Wide Area Motion Tracking Using Consumer Hardware

Christian Schönauer and Hannes Kaufmann
Vienna University of Technology, Austria

Editor: Zhigeng Pan

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.

***Index Terms*—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 [1, 2]. 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 [3]). However, the application of wide area tracking in such a context provides a whole set of new possibilities: As introduced in [4] *MoCap* data can be used to interact with three-dimensional menus. These gestures combined with wide area tracking could, e.g., control audio or video playback in different entertainment products from arbitrary positions in the user's environment. In a game a virtual avatar could follow around a player in his apartment and could be interacted with by gestures. If the game is distributed on different platforms, a PC, game console, or smartphone could then provide visualization of the game content. Also, beyond entertainment, a wide area tracking system could be used for safety applications monitoring, e.g. elderly and detect falls or gestures calling for help.

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 [5] or Vicon [6]. These setups can be scaled to cover huge areas with multiple users by using large numbers of cameras. Xsense [7] 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 [8]. 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 [9, 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 [11-13]. 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 [14]. While structured light has been used for shape reconstruction before [15], 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 [16] or posture information for, among other things, 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 [3, 4] 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 [4]. 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* [12] or Asus *Xtion* [11]) together with the *OpenNI* [2] 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 [18], 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 positions of the other sensors are calibrated relative to the master. Therefore the sensors have to have an overlapping field of view. With the *Kinects*’ cameras we take multiple pictures of an A0-sized checkerboard calibration pattern in the overlapping regions. From the images we calculate the transformation from one camera space to the other using the calibration tool. For sensors that don't overlap with the master, transformations can be concatenated as long as they can be calibrated relative to some other camera. This might significantly reduce accuracy of the calibration, but for many applications a good calibration relative to the next sensor will be sufficient.

Good calibration of the sensors requires some effort because multiple pictures have to be taken with the calibration pattern in different positions that require manual annotatioon or deselection where automatic detection of the pattern fails. Calibration of one camera relative to another can therefore take up to an hour. Alternatively, other methods of calibration incorporating visual features, the depth map, or prior knowledge of the environment can be used to improve the global registration of the cameras. In future work we are planning to use the point clouds and iterative closest point algorithms to determine the extrinsic calibration.

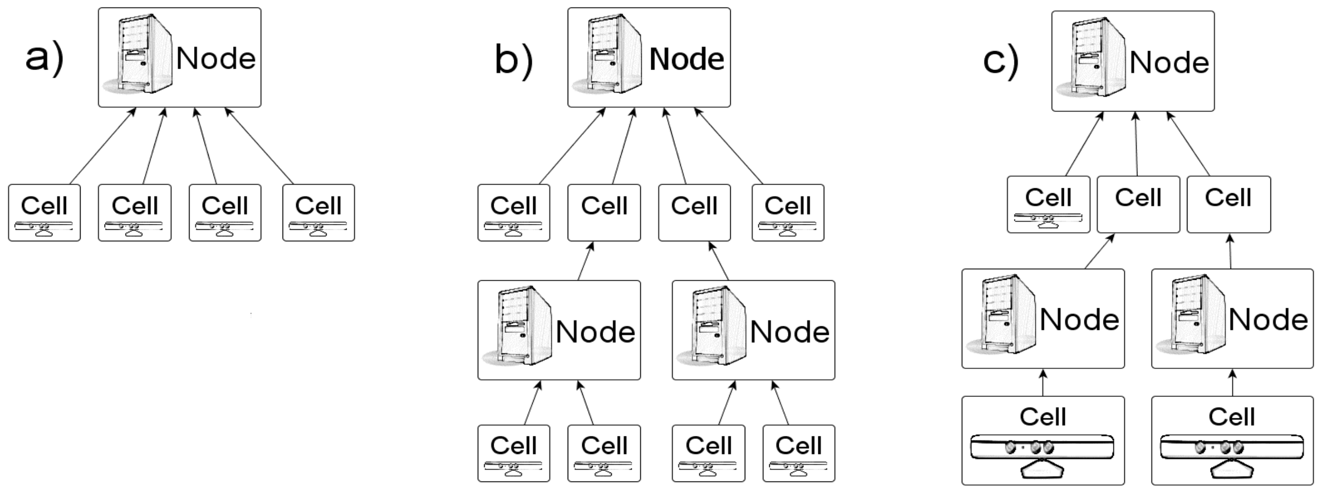

Figure 1: Arrangement of cells and notes: (a) node with local cells, (b) nodes with local and networked cells, (c) nodes with one cell each connected to a root node via a network.

