

Wide Area Motion Tracking Using Consumer Hardware

Christian Schönauer
schoenauer@ims.tuwien.ac.at

Hannes Kaufmann
kaufmann@ims.tuwien.ac.at

Interactive Media Systems Group
Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria

ABSTRACT

In this paper, we present a wide area tracking system based on consumer hardware and available motion capture modules and middleware. We are using multiple depth cameras for human pose tracking in order to increase the captured space. Commercially available cameras can capture human movements in a non-intrusive way, while associated software-modules produce pose information of a simplified skeleton model. We calibrate the cameras relatively to each other to seamlessly combine their tracking data. Our design allows an arbitrary number of sensors to be integrated and used in parallel over a local area network. This enables us to capture human movements in a large arbitrarily shaped area. In addition we can improve motion capture data in regions, where the field of view of multiple cameras overlaps, by mutually completing partly occluded poses. In various examples we demonstrate, how human pose data is being merged in order to cover a wide area and how this data can easily be used for character animation in a virtual environment.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces – *Input devices and strategies*; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – *Animation, Virtual Reality*

General Terms

Algorithms, Measurement, Design, Human Factors

Keywords

Wide Area Tracking, Motion Capture

1. INTRODUCTION

Today, human Motion Capture (MoCap) is a central element of games and entertainment products. On the one hand, it is used to animate virtual actors in movies and computer games using professional and costly setups. On the other hand, recent developments in consumer hardware have made it possible to use MoCap - with and without a controller - as input to games and applications. Low-cost depth cameras enable consumers to capture human movements in an non-intrusive and stable way. Ready-to-use software modules and middleware, for retrieving

sensor data and human pose information of a simplified skeleton model, has been distributed by sensor manufacturers [7; 9]. Furthermore, application interfaces make integration in existing and customized applications possible. Finally, ease of deployment and good usability makes them applicable in a home environment. By moving control into the third dimension compelling experiences can be created and new ways of interaction established.

To a limited extent this has been explored in different ubiquitous computing projects, where boundaries between devices (ideally) become transparent to the user and a unified form of interaction can be applied (for an overview see [11]). However, the application of wide area tracking in such a context provides a whole set of new possibilities.

Nevertheless, systems, such as Microsoft's Kinect, are designed for use within a classical home entertainment setup and are therefore limited to confined spaces. The tracking volume is rather small due to the limited field of view of the integrated camera and technical constraints (i.e. range of IR illumination) restrict the possible distance between user and sensor. In addition, occlusions, especially with multiple users, result in incomplete postural information. To our knowledge, our system is the first to use multiple depth cameras to enhance full-body tracking data and expand the MoCap area beyond a single room. We are integrating commercially available depth cameras and motion capture middleware in a flexible networked software environment, which allows us to improve the tracking quality and scale the captured volume almost arbitrarily.

2. RELATED WORK

2.1 Wide Area Motion Capture Systems

Professional systems currently used for performance animation in movies and games are usually marker based infrared optical solutions, such as those from Motion Analysis [8] or Vicon [19]. These setups can be scaled to cover huge areas with multiple users by using large numbers of cameras. Xsense [22] and other companies, offer solutions for MoCap using inertial, mechanical or magnetic tracking. When using inertial sensors or mechanical tracking, larger volumes are possible, while the tracking quality is bound by the used technology e.g. subject to drift of inertial sensors.

Nevertheless, the high price of these systems hinders wide spread use in the near future. These solutions, by requiring markers or specific suits, put constraints on users, which in turn would drastically reduce acceptance for home use. Therefore, in the context of games and entertainment, non-invasive MoCap techniques have to be applied.

2.2 Motion Capture using Depth Cameras

A large effort in computer vision research has been put on human MoCap from camera RGB-images [3]. However, this is still a difficult problem in terms of reliability and stability, especially, if only few cameras are used.

Therefore, in previous years the use of time-of-flight and other depth cameras for MoCap has become an active research area [5; 10]. Only recently, this trend has extended to the use of depth cameras in games and entertainment. Multiple products have been released using structured light projection together with an infrared camera to acquire depth information [1; 6; 12]. Together with these sensors, software bundles are available, which provide human posture data calculated from the depth images in real-time e.g. using the algorithm described in [16]. While structured light has been used for shape reconstruction before [4], these systems offer entirely new possibilities for entertainment, technical enthusiasts and scientists.

