FusionKit: A Generic Toolkit for Skeleton, Marker and Rigid-Body Tracking

Michael Rietzler[1] **Florian Geiselhart**[1] **Janek Thomas**[2] **Enrico Rukzio**[1]
[1] Institute of Media Informatics, Ulm University, Ulm, Germany
[2] Department of Statistics, University of Munich (LMU), Munich, Germany
[1] *<firstname>.<lastname>@uni-ulm.de,* [2] *janek.thomas@stat.uni-muenchen.de*

ABSTRACT

We present a toolkit for markerless skeleton tracking and marker-based object tracking utilizing data fusion with an arbitrary number of depth cameras. As depth-camera based skeletal tracking is always inaccurate due to technology limitations, our goal was to be able to preestimate systematic errors for given tracking situations to improve fusion.
Previous work analyzed various aspects of depth camera accuracy, however to our best knowledge, there has been neither systematic error modelling nor an application of such a model for skeletal fusion.
Our paper presents such a model for the Kinect v2 camera, by using statistical modelling on capture datasets using such cameras and a marker-based ground truth capture system. By applying this model, we are able to improve the overall accuracy of the fusion output by 68% by predicting data quality with an error of around 3.2 cm.
Our toolkit is available for use by other researchers to easily create larger capture spaces with higher tracking accuracy based on the error model when compared to single depth cameras.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g. HCI): Input devices and strategies; I.4.8 Scene Analysis: Motion; I.4.8 Scene Analysis: Sensor fusion

Author Keywords

Multi-depth camera fusion; Toolkit; Kinect error modelling

INTRODUCTION

In the area of human motion tracking, for a long time commercial motion capture systems were the only way of capturing skeletal movements for research and industry purposes. These systems are expensive and force the user to wear a special marker or sensor suit. However, with the release of consumer-grade, affordable depth cameras like the Microsoft Kinect in 2010, more and more applications began to utilize this low-cost body tracking devices for a large number of different use cases. Since these cameras are originally designed as consumer gaming devices, their short tracking range and volatile tracking quality limits the use in scenarios where large tracking spaces, occlusion robustness and constant quality are of importance, e.g. in VR or smart home environments as well as industrial tracking scenarios.

One way to overcome these limitations – soon presented by different researchers – is the use of multiple depth cameras to create an improved tracking system. While some works (e.g. [11]) showed that is possible to fuse the raw point clouds of the depth cameras, for a number of applications e.g. in human-computer interaction which depend on real-time data, it proved to be more useful to fuse the skeletal tracking data. This data is often already provided out-of-the box by the camera SDKs. We make such a system available to the manifold community of potential tracking system users by introducing a generic toolkit named *FusionKit* for skeleton, marker and rigid body tracking based on multi-depth-camera fusion. It utilizes an arbitrary number of depth sensors and is designed for, but not limited to human computer interaction use cases. Through its real-time capabilities, it is also suitable for delay-sensitive applications like virtual reality applications.

As the fusion of skeletal data is heavily dependent on the individual sensor performance and the actual fusion approach, we propose a statistical model of systematic errors for the Kinect v2 tracking device as well as an evaluation of the system performance, and show the dependency of the performance on this error model together with other factors like different data models.

The main contributions of our paper thus are:

1. A statistical model which allows to predict tracking errors with the Kinect v2 with high accuracy
2. A fusion system based on this model, using novel, skeleton-optimized fusion
3. A filtering mechanism for an Iterative Closest Point (ICP) using potentially error-prone skeletal data
4. An evaluation of the fusion system itself, showing large improvements in tracking accuracy with our approach

Besides this, the presented ready-to-use toolkit software is made available as open source for further use by researchers and other users to build, support or evaluate interactive software.

EICS'16, June 21 - 24, 2016, Brussels, Belgium

ACM 978-1-4503-4322-0/16/06...$15.00
DOI: http://dx.doi.org/10.1145/2933242.2933263

RELATED WORK

Various research has been done related to multi-depth sensor fusion for skeletal tracking, covering a variety of use cases and applications. However, the described approaches have been largely developed to fit a specific use case and are restricted in the amount of sensors or users. Furthermore, to the best of our knowledge, neither of the described systems were made available for further use, nor was there any in-depth evaluation of the systematic errors in the source data or the improvements accomplished by applying the particular methods proposed.

