is the brain of the application and the system, hence the most complex. There are no generic data structures in the whole application except the tracker.

Logger Implementation

TODO

"Disjoined" or called something else

Integration with Heart Rate Monitoring

TODO

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.

It is worth noting that the best tracking results, which minimize overall differences in joint coordinates, are when the Kinects are placed parallel to the ground floor with near 0 degree tilting angle. The researcher then investigates the performance of the tracking algorithm with the aforementioned contraints in various artificial scenarios. See the sections on User Studies for more details and the section on Future Work for improving the current system.

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.

todo: show different cases where the system is working properly, with images

USER STUDIES

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.

Figure 6

Hypotheses

The null hypotheses are as follows:

1. The differences in each joint coordinate among all potential skeletons of a person are consistent across time and are within five centimeters

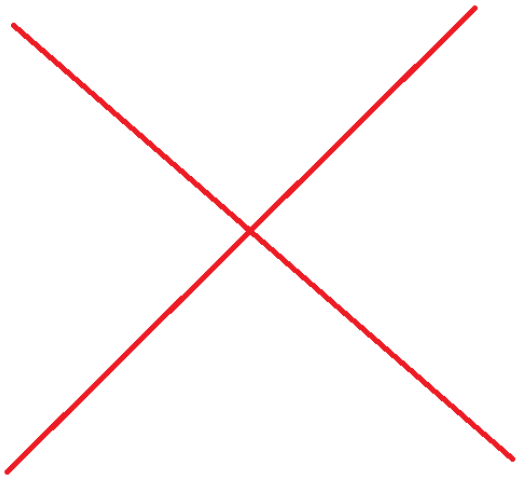

Figure 6: A participant doing the experiment

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

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.

Computer Specification

The server machine is running Microsoft Windows 8 on a i5 processor and 8 Gb RAM **(todo: add details)**. The other client machine is also running Microsoft Windows 8, on a i7-3610QM CPU at 2.30 GHz and 8 GB RAM.

Kinect Specification

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

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.

Setting

The studies take place in a semi-controlled environment (See Figure 7). The two Kinects are placed at either three predefined 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.

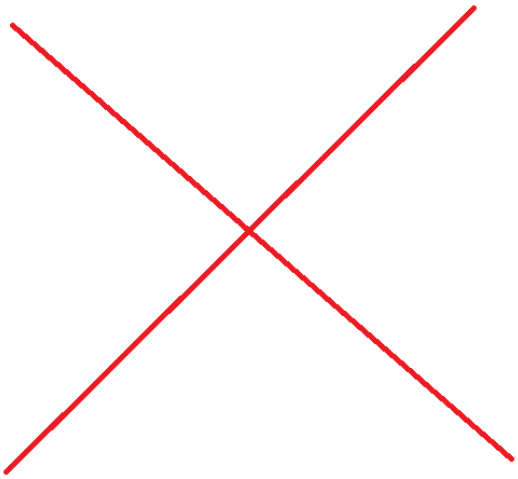

Figure 7: The setting where the user studies took place

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 would tell the participant to move forward (See Figure 8). 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.

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

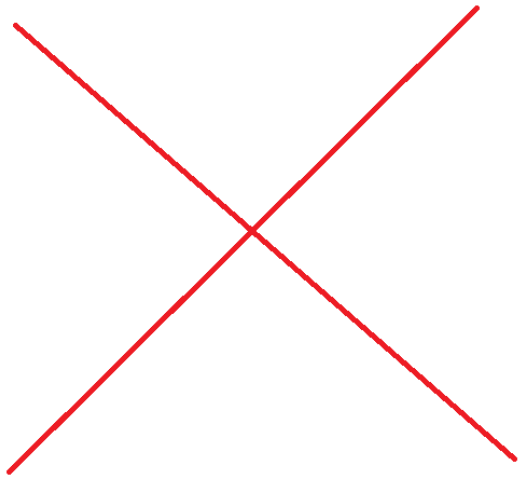

Figure 8: Instructions during the user studies

feels uncomfortable with the guideline is welcome to speak to the researcher and his supervisor.

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.

Study 2: 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.

Study 3: 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.

Study 4: Two people Cross-over

The fourth 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.

Study 5: Interaction 2?

TODO

Study 6: Occlusion

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 9). 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.