Multiple research applications have been developed. These are using the depth information for e.g. facial animation [20] or posture information for e.g. gesture recognition [17].

With an easy-to-use hardware setup and reasonable computational effort, cheap MoCap solutions have made it into the living rooms of users and provide input to console and PC applications. For many games these systems are sufficient in terms of tracking quality and capture volume. However, with a view towards ubiquitous computing [11; 21] and setups other than the classical home entertainment center (with a TV placed in front of the couch in the living room), improvement is needed.

Work has already been conducted on merging multiple depth cameras to increase the volume of interaction and compensate for occlusions [21]. However, these approaches do not use full body tracking and are still limiting interaction to a room-sized environment.

3. THE SYSTEM

3.1 Hardware and Software

We are using consumer hardware (i.e. Microsoft's Kinect [6] or Asus Xtion [1]) together with the OpenNI [9] software

framework. OpenNI offers interfaces for low-level communication with devices as well as higher-level features such as skeleton tracking. For reasons of simplicity we will call the hardware components 'sensors' in the rest of the paper.

These sensors use only one camera for MoCap and work without additional markers or controllers. Instead, a dot pattern is projected onto the environment by an infrared laser projector. Projected points are traced by the camera. Projector and camera are positioned at a certain calibrated distance from each other within the sensor casing. Therefore, the dots recorded in the camera image and the original pattern can be used to reconstruct a depth image. The depth image in turn is used to calibrate and fit a human skeleton model, resulting in a skeleton pose.

In order to use data from multiple sensors, it has to be transformed into the same coordinate system. For that the relative positions of sensors to each other are needed. We utilize the MIP Multi Camera Calibration tool [15], which can be used to calibrate intrinsic and extrinsic parameters of RGB and depth cameras. The origin of our world coordinate system is that of an arbitrarily chosen master sensor. The other sensors' positions are calibrated relative to the master. Therefore, the sensors have to have an overlapping field of view. For sensors, which don't overlap with the master, transformations can be concatenated, as long as they can be calibrated relative to some other camera. Evidently, this might significantly reduce accuracy of the calibration, however, for many applications a good calibration relative to the next sensor will usually be sufficient. Alternatively, other methods of calibration incorporating visual features, the depth map or prior knowledge of the environment could be used to improve the global registration of the cameras.

3.2 Design

We are applying a software design, which enables us to almost arbitrarily scale the tracking volume. To achieve this, we are using a local area network to connect different sensors and to distribute tracking data. Furthermore, sensors are grouped in a modular way and arranged in a tree-structure connected by the network. Possible arrangements are shown in Figure 1.

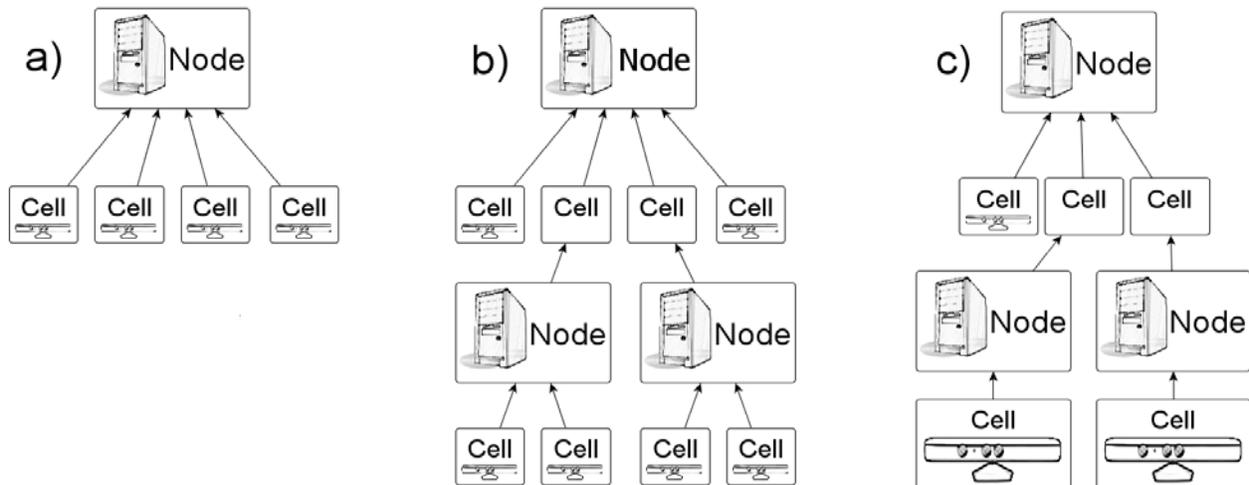


Figure 1: Arrangement of cells and nodes. a) node with local cells b) nodes with local and networked cells c) nodes with one cell each connected to a root node via network