Kinect Accuracy

Yang et al. [22] analyzed the accuracy of the Kinect v2 for Windows and suggest improvement strategies using multiple sensors. They also provide measurements regarding the accuracy distribution which proved the depth accuracy being highly dependant on the distance of the tracked pixel as well as from its angle towards the sensors origin. They also suggest to overcome these limitations and to improve the depth accuracy by using multiple Kinect sensors and a trilateration method. Using multiple Kinect v1 sensors, the problem of interferences between multiple sensors has to be solved [5].

Regarding skeleton tracking, Wang et al. [20] gave a first insight of the accuracy, however restricting the angle between participant and sensor as well as the amount of tracked movements.

Multi-depth-camera Skeleton Tracking System Use Cases

Following, we first take a look at different use-case-centric multi-depth-camera implementations, which show the state of the art as well as the broad applicability of such approaches.

Regarding the fusion of skeletal data, earlier works e.g. by Caon et al. [4] use a low-complexity fusion approach for their smart-home-related application, based on Kinect v1 cameras. They propose a weighted-averaging approach for the fusion of joint coordinates, with the weighting factors being derived from the Kinect SDK tracking state, either on a per-joint basis or also considering the total number of joints being tracked by a camera for a certain user as an overall quality criterion. A similar fusion approach is presented by Schönauer and Kaufmann [19] for motion tracking.

Kaenchan et al. [13] use the fused results of multiple Kinects to analyze the walking posture of a human subject in order to check walking balance. For the fusion, the tracking state of the Kinect is used as quality indicator. The fused position is calculated as the average of all joints having the highest possible accuracy rating by the Kinect. Furthermore, they also feature an automatic, skeleton-based extrinsic registration mechanism and a small evaluation of the fusion results, however only with respect to the raw camera data without further ground truth. Multiple Kinects can also be used for action or activity recognition, as shown by Azis et al. [2], Haller et al. [9] or Hachaj et al. [8]. Using two perpendicular placed Kinects, Azis et al. define one as main sensor. If a joint is not tracked by the main sensor, it is substituted by the one tracked by the second one. Hachaj et al. could show an improvement for the activity classifier using multiple Kinect sensors with similar methodology.

Kitsikidis et al. [15] present a skeleton based ICP approach for sensor registration, which is calculated repeatedly out of single frame data. The result of the frame having the most tracked joints in both compared skeletons is used as the final result. The fusion is done by finding the skeleton with the highest number of tracked joints as basis and exchanging the positional information of untracked or badly tracked joints with the average of the tracked samples of other sensors. The fused and smoothed result is used for motion analysis for dancers.

Williamson et al. [21] use a multi-Kinect setup for dismounted soldier training. In this work, four Kinects are fused by two different approaches. For the rating of joints the SDK-provided joint status is used, but extended by using a depth-to-joint cross-validation, which allows to recognize occlusion-induced measurement errors by comparing the joints depth (Z) measurements to the raw point-cloud Z measurement at the same X-Y position. In the first fusion approach a weighted average over all joints with a value higher than a defined confidence value is used. The second approach is orientation based and works on the assumption that a soldier is carrying a gun which aims into the direction he wants to face. This direction is used to choose the most suitable Kinect at a time, with a hardcoded optimal range. The system was also evaluated, but without ground truth measurements.

Asteriadis et al. [1] use multiple Kinects to enhance motion recognition, using an advanced rating and fusion approach. For the extrinsic registration of the different sensors, they use the tracked torso joints over time as a point cloud which is analyzed to gather an optimal rotation and translation. The presented fusion method is based on a fuzzy inference approach, considering not only the SDK-based tracking states, but also different other quality parameters, like the depth-joint-correlation already presented by Williamson, and aspects like temporal continuity of motions, kinematic restrictions of the human body model or modeled sensor noise. The presented evaluation proves the performance of the method, but does not provide absolute accuracy measurements.