3.2 DESIGN

We are applying a software design that 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.

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).

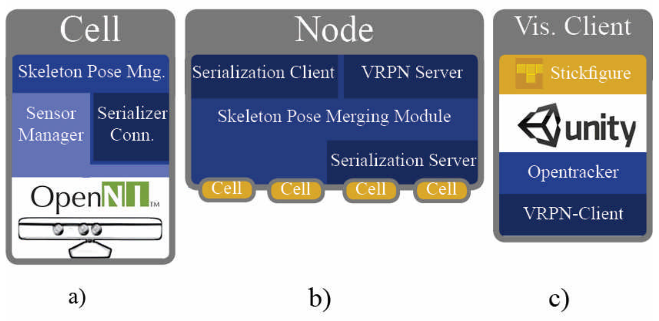

Figure 2: System diagrams of the different elements of our system. (a) cell, (b) node, (c) visualization client

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* [2] to retrieve data from the sensors. NITE [13] 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 [19] 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 [2]. 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 [13] 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 for Figure 2 (b): 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 as well as 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. Thus 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.

a) 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 (i.e. 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, except that it streams data from the network instead of from a sensor. Configuration of the connections of each node is established via 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 the 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 [2]. For cells receiving data over network connections the transformations are updated as soon as they arrive, and are processed in the next loop of the

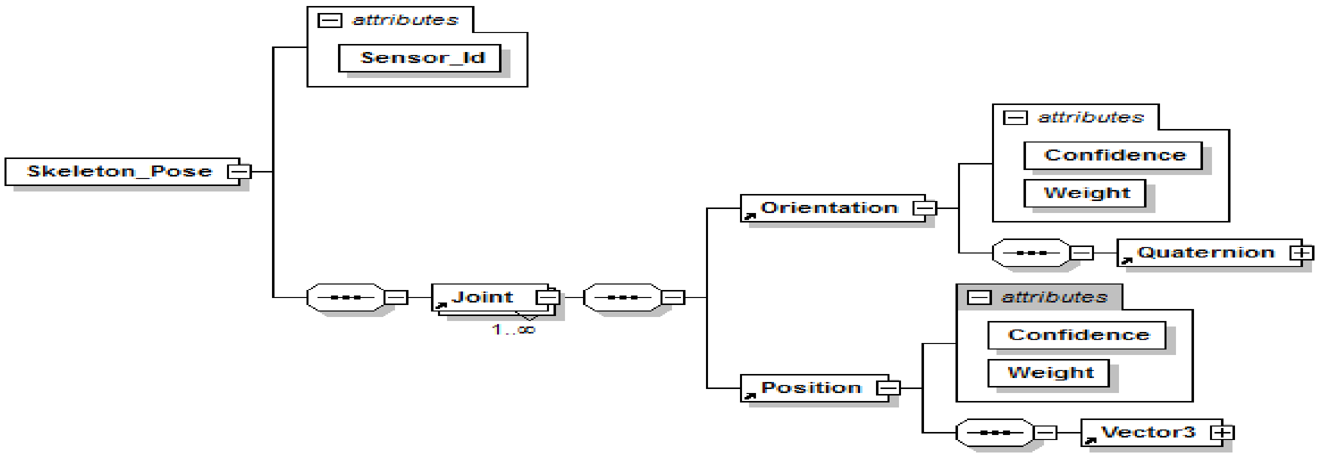

Figure 3: Schema of our skeleton pose structure.

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. Otherwise empty pose objects are 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 the network load and ensures fast updates.

b) Merging

Once a node has received updates from all connected cells, both 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 shall discuss only position merging in the following.

First, the positions are sorted according to the level of 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. 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 is usually 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 [20], 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* [21] is a multi-modal data flow framework that 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* [22]. *Unity3D* is a game engine and editor with a rapidly growing user base, and it can easily be extended using customized plug-ins and scripts. From within *Unity3D* we are starting an *Opentracker* context with a custom

4. Preliminary Results

We have evaluated our system with different sensor placements to simulate situations that 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. The quality of the merged tracking data is, of course, to a large degree dependent on the accuracy of the posture delivered by NITE to each cell. A short evaluation of the accuracy of the tracking data produced by one *Kinect* can be found in [23].

Moreover, the quality of fused data depends on the extrinsic calibration of the sensors. With the method and tool presented in section 3, we have achieved calibration accuracy of a few centimeters, limited by the sensor's limited shared field of view. However, for most applications in entertainment, e.g. animation of an avatar, the current calibration accuracy is sufficient since with the sensor's given limited tracking accuracy it doesn't produce new visible artifacts. Nevertheless, we hope to improve the extrinsic calibration in the future.