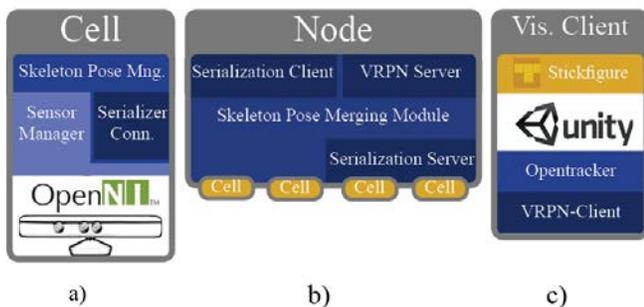


Figure 2: System diagrams of the different elements of our system. a) cell b) node c) visualization client

3.2.1 Cells and Nodes

The different elements in the structure of our system are called cell and node.

A cell consists of a single sensor together with associated software components to derive human pose data from it. A system diagram of a cell can be seen in Figure 2 a). It is the basic structure of our system.

A node is placed on top of one or more cells, collects tracking data and manages interaction between its cells. It communicates with other nodes and is therefore also responsible for network communication. A diagram of a node is depicted in Figure 2 b). One of the most important tasks of a node is the merging of MoCap data from different cells. It waits for input from all cells, which are currently tracking a user, and then utilizes the collected data to create the most probable skeleton pose. How different skeleton poses are fused is described in the next section. Usually, for reasons of applicability (e.g. length of sensor cables), a node will also be associated with the spatial structure of the building (i.e. a node will usually not extend beyond a room).

The root node of the tree-structure has a special task. It collects and merges data from all nodes and makes it available to other applications.

The elements described above enable us to create flexible network structures as depicted in Figure 1. The elementary setup consists of one node only with one or more local cells connected to it and is depicted in Figure 1 a). In a more elaborate configuration, multiple nodes are connected to cover a larger area as shown in Figure 1 b). With this setup multiple rooms can also be covered easily. Finally, Figure 1 c) shows a special case, where each node has only one local cell attached. This is the configuration we used in our tests for reasons described in the next section.

3.3 Implementation

Our system is implemented in C++ and uses OpenNI [9] to retrieve data from the sensors. NITE [9] provides software implementation for all OpenNI modules including the user generator. This module performs human skeleton tracking on a per-cell basis. Furthermore, we make extensive use of the Boost [2] library for various purposes like serialization and network communication.

3.3.1 The Cell: Acquisition of Pose Data

Each cell encapsulates OpenNI modules, which are necessary to retrieve user and depth data. The latter is used purely for visualization.

We implemented custom data structures for the use and serialization of data. Data is distributed over the local area network using TCP-connections. Two types of data have to be distributed: calibration data and skeleton pose data.

Initially, the skeleton of each user must be calibrated to reflect his real measurements. This has to be done only once while the user is within the tracking area. To perform the calibration the user has to stand in front, facing the sensor in Y-pose for about half-a-second as described in [9]. Once finished, the skeleton calibration is handed over to the containing node for distribution to other cells. This accounts for the current version of the NITE-middleware. However, Primesense [12] has already announced, that they are working on a new version, where no calibration pose is required.

Skeleton pose data is generated in all cells for every frame, whenever the sensor delivers an update. This is also the case for frames, where no user is detected by a sensor and an empty pose is generated. Updates in every frame are important for the merging process as discussed in 3.3.2.2. The cell sends the pose to the node, in which it is contained, for further processing. A schema of a skeleton pose object is depicted in Figure 3. It contains translation and rotation for each joint and a confidence value for each transformation, as delivered by OpenNI. In addition, for each joint transformation of the pose we add a weight attribute indicating the number of sensors used to generate it. This is important for the merging process, where a pose calculated from two sensors should have more importance than one from just one sensor. For one user the payload data of a pose update can be up to approximately one kilobyte. Therefore, the bandwidth required for the transmission of this data is rather small. Furthermore, in most cases the pose will be empty for a larger environment, since a user will only be tracked by a small number of cells.