Otto et al. [18] propose a multi-Kinect system for ergonomic assessments. The neck joint is captured over time and used to calculate the extrinsic camera registration by an ICP algorithm. Additionally, the floor plane estimation of the Kinect SDK is used to refine the registration. They also include an approach for synchronization and interpolation of the sensor data, and a set of quality heuristics to rate the different sensor skeletons, like user angle, distance or position. The fused skeleton joints are calculated using the weighted middle based on the quality ratings. However, the joints are not weighted separately.

A broad overview of such use cases using Kinect sensors is given by Han et al in [10]. Besides that, Gillian and Paradiso [7] intruduce the *Gesture Recognition Toolkit* that was designed to enable even non-specialist users to work on real-time machine learning and gesture recognition. Munaro et al. [16] use the point cloud of a consumer depth sensor to create a 3D-model of a person for re-identification. Skeletal data is utilized to overcome the problem of comparing different poses.

They report on implausible poses and the limited tracking space of the Kinect and its skeleton tracker.

Other Multi-depth-camera Systems

Besides these works which mainly focus on skeletal tracking, there are some toolkits working with multiple depth cameras, but not including skeletal tracking. As the underlying technical challenges are partially the same, we also survey some of them in the following.

The *RoomAlive* [12] toolkit uses multiple Kinects for interactive projection mapping to augment any room using projectors. Though no skeletal data or fusion is used, interaction is accomplished using either controllers or the captured depth map and a real-time physics simulation. In latter case, moving objects or users are tracked using particle models and a depth-aware optical flow algorithm. *OpenPTrack* [17] is an open source software to track people using multiple networked depth sensors e.g. for surveillance scenarios. While not able to track the users' poses or skeletons, it is possible to track the positions of more than six people (compared to the Kinect) over an arbitrarily large area.

MOTIVATION

As visible from the related work, a range of different challenges needs to be addressed to perform fusion of skeletal tracking data. Many of them are not specific to skeletal tracking fusion, but also apply to a multitude of data fusion methods, and were previously defined e.g. in [14].

In our work, we particularly focus on two of those problems, which were not addressed in detail in previous work on skeletal fusion, but greatly influence the fusion result:

Systematic Tracking Errors

Real-world sensor data is always subject to imperfections e.g. due to noise or measurement errors. When using skeletal tracking on top of depth data, an additional source of error is introduced as the pose estimation mostly follows a probabilistic approach. We hypothesized that part of this error is systematic, i.e. depending on factors like the user's pose, position or rotation. Being able to predict the dimension of this error based on those factors enables the subsequent fusion method to pick the data which is more likely to be correct, thus leading to a better fusion result.

Fusion Data Models

While many of the already known skeletal fusion systems rely on simply averaging positions of body joints or even treat the skeletons as a whole, they also mostly do not respect the kinematic structure of the human body. Few approaches already consider kinematic constraints like invariant bone lengths or joint limits, however we assume that using a more kinematics-driven model for the fusion process would yield improved results over current models.

SKELETON TRACKING ACCURACY

When working with single depth cameras for skeleton tracking, the aforementioned imperfections related to tracking accuracy and range become quickly visible. Besides the low-level restrictions imposed by hardware and the optical system per se, we also observed a range of phenomena concerning the limitations of the Kinect SDK's skeletal tracking algorithms, which further deteriorated the overall tracking result. This included rather obvious factors like optical occlusion leading to badly tracked skeletal joints, but also more subtle parameters, like the user's position within the tracking space.

The earlier analysis of the accuracy of Kinect cameras (e.g. [20]) provide first insights on accuracy errors, but were only focused on restricted motions without 360° rotational freedom which is relevant to a fusion system with multiple cameras. Besides this, no further evaluation on how the different factors influence the overall accuracy were conducted. Therefore, we decided to perform a more in-depth evaluation of those factors as a first step to improve the overall fusion result.

Apparatus

We designed and performed a study which was aimed at uncovering systematic errors in the skeletal tracking output. This study is based on several captures using multiple Kinects and a ground truth, as described subsequently.

Sensor Setup

To collect a sufficient amount of data for the analysis, we placed multiple Kinect v2 sensors on tripods around a rectangular capture area faced towards the center. As a ground truth, we used a traditional, marker-based motion capture system based on 16 OptiTrack Flex 13 cameras surrounding the capture space (see also fig. 1). All capture instances were temporally synchronized using NTP with offsets not exceeding one millisecond. The Kinect positions were also tracked by the marker-based system to establish a common coordinate space to which each Kinect skeleton was transformed. The marker suit included 46 markers tracked with a mean residual < 1.0 mm.