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 single limbs already outside the field of view of the depth camera can occur 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 a 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 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 can also be arbitrarily extended. Figure 4 shows the setup of the three sensors placed at an approximate height of 1.3 meters and 2.5 meters apart. Experiments with locating them in higher positions and tilting downwards strongly decreased tracking quality of the lower extremities. In our experience level alignment works best. Overlap was about 0.5 meter at the range of the tracked user. 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. Facing the sensors and stepping sideways from one sensor's field of view to the next works very well because in this position there is a lot of data available to infer the user's posture. On the other hand, tracking a user from the side usually produces less favorable results because one arm and leg are at least partly occluded by the rest of the body, limiting. posture information, and poor confidence often results. In this scenario compensating by merging data is not good because all sensors have similar perspectives on the user. Placing sensors orthogonally, as described in the following subsection, greatly improves tracking quality in these cases.

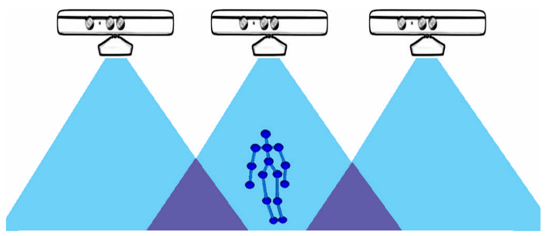
Figure 4: Sketch of the setup with 3 sensors

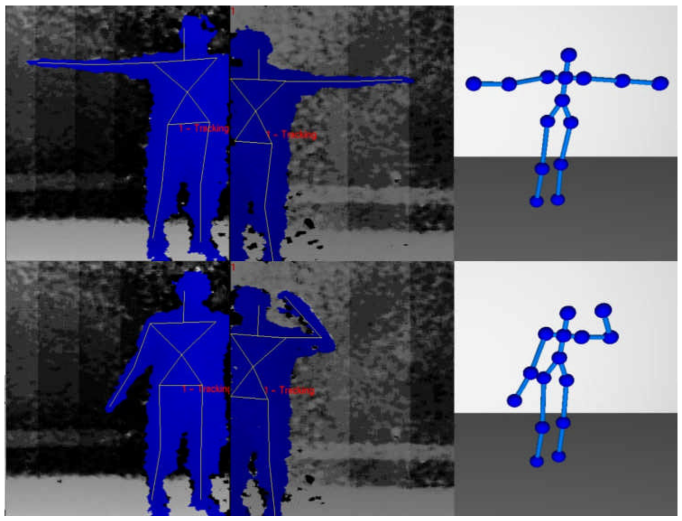

Figure 5: Demonstration of mutual completion of the pose by merging data from different nodes. Visualizations show the tracked user in the depth images of two sensors and corrsponding skeleton pose as mapped to the stick figure by our visualization client.

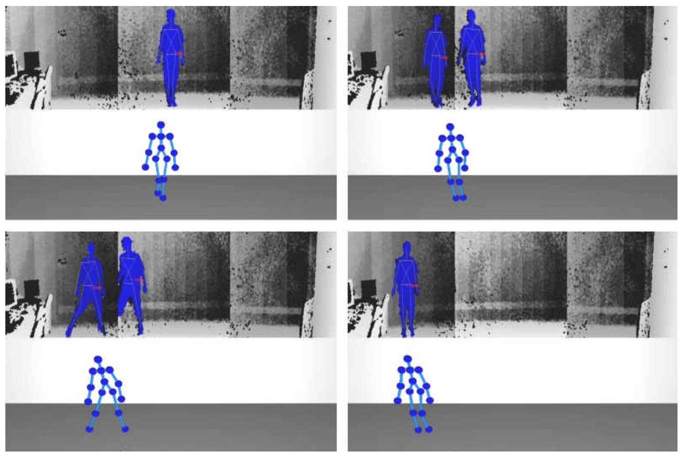
Figure 6: Different snapshots of one user changing from one cell to another.

4.2.2 Extending from Room to Room

For the final tests we looked at the possibility of covering a larger indoor environment with multiple rooms. We placed two sensors inside the room and one outside the door as depicted in Figure 7. Figure 8 shows the correct transition from one room to the other. Placing sensors normal to each other also improves tracking of movements, which can only be captured poorly from some perspectives (e.g. tracking the walking from the side is difficult due to occlusions). However, small pathways make an exact extrinsic calibration of the sensors difficult and sometimes resultx in artifacts due to limited overlaps.