3.3.2 The Node: Merging and Communication

As stated before, the node has two major purposes - merging of MoCap data and communication.

3.3.2.1 Communication

For communication, each node has a client and server available, which can establish TCP-connections between nodes. The client can be used to connect to one node higher up in the hierarchy (in the direction of the root). The server, on the other hand, can establish multiple connections with other nodes one level below in the hierarchy. One connection is established per node. It is communicated to the rest of the nodes as a networked cell, with the same properties as a local cell, only that it streams data from the network instead of from a sensor. Configuration of the

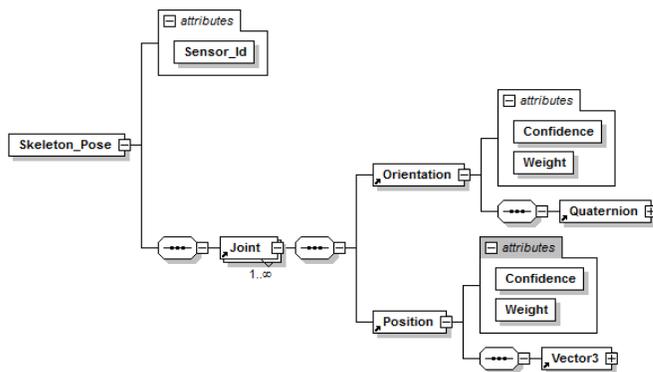


Figure 3: Schema of our skeleton pose structure

connections of each node is established via a local XML-configuration file or application arguments. Once the nodes are connected, the network can be used to serialize and pass objects between them.

The skeleton calibration is distributed from the root node via optional intermediate nodes to all cells. Therefore, it has to be performed only in one cell of the root node. The calibration is then automatically stored in a file and distributed to all cells. While local cells are loading the calibration from file, for distributed cells the data is serialized over a TCP-connection.

MoCap data is collected in a node in the form of the pose objects described in 3.3.1. No hardware synchronization of multiple sensors is available. For local cells synchronization of the cell data can be easily achieved on the software side by the mechanisms described in [9]. For cells receiving data over network connections, the transformations are updated as soon as they arrive and processed in the next loop of the merging process. If no update arrives for two frames the old data is considered invalid. If an empty pose object arrives the data of the corresponding sensor is also invalidated. Empty pose objects are otherwise ignored in the merging process. Due to the delay in the network communication the latency is slightly increased with networked cells. However, the small packet size of the pose updates limits network load and ensures fast updates.

3.3.2.2 Merging

Once a node has received updates from all connected cells (local and networked), it attempts to merge the MoCap data into one skeletal pose. This fusion is performed on a per-joint basis taking into account the confidence and weight attributes. Position and rotation of a joint are treated similarly, except that position vectors are linearly interpolated and for the rotation quaternions spherical linear interpolation is used. Therefore, we will discuss only position merging in the following.

First, the positions are sorted according to confidence. If only one position with the highest confidence is available, it is returned. Otherwise, starting with an empty combined position, the positions from the cells are added one by one. This is done by linear interpolation between combined and new position. The weight-ratio to the combined positions determines the contribution of the new position in the interpolation. After each added position, the weights are updated for the combined position. For positions this can be considered as the weighted average position. However, for future work we want to keep the implementation more flexible so more elaborate merging strategies can easily be incorporated, which might also take into account positions with less confidence or other attributes. However, in the current NITE implementation confidence values are limited to 0, 0.5 and 1 to distinguish between low and high confidence. Only transformations with a confidence of 1 were useful for our purpose, because lower confidence usually is associated with a very rough approximation. This largely decreases the options for merging the data. Once position and orientation are merged, they are updated for each joint in the merged skeleton pose, which is then sent up the hierarchy to the root node. Once all updates have arrived and are merged at the root level, tracking for the whole tracking area is complete and can be passed to the application.