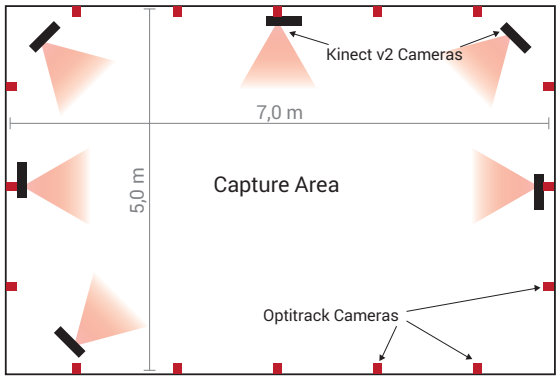

Figure 1. Kinect and Optitrack Setup with the approximate capture area

The skeleton tracked by the OptiTrack system was used as a reference skeleton. We then measured errors as the euclidean distance between the joint position in the respective Kinect's skeleton and the reference skeleton. Since the skeletal model slightly differs between the OptiTrack system and the Kinect, a constant retargeting offset was applied.

Participants' Movements

Using the described setup, we captured 20 minutes of movements of 5 participants (3 male, 2 female, age 24 - 28, skinny

to normal build), consisting of different motion types including frontal movements towards and away from one sensor, lateral movement at fixed distance (frontal to one sensor), walking a circular path, pointing with extended arm, arm rotations, bending arms and knees at fixed position and random natural movements across the capture space.

Overall, over 250.000 motion frames were captured, containing 25 joint positions in three-dimensional space, which served as a dataset for the analysis described in the following.

Kinect Accuracy Results

To obtain an overall view on the Kinect's accuracy, we first analyzed the whole dataset regarding its mean error, defined as the euclidian distance between the ground truth measurement and the sensors'. The mean regarding all joints showed to be around 12 cm while analyzing the natural movement task. A closer look by analyzing the mean deviation for each joint separately (Fig. 2) revealed that the error greatly differs throughout the various joints of the skeleton, ranging from values around 3 cm with the spine joints up to around 25 cm for the right wrist joint.

This is presumably a consequence of the user occluding parts of his own body when standing sideways to the sensor. If for example only one arm is visible to the camera, we observed that the left and right arm joints are mistaken to be at the same position. In addition, the Kinect SDK does not distinguish if a user faces the sensor or is turned around. The skeleton is mirrored in the latter case, which also leads to a sharp increase of the mean error values especially of distal joints.

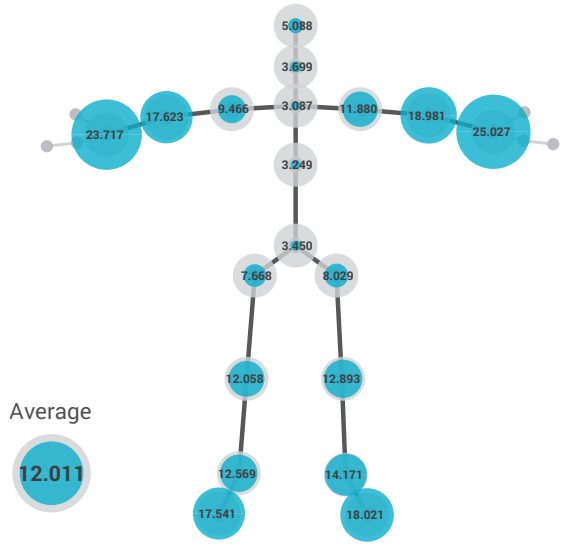

Figure 2. Mean error per joint (in cm) of one Kinect while performing random natural movements.

Error Prediction

A reliable prediction of errors based on the tracking situation is necessary for the fusion of data. Put simply it has to be decided which sensor can be trusted at a time.

Heuristics: Error Influencing Factors

Since the error cannot be predicted reliably only by using the SDK tracking states, we analyzed how the different factors like occlusion and self-occlusion, position and rotation contribute to the overall error. For this, we reviewed the captured motions with the goal of manually identifying key factors, which we further on call heuristics, which are likely to influence the tracking quality and also included the influence factors already found in the related work. Overall, we identified 8 of those heuristics for further analysis, which are described briefly below.