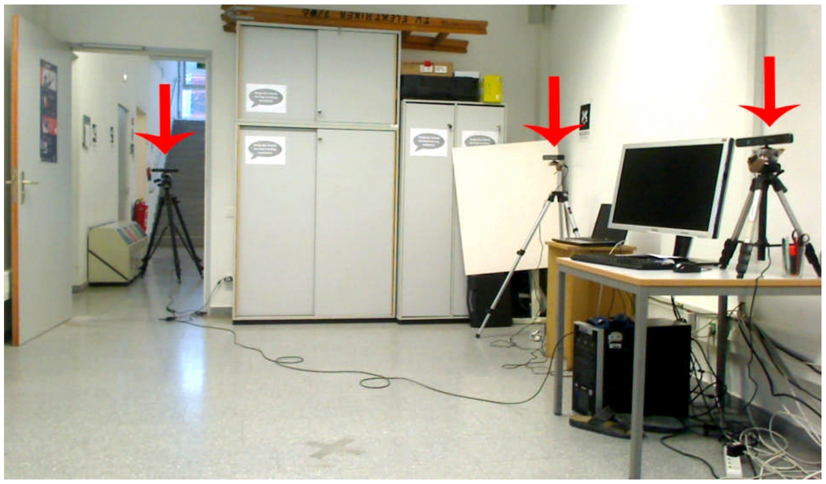

Figure 7: Photo of our setup with one sensor in the next room.

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 to scale.

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. Thus serialization and transmission take a considerable amount of time in our current implementation. While a delay of 20 seconds might be acceptable for a single user, longer interruptions with every newly added user in a multi-user environment would strongly decrease usability. Microsoft's *KinectSDK* would offer calibration-less tracking, but it provides no estimate for the confidence of the tracked joints. Thus 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 confidence values. We are therefore hoping for refined confidence values in Primesense's NITE implementation.

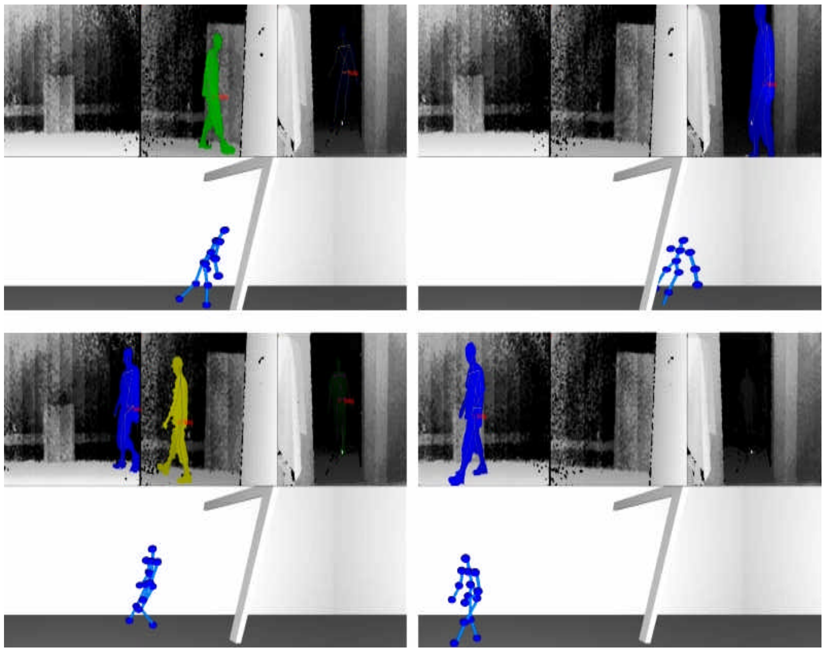

Figure 8: Snapshots of a tracked user from setup with orthogonal sensor placement in two rooms. Left side shows how multiple perspectives improve tracking. Right side shows the user in two different rooms.

REFERENCES

[1] Microsoft, KinectSDK for Windows. http://research.microsoft.com/enus/um/redmond/projects/kinectsdk/, last visited Sept. 2011.

[2] OpenNI, Natural Interaction Middleware. http://www.openni.org/, last visited Sept. 2011.

[3] S. Posland, *Ubiquitous Computing: Smart Devices, Environments and Interactions*: John Wiley & Sons, 2009.

[4] A. D. Wilson, and H. Benko, Combining multiple depth cameras and projectors for interactions on, above and between surfaces. Proceedings of ACM UIST '10, pp. 273-282, 2010.