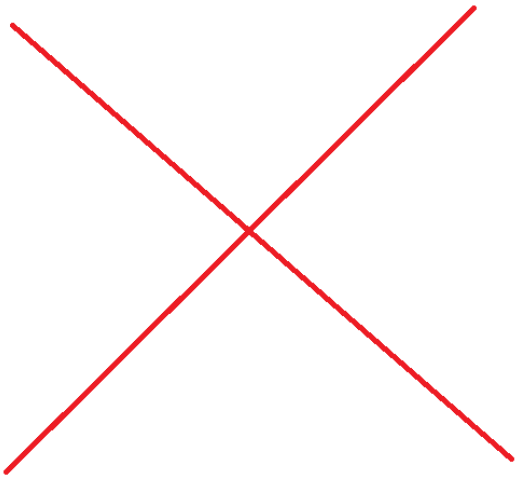

Figure 9: The obstacle in the actual experiment

EVALUATION

Discuss results. Compare them with Wei et al.

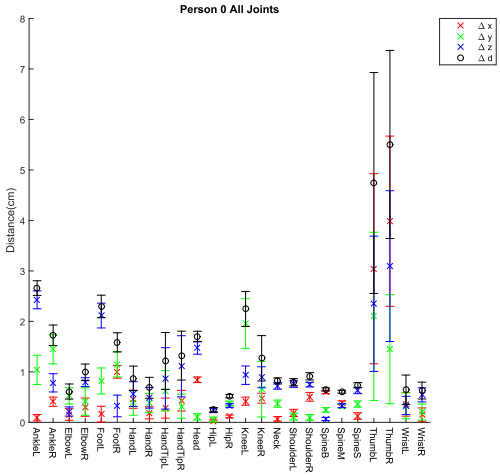

Figure 10: One person all joints

FUTURE WORK

TODO

User studies

TODO

Figure 11: One person average

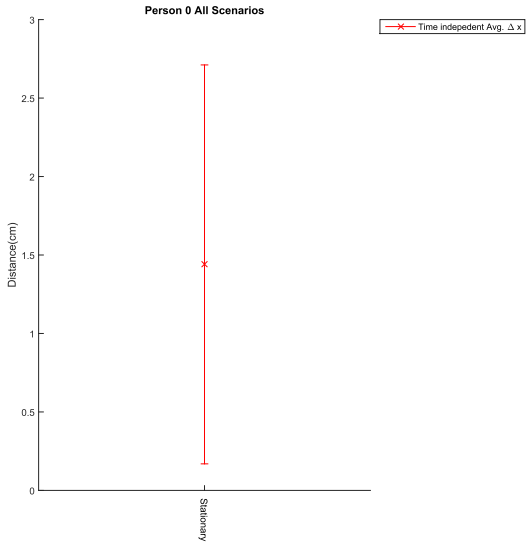

Figure 12: All scenarios

Application
TODO

SH PROJECT REFLECTION
TODO

Requirements

ACKNOWLEDGEMENTS

I, Chi-Jui Wu, hereby declare that, unless otherwise stated, the work is completed by myself, under the supervision of Dr. David Harris-Birtill. The code for skeleton visualization is modified based on the official Microsoft Kinect Developer example. All infographics are created by Ching Su. I own the copyright to this work.

REFERENCES

1. D.W. Eggert, A. Lorusso, and R.B. Fisher. 1997. Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications* 9, 5-6 (1997), 272–290. DOI: http://dx.doi.org/10.1007/s001380050048
2. B.K. Horn. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A* 5, 5 (1987), 629–642. http://dx.doi.org/10.1364/JOSAA.4.000629
3. Microsoft. 2015. Coordinate mapping. (2015). https://msdn.microsoft.com/en-us/library/dn785530.aspx.
4. T. Wei, Y. Qiao, and B. Lee. 2014. Kinect Skeleton Coordinate Calibration for Remote Physical Training. In *The Sixth International Conferences on Advances in Multimedia (MMEDIA '14)*. IARIA, 66–71. http://www.thinkmind.org/index.php?view=article&articleid=mmedia_2014_4_20_50039

Appendices

KINECT BODYFRAME SERIALIZATION LIBRARY

Serialized BodyFrame contains:

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

Serialized Body contains:

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

Serialized Joint contains:

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