The Kinect SDK itself already provides coarse information about the joint's accuracy. This rating includes three states named *tracked, inferrend* and *untracked*. Joints marked as tracked are supposed to have an accurate position value. The accuracy of a joint marked as inferred ranges from values similar to the tracked state to deviations of more than one meter at the boundaries of the capture space during our analysis. Joints which are marked as untracked by the SDK do not provide any useful tracking data (e.g. zero coordinates), but did also not occur during our analysis in regular tracking situations, as the SDK always tries to keep on tracking all skeleton joints (marking them as inferred) until the skeleton is not tracked any more as a whole. These states are used as our first heuristic.

The second identified heuristic is the absolute distance of joints to the sensor. We observed that the accuracy of a joint is varying with the distance, having the maximum accuracy at around two and a half meters.

When getting closer or farther, the quality decreases until the skeleton detection fails due to mostly invisible body parts, missing depth data at greater distances or decreasing angular resolution with increasing distance.

The third and fourth additional heuristics consider the horizontal and vertical field of view (FoV) of a sensor. We observed a decrease in accuracy of joints when moving closer to the edge of the sensor's FoV in X or Y direction.

Due to self occlusion, we also suspect the user's rotation towards the sensor to be a contributing fifth heuristic for the overall error. For this, the angle between the user's viewing direction and the inverted direction of the sensor is calculated. In the manual analysis, we saw that with this angle exceeding ± 30 degrees, the resulting accuracy of joints that suffer occlusion (e.g. on the far side of the body) decreases rapidly.

The sixth heuristic considers spatial jitter of the joint position over time, by comparing the actual position measurement to an estimation based on previous velocity and acceleration. The difference between actual and estimated position is used as the heuristics' value.

As already shown by Wang et al. [20], the bone lengths vary over time, which we use as a seventh heuristic by calculating the difference to fixed, known values.

The last heuristic is based on the angle of one bone. We assume that the more perpendicular the bone is oriented relative to the camera's viewing direction the more precise is the tracking

accuracy, due to occlusion effects and the skeletal tracking algorithm itself.

Linear Regression Models

For the evaluation of the predicted errors we cross validated the data to prevent overfitting in all of our models. This means, that the validation of the model was done using different data as it was used to set up the model.

We first tried to predict an error based on the three states provided by the Kinect SDKs using a linear regression model. Since only one influence is used, an optimal (in this case mean error) is assigned to each state.

The predicted error differed by around 8.5 cm compared to the measured one. This offset is too high considering the mean measured error of around 12 cm.

Using a linear regression model we could show that each described heuristic has a significant influence on the 5% level and can therefore be used in a more detailed heuristic model. Using the full set of heuristics, we tested the error prediction again using a linear regression model. Though the prediction was improved, the measured offset between predicted and measured error is still around 6.4 cm in mean.

One drawback of the simple linear model is the potential prediction of negative errors as well as the fact that the error is long-tailed, meaning that most errors will be near zero but few can become very large. Therefore a second linear model was tested that assumes a gamma distribution of the error. In this case, the predicted errors are linked to the heuristics via an exponential function. We were able to reduce the difference between predicted and measured error by another 2 cm to around 4.4 cm in mean by using this model. As this result is mainly sufficient for centered joints, where the offset is only around 1.9 cm, the offset for the wrist joints is around 7.5 cm. We therefore also tested a non linear model.

Non-Linear Boosting Model

The last tested model was a non-linear gradient boosting model, based on a large ensemble of small regression trees, to estimate more complex dependencies [6].
Using this more complex model we were able to predict the error with a mean offset of around 3.2 cm. The errors of the spine shoulder and neck joint were predicted most accurate with an offset of around 1.2 cm while the wrists and feet suffered an offset of around 5.5 cm.

Explained and Systematic Error

The relative influence of the heuristics on the error was measured with the boosting model using variable importance [6]. Each influence can be interpreted as how much closer the predicted error gets to the measured one when a heuristic is included to the model. Therefore the explained error is a sum of all influences which can be calculated for each joint. The explained error is systematic and predictable by our set of heuristics, while the unexplained error is either not systematic or arises by a not tested heuristic.