3.3.3 Interfacing Applications

Nodes also contain an interface, which can be accessed by applications. This interface is usually only active for the root node

and can be configured via XML-file. It consists of a VRPN-server[14], which sends updates for each frame and has a station defined for each joint transformation.

3.3.4 Interfacing the Game Engine

To retrieve data from the root node we are using the VRPN-client implementation of OpenTracker. OpenTracker [13] is a multi modal data flow framework, which already provides interfaces to different tracking devices. Within OpenTracker tracking data can easily be manipulated and post processed (e.g. filtered) if necessary.

Furthermore, we have implemented an integration of OpenTracker into Unity3D [18]. Unity3D is a game engine and editor with a rapidly growing user base and can easily be extended using customized plug-ins and scripts. From within Unity3D we are starting an OpenTracker context with a custom configuration. Then our Unity plugin receives joint pose and transformation updates for the joints. These can be easily mapped to game objects or avatars by our animation scripts. For this work we have chosen to apply the transformation to a stick figure for visualization purposes. A schematic overview of this visualization client can be seen in Figure 2 c).

4. PRELIMINARY RESULTS

We have evaluated our system with different sensor placements to simulate situations which occur in wide area tracking. The test setup consists of three nodes with one cell/sensor attached to each as depicted in Figure 1 c). The nodes are run on three different Windows-PCs with moderate hardware – two dual-core notebooks and one quad-core PC. Currently, only one sensor can be used per PC in our setup. This is caused by the fact, that each sensor needs a separate USB-host controller but mainly due to limitations of the current NITE implementation. Its user generator only works properly with one sensor per PC.

4.1 Completing Tracking Data in Overlapping Regions

Here, we are taking a look at an overlapping area of two sensors. At the edge of the view frustum it can occur, that single limbs are already outside the field of view of the depth camera and are therefore not properly tracked. However, as long as the torso is visible, a good estimate for the rest of the body is usually possible. In case the missing limb is visible in another sensor, the pose can be easily completed. Therefore, placement of the sensors is crucial. The overlapping has to be large enough, so the seamless transition from one cell to the other is possible. On the other hand overlapping sensors produce disturbance (due to overlapping projected dot patterns) and therefore should be kept to a minimum. Figure 4 depicts the setup for this test of sensors placed side by side with a small overlap.

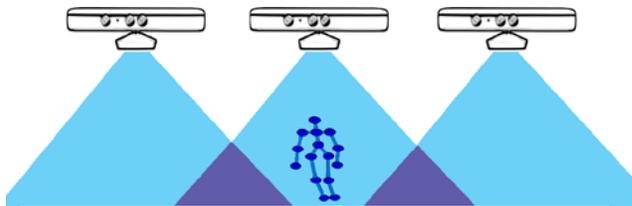


Figure 4: Sketch of the setup with three sensors

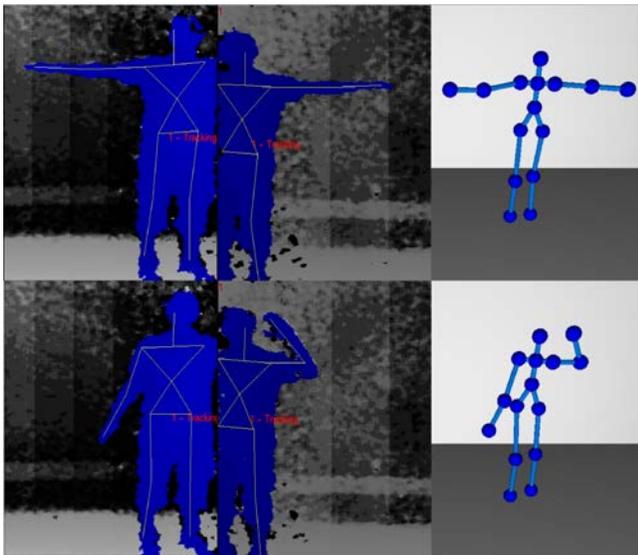


Figure 5: Demonstration of mutual completion of the pose by merging data from different nodes. Visualizations show the tracked user in the depth images of two sensors and corresponding skeleton pose as mapped to the stick figure by our visualization client

Figure 5 shows, how each arm is only visible in one sensor. Also, one leg is only tracked by one node. After merging the data our visualization client uses the completed correct pose for animation of the stick figure.