[5] MOTION-ANALYSIS, Motion Analysis: Passive Optical Motion Capture. http://www.motionanalysis.com/, last visited Sept. 2011.

[6] Vicon, Vicon motion capture system. http://www.vicon.com, last visited Sept. 2011.

[7] XSense, Wireless Inertial Motion Capture. http://www.xsens.com/, last visited Sept. 2011.

[8] J. Deutscher, and I. Reid, “Articulated Body Motion Capture by Stochastic Search,” *International Journal of Computer Vision*, vol. 61, no. 2, pp. 185-205, February 2005.

[9] S. Knoop, S. Vacek, K. Steinbach *et al.*, Sensor fusion for model based 3D tracking. *In Proceedings of the Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*, pp. 524-529, Sept. 2006.

[10] C. Plagemann, V. Ganapathi, D. Koller *et al.*, Real-time identification and localization of body parts from depth images. *In Proceedings of the Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3108-3113, May 3-7, 2010.

[11] Asus, Xtion Pro Motion Sensing. http://event.asus.com/wavi/product/WAVI_Pro.aspx, last visited Sept. 2011.

[12] Microsoft, Kinect full body interaction. http://www.xbox.com/kinect, last visited Sept. 2011.

[13] Primesense, PrimeSensor 3D-sensing technology. http://www.primesense.com/, last visited Sept. 2011.

[14] J. Shotton, A. Fitzgibbon, M. Cook *et al.*, “Real-time human pose recognition in parts from single depth images,” in CVPR, 2011.

[15] H. Kawasaki, R. Furukawa, R. Sagawa *et al.*, Dynamic scene shape reconstruction using a single structured light pattern. *In Proceedings of the Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1-8, June 23-28, 2008.

[16] T. Weise, S. Bouaziz, H. Li *et al.*, Realtime Performance-based Facial Animation. *In Proceedings of the ACM Transactions on Graphics, SIGGRAPH 2011*.

[17] E. Suma, D. Krum, B. Lange *et al.*, “FAAST: The Flexible Action and Articulated Skeleton Toolkit. ,” in IEEE Virtual Reality, 2011.

[18] I. Schiller, MIP - MultiCameraCalibration. http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Calibration, last visited Sept. 2011.

[19] Boost, Boost C++ Libraries. http://www.boost.org/, last visited Sept. 2011.

[20] M. RUSSELL, T. C. Hudson, A. Seeger *et al.*, “VRPN: a device-independent, network-transparent VR peripheral system,” in ACM symposium on Virtual reality software and technology, Baniff, Alberta, Canada, pp. 55-61, 2001.

[21] G. Reitmayr, and D. Schmalstieg, “An open software architecture for virtual reality interaction”. *In Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 47-54, 2001.

[22] Unity-Tech., Unity3D game engine. http://unity3d.com/, last visited Sept. 2011.

[23] C. Schönauer, T. Pintaric, and H. Kaufmann, “Full body interaction for serious games in motor rehabilitation,” in Proceedings of the 2nd Augmented Human International Conference, Tokyo, Japan, pp. 1-8, 2011.

AUTHOR BIOGRAPHIES

Christian Schönauer is a research assistant and PhD candidate at the Interactive Media Systems group at Vienna University of Technology, Austria. After finishing his master thesis on “Skeletal Structure Generation for Optical Motion Capture” he received his MS in computer science in 2008 from Vienna University of Technology. He was involved in different research projects and corporations (e.g. European Union funded FP7 projects Paymancer and Profitex) and since September 2011 participates in a research collaboration with Massachussets Institute of Technology focusing on multimodal feedback technologies. His research interests include augmented and virtual reality, tracking, and especially full body Motion Capture and its application in serious games for medical purposes.

Email: schoenauer@ims.tuwien.ac.at

Hannes Kaufmann is associate professor at the Institute of Software Technology and Interactive Systems at Vienna University of Technology, and head of the VR group since 2005. He participated in projects and conducted research in the areas of virtual reality, tracking, mobile augmented reality, training spatial abilities in AR/VR, tangible interaction, medical VR/AR applications, real time ray-tracing, redirected walking, geometry and educational mathematics software. His Habilitation (2010) was on “Applications of Mixed Reality” with a major focus on educational mixed reality applications. He has acted on behalf of the European Commission as a project reviewer, participated in EU projects in FP5 and FP7, managed over 10 national research projects, and published more than 60 scientific papers.

Email: kaufmann@ims.tuwien.ac.at