The explained error varied between 31% for the wrist and 80% for the neck joint. All influences are summarized in figure 3. For 23 of 25 joints the orientation of the tracked person has the highest influence (34% to 74%) on the explained error. Only tracking of the head and base of spine are almost independent of the person's viewing direction. Both distance and the vertical displacement have high influence as well, respectively between 5% to 25% and 8% to 30%.
A reason why the error for example of the wrist joints could not be predicted as accurate as the ones for the more centered joints can be found in the nature of the Kinect skeleton tracker. Each joint that is not tracked is only inferred. This guess about the position varies the most for limb joints which leads to unpredictable errors.

The great impact of the distance and the horizontal and vertical periphery can be explained by the results of Yang et al. [22] who described the optimal tracking area with a limited distance and field of view. Other features, like the user or bone angle result from the nature of optical devices e.g. body parts that occlude others.

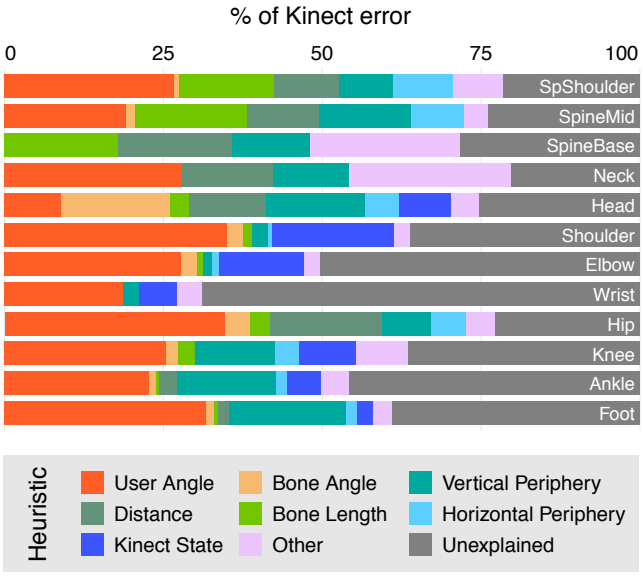

Figure 3. Influence of the different heuristics on the tracking error according to the boosting model.

THE FUSIONKIT

With the results of the first analysis we designed and implemented a system which is able to improve the tracking accuracy as well as to enlarge the tracking space by fusing the data from multiple depth sensors. Besides the prediction of errors there are further challenges when building such a system like extrinsic registration, temporal synchronization and interpolation, and the fusion itself. With our *FusionKit* implementation, we strive to provide a stable, reliable and usabe solution for all of those challenges by integrating common but also novel methods to deal with those challenges.

On the hardware side, we chose to focus on Kinect v2 sensors because of their ubiquitous availability and tracking performance, but the system architecture is designed to also accommodate to other types of depth cameras. The system itself is designed as a distributed system, on the one hand because the Kinect SDK only allows one camera to be used per PC, but on the other hand also to be able to scale more easily to a large

number of cameras, e.g. over a large area by using existing networking infrastructure. This also balances the calculation load, as the costly tasks like skeleton tracking and image segmentation can be done on the respective node, while the main fusion node only needs to handle preprocessed data.

The main processing pipeline used by the fusion node in principle uses a similar strategy for skeleton and marker data as shown in figure 4 and is described in more detail in the following sections.

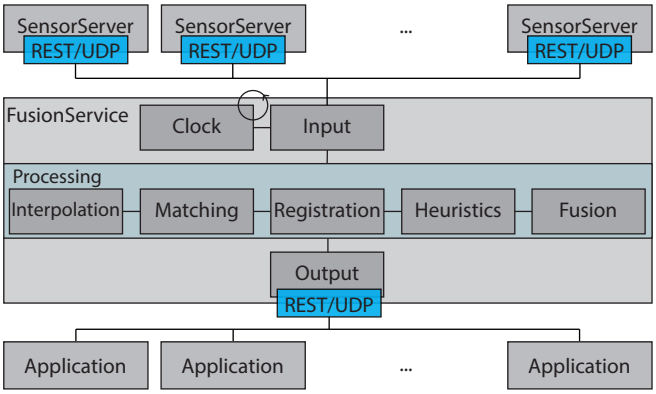

Figure 4. The FusionKit workflow: The sensor data of all connected sensors is forwarded to the processing pipeline in defined time steps. The fused data is stored in the output module and can be collected by different applications.