4.2 Extending the tracking area

4.2.1 Increasing the volume

To increase the tracking volume three sensors are placed side-by-side in a larger area with small overlapping regions. This setup could also be arbitrarily extended. Figure 4 shows the setup of the three sensors, while Figure 6 depicts snapshots from the depth images and avatar animation, while a user is changing from one cell to another. Note that the animated avatar in the bottom part of the figures appears slightly tilted in the areas on the side due to perspective projection.

4.2.2 Extending from room to room

For the final tests we were taking a look at the possibility of covering a larger indoor environment with multiple rooms. We have therefore placed two sensors inside the room and one outside the door. Figure 8 shows the correct transition from one room to the other. Placing sensors normal to each other also improves

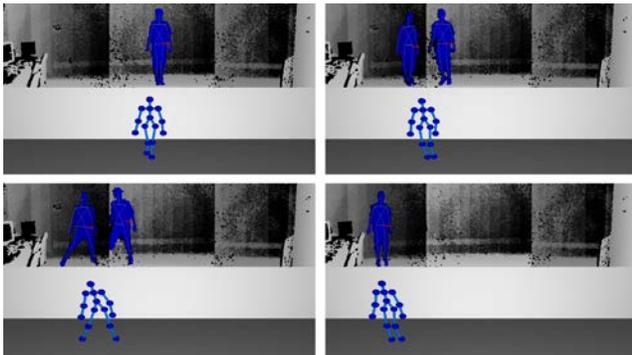


Figure 6: Different snapshots of one user changing from one cell to another.

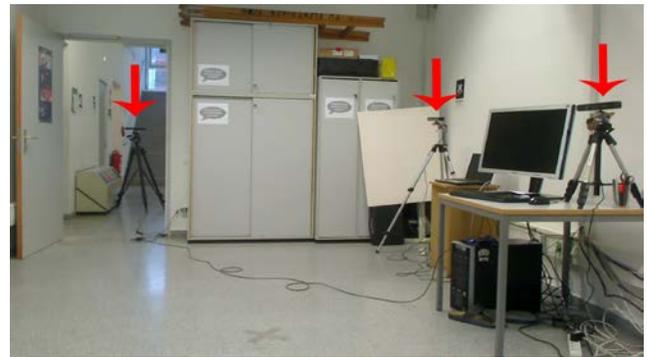


Figure 7: Photo of our setup with one sensor in the next room.

tracking of movements, which can only be captured poorly from some perspectives e.g. tracking walking from the side is difficult due to occlusions. However, small pathways make an exact extrinsic calibration of the sensors difficult and sometimes result in artifacts due to limited overlaps.

5. DISCUSSION AND FUTURE WORK

We have evaluated our wide area MoCap system in different situations, which frequently occur in multi-camera setups. Merging the skeleton pose at a higher level (as opposed to merging the depth data before a skeleton is fitted) improves tracking data in many cases. In addition, it makes the system easier scalable.

In the future we want to extend our work to the tracking of multiple users. In such a scenario we would have to keep track of the global position of each user and identify each newly detected user in a cell by a global id. This plan is somewhat hindered by the fact that the skeleton calibration produced by OpenNI is only available in a binary file of one megabyte or above. Therefore, serialization and transmission takes a considerable amount of time in our current implementation. While a waiting period of 20 seconds might be acceptable for a single user, longer interruptions with every newly added user in a multiuser environment would strongly decrease usability. Microsoft's KinectSDK would offer calibration-less tracking, but has no estimate for the confidence of the tracked joints. Therefore, combining data from multiple sensors in a meaningful way is difficult.

Finally, we want to apply more elaborate pose merging strategies incorporating movement trends and stronger emphasize confidence values. We are therefore, hoping for refined confidence values in Primesense's NITE implementation.

