Tiny: Tracking People using Multiple Kinects

Chi-Jui Wu
January 30, 2015

1 PROJECT

People detection and tracking in realtime are essential aspects of surveillance, medical monitoring, personalized robotics, and interactive systems. The task of detecting and tracking moving targets in real world environment using time-of-flight cameras is non-trivial. There are many sources of tracking errors, including sensor data noise, illumination levels, changing backgrounds, and occlusion. The current project aims to resolve the problem of occlusion in tracking people using multiple Kinects.

Real world environments are complex, unpredictable, and dynamic. Occlusion occurs when a tracked target is masked by other objects in the field of view of one or more cameras. The position and movements of the occluded target are unknown, hence increasing the difficulity of detection and tracking. There are two types of occlusions that are unfavorable to any tracking algorithm. Static occlusion refers to stationary objects that obstruct the visibility of the targets, for instance, a wall blocking the view of a target. Dynamic occlusion arises from the interactions of targets in the environment. An example would be two people walking past each other, where one person temporarily blocks the view of the other person. The proposed algorithm would consider both types of occlusions.

1.1 CONTRIBUTIONS

The goal of the project can be best illustrated in Figure 1.1. Both targets are occluded by the red obstacle, hence invisible to the Kinect on the other side of the obstacle. However, they are still visible to the other Kinect. The algorithm would use depth sensor information from multiple Kinects to track targets when they are not occluded, partially occluded, and fully occluded. Essentially, the algorithm leverages multiple Kinects and larger field of views to reconstruct the skeletons of the targets that would otherwise be impossible with a single Kinect. The algorithm makes tracking feasible by merging different field of views from multiple Kinects into a new aritifical coordinate system.

2 BACKGROUND

2.1 KINECT

3 SETUP

The current project is written in C# and uses the Microsoft Kinect SDK (v.2.0.1410.019000). The current Kinect SDK does not support multiple Kinects on a single machine; as a result, two machines are used for running two Kinects. Machine 1 is the server which runs the tracking algorithm. It has a 2.9 GHz CPU and 8.0 GB RAM. Machine 2 has a 2.3 GHz CPU and 8.0 GB RAM. Both machines runs the

Figure 1.1: The problem of occlusion

Microsoft Windows 8 Operating System, and they stream Kinect frames to the server on Machine 1. The server must start first before running the clients.

4 Method

4.1 Sockets

TPC soockets are used to transmit serialized Kinect skeleton data from the client to the server.

4.2 Serialization

Kinect frames are not serializable by default in the Kinect SDK. A library is written to serialize Kinect data into bytes, which would allow clients to trasnmit them to the server through TPC sockets. The following data are serialized and transmitted:

- BodyFrame
  1. Timestamp
  2. Depth frame width
  3. Depth frame height
- Body
  1. Is tracked
  2. Tracking id
- Joint
  1. Tracking state
  2. Joint type

3. Camera space point
4. Depth space point
5. Orientation

These data are then deserialized on the server-side. For more information on the data types, visit the Microsoft Kinect documentation [1].

4.3 Calibration

The tracking algorithm should calibrate all of the Kinects connected to the server. The process calculates the initial angle and center position of each skeleton with resepct to every Kinect. The initial angle is defined as the angle between the Kinect and the skeleton [INSERT Code Initial Angle]. The center position is defined as the average position of all the joints in the skeleton [INSERT Code Center Position]. These information are essential for converting 3D coordinates in the Kinect camera space to 3D coordinates in the world view coordinate system. This step is the foundation of matching skeletons across different Kinect field of views. Later, the algorithm would match skeletons from different field of views that have the most proximity to each other.

The 3D coordinates (x, y, z) in the Kinect camera space are expressed in meters. The origin of the coordinate system (0, 0, 0) is the center position of the IR sensor on Kinect [2]. The x axis grows to the left of the sensor. The y axis grows upward. The z axis grows outward from the direction the sensor. See [Image 1].

[Describe the worldview coordinate system]

[Insert image 1] The Kinect Camera Space

The server synchronizes the calibration, which is perfomred only once prior to the tracking process. For every Kinect, the algorithm uses the most recent 120 Kinect Body frames, or an equivalent of four seconds, to perform calibration. The server will automatically start the calibration process once it has received sufficient frames from all the clients. The calibration algorithm is taken from [2].

4.3.1 Initial angle

4.3.2 Initial center postion

4.3.3 Tracking

5 Testing

Figure 5.1 shows the tracking of one person without occlusion using two Kinects. The left image displays the skeletons of the same person from two difference Kinects' perspective. The right image displays the person's position calculated by the tracking algorithm after matching the skeletons belonging to that person. The skeleton in white color represents the average position of the skeletons from different Kinect FOVs.

6 Evaluation

7 References

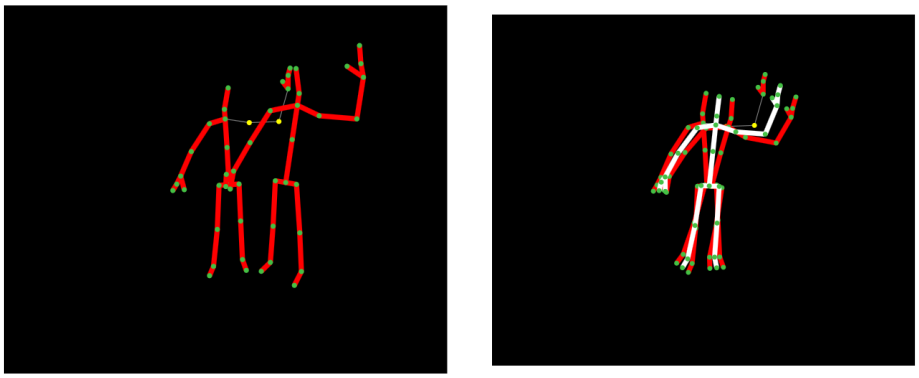

Figure 5.1: Tracking of one person with respect to the Kinect FOV of the left skeleton

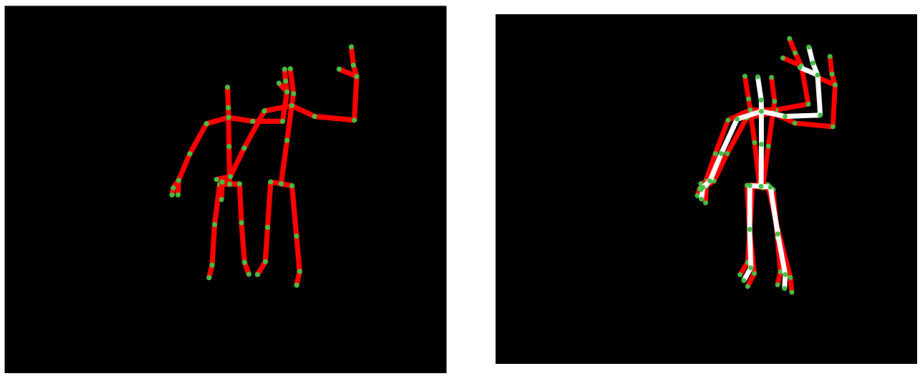

Figure 5.2: Tracking of one person with respect to the Kinect FOV of the right skeleton