Data Capturing

Data capturing is part of the node software, which is called SensorServer. For the skeleton tracking we use the Kinect SDK tracker while marker tracking runs on the infrared and depth image.

Skeleton Tracking

The Kinect SDK provides an own skeleton tracker based on the depth image. The human skeleton is represented by a set of 25 joints, like neck, head or wrists, each having a 3D position as well as an orientation in some cases. In each frame, the depth image is used by the SDK to perform a per-pixel body part classification and thereafter a presumption about the joints. Finally, prior knowledge and temporal continuity are used to map the joints to fit the skeleton and its joints [24].

Marker Tracking

Besides tracking of skeletal data, it is often useful to be able to track arbitrary objects in 3D space. With common, 2D-based motion capture systems, this is mostly accomplished using either active (IR-emitting) or passive (IR-retroreflective) spherical markers, which are tracked with multiple 2D infrared cameras. The 3D position is then reconstructed by solving a multiple-view geometry problem based on the 2D positions and the known 3D registrations of the cameras.

With the Kinect it is also possible to capture infrared imagery which can be used for marker tracking. We provide such a feature in the SensorServer part of our toolkit, mainly to enable object tracking and interaction, which is especially useful in e.g. VR applications to enable the representation of real objects in virtual space.

The Kinect captures infrared images at a resolution of 512x424 pixels with 13 bits of dynamic range. While this imagery is mainly used within the camera to generate the depth image, it can also be retrieved separately for further processing. In our system, the IR images are retrieved and segmented after some preprocessing like amplification and thresholding to find possible markers in the image and their respective center x and y pixel coordinates using OpenCV. Additionally, we query the depth data generated from the IR image, to gather z values for the area of the image where possible markers are assumed to be located.

Unfortunately, we found that the markers themselves suffer from overexposure in the infrared image, which, while simplifying the task of finding 2D positions, prevents the camera's ToF algorithms from computing a correct depth estimation. However, through analysis and measurements of the IR and depth marker images, it was found that usable depth values can be gathered from the area which is directly surrounding the marker. This "corona" (see figure 5) was found to reliably deliver correct depth values for the marker z coordinate. By sampling this area, a list of possible values is acquired and post-processed to gather a single z value for a suspected marker.

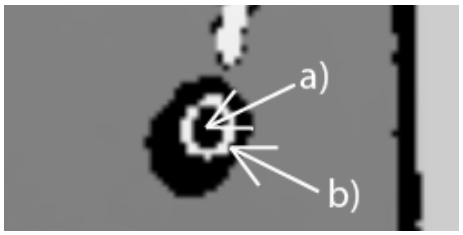

Figure 5. Coronary features (b) around markers (a) observed in the Kinects depth data

Using the pixel's x and y as well as the z values, it is possible to calculate the position in metric 3D space in the camera's coordinate system using the Kinect SDK's internal coordinate mapper features. Based on this metric values, further validation steps like examining the real-world size of markers can be carried out. To allow for an easy use during common tracking situations, we also implemented masking methods common with traditional motion capture systems, which provide automatic and manual masking possibilites to eliminate stray markers and reflective areas in the capture space. Finally, the marker tracking data is provided to the fusion core in real-time using the same interfaces like the skeletal tracking data.

Data Transmission

The system components provide basic interfaces, which are accessible via REST, WCF and UDP, for polling skeletons or markers, to start and stop a capture or to change a setting. In case of data polling, the returned data consists of a list of joints containing position data as 3D vectors, as well as orientational data as quaternions and a assumed error value in centimeters gathered by our heuristics. The marker information only contains a 3D position and an ID for re-identification.