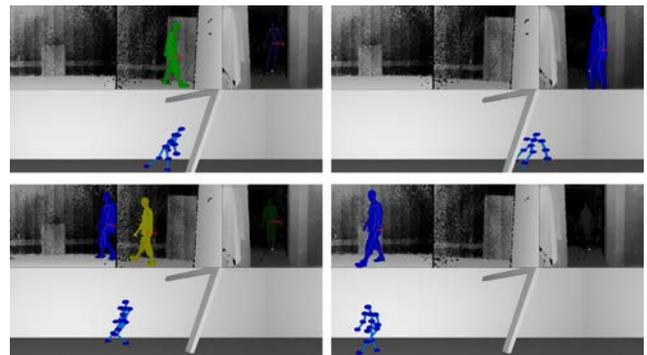


Figure 8: Snapshots of a tracked user from setup with orthogonal sensor placement in two rooms. Left side shows how multiple perspectives improve tracking. Right side shows the user in two different rooms.

6. REFERENCES

- [1] ASUS 2011. Xtion Pro Motion Sensing http://event.asus.com/wavi/product/WAVI_Pro.aspx, last visited Jul. 2011.
- [2] BOOST 2011. Boost C++ Libraries <http://www.boost.org/>, last visited Jul. 2011.
- [3] DEUTSCHER, J. AND REID, I. 2005. Articulated Body Motion Capture by Stochastic Search. *International Journal of Computer Vision* 61, 185-205.
- [4] KAWASAKI, H., FURUKAWA, R., SAGAWA, R. AND YASUSHI, Y. 2008. Dynamic scene shape reconstruction using a single structured light pattern. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1-8.
- [5] KNOOP, S., VACEK, S., STEINBACH, K. AND DILLMANN, R. 2006. Sensor fusion for model based 3D tracking. In *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*, 524-529.
- [6] MICROSOFT 2011. Kinect full body interaction <http://www.xbox.com/kinect>, last visited Jul. 2011.
- [7] MICROSOFT 2011. KinectSDK for Windows <http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/>, last visited Jul. 2011.
- [8] MOTION-ANALYSIS 2011. Motion Analysis: Passive Optical Motion Capture <http://www.motionanalysis.com/>, last visited Jul. 2011.
- [9] OPENNI 2011. Natural Interaction Middleware <http://www.openni.org/>, last visited Jul. 2011.
- [10] PLAGEMANN, C., GANAPATHI, V., KOLLER, D. AND THRUN, S. 2010. Real-time identification and localization of body parts from depth images. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 3108-3113.
- [11] POSLAND, S. 2009. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. John Wiley & Sons.
- [12] PRIMESENSE 2011. PrimeSensor 3D-sensing technology <http://www.primesense.com/>, last visited Jul. 2011.
- [13] REITMAYR, G. AND SCHMALSTIEG, D. 2001. An open software architecture for virtual reality interaction. In *Proceedings of the ACM symposium on Virtual reality software and technology*,
- [14] RUSSELL M. TAYLOR, I., HUDSON, T.C., SEEGER, A., WEBER, H., JULIANO, J. AND HELSER, A.T. 2001. VRPN: a device-independent, network-transparent VR peripheral system. In *Proceedings of the Proceedings of the ACM symposium on Virtual reality software and technology*, Baniff, Alberta, Canada2001 ACM, 505019, 55-61.
- [15] SCHILLER, I. 2011. MIP - MultiCameraCalibration <http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Calibration>, last visited Jul. 2011.
- [16] SHOTTON, J. 2011. Real-Time Human Pose Recognition in Parts from a Single Depth Image. In *Proceedings of the CVPR2011 IEEE*.
- [17] SUMA, E., KRUM, D., LANGE, B., RIZZO, S. AND BOLAS, M. 2011. FAAST: The Flexible Action and Articulated Skeleton Toolkit. In *Proceedings of the IEEE Virtual Reality, to appear 2011*.
- [18] UNITY-TECH. 2011. Unity3D game engine <http://unity3d.com/>, last visited Jul. 2011.
- [19] VICON 2011. Vicon motion capture system <http://www.vicon.com>, last visited Jul. 2011.
- [20] WEISE, T., BOUAZIZ, S., LI, H. AND PAULY, M. 2011. Realtime Performance-Based Facial Animation. In *Proceedings of the SIGGRAPH 2011*.
- [21] WILSON, A.D. AND BENKO, H. 2010. Combining Multiple Depth Cameras and Projectors for Interactions On, Above, and Between Surfaces. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology* Association for Computing Machinery, Inc., 273-282.
- [22] XSENSE 2011. Wireless Inertial Motion Capture <http://www.xsens.com/>, last visited Jul. 2011.