Data Modelling

The Kinect skeleton is defined by a collection of joints that can be located by a 3D position vector. The vectors' origin is the zero position in the sensor's coordinate system. This kind of view on the skeleton will further on be called *absolute skeleton* since it relies on absolute positions of the joints. We also use a different view on the skeleton that is further on called *relative skeleton*, which is based on the definition of a kinematic chain. Such a model is already used for e.g. character animation, but not for processing or fusion. A kinematic chain is a graph with joints as nodes. Therefore the edges can be interpreted as bones. In the relative skeleton, the root is the only joint with an absolute position information. The relative position P_r of every other joint j is defined by the direction vector between the previous joint j_{-1} and j ($P_r = P_j - P_{j_{-1}}$). Each position information of the joints (except the main joint) can be interpreted as a bone with a certain direction and length (see figure 6). We use this relative data representation extensively in user and registration handling, as well as for the fusion itself. A more detailed description follows in the respective sections.

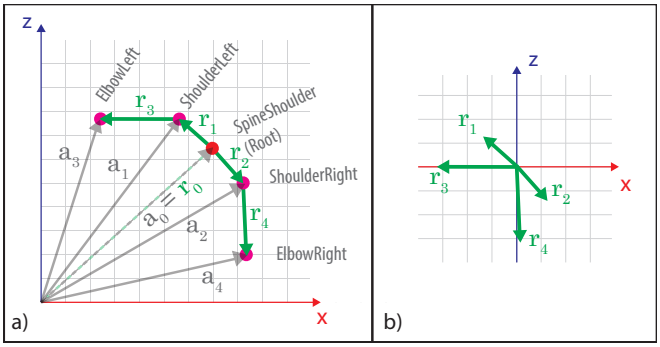

Figure 6. a) Example of a four joints in the absolute skeleton (a_1 ... a_5) and the relative direction vectors (r_1 ... r_4), a_0 remains the same for the relative skeleton. b) The relative direction vectors ignoring the kinematic chain.

Data Processing

In this section the main functionality of the skeleton and marker fusion process is described. Our application uses a multi-threaded pipeline architecture in which components can be chained to each other, to create a customizable dataflow with optimized skeleton and marker tracking for specific use cases. We also provide an optimized implementation of such a dataflow including a UI, called Studio which can be used out of the box.

Synchronization

Due to the nature of a distributed system using different computers attached to different sensors, temporal aspects are an important factor. Regarding synchronization it is likely that time stamps of incoming data vary due to clock offsets and network latency. Therefore each node has to be synchronized to a common clock which is provided by the fusion service. Each node provides a REST interface for polling the current time in ticks. The timestamps are polled every 10 seconds to calculate and save the offset to the local time, accounting the half round trip time (RTT).

Interpolation

When performing faster movements (like waving a hand with around 3 m/s) the data provided by the sensors may vary by around 10 cm within the duration of one frame (with a Kinect capturing at 30 fps). Though the data is synchronized, the capture timestamps still differ. Therefore the data from different sensors has to be interpolated to a common time to accommodate for this possible offset. Therefor, all frames are interpolated in a linear way using a configurable constant delay applied to the actual time. The frames before and after this time slot are used for interpolation to calculate the position. The interpolated position is computed under the assumption of a linear velocity between the past position and the future one. If the offset is smaller than the frametime, the state of this frame has to be predicted. In this case an linear extrapolation is done for predicting the position. To be still able to work in real-time the delay is set to 35ms by default but can be increased (e.g. for standard motion capturing) or decreased (e.g. for VR applications). Interpolation also enables a variable frame rate. This allows the system to run at higher frame rates than the Kinect default of 30 fps.

Filtering and Optimization

The FusionKit also offers a variety of improvent strategies to optimize the tracked skeletons and markers as well as the fused results. A Kalman filter can be used to smooth the input and output skeletons or markers.

The predicted error that is used to calculate weights for the fusion can also be Kalman filtered. Though decreasing the accuracy of the prediction, the Kalman filter smoothes the fused motions by prohibiting fast changes of the weights. If cameras are set up in a circle, skeletons that are tracked from behind can be removed since they are the less reliable ones. In addition some assumptions about the human skeleton are used to optimize the positions of joints. For example we correct the bone lengths and the shoulder positions, which should always be in roughly aligned with the spine shoulder joint. It is also possible to enable a limit for the velocity of joints as a simple kinematic restriction.

User and Marker Handling

The user handling (or correspondence finding between users from different sensors) is a critical part of skeleton fusion since a wrong mapping of a sensor skeleton or marker to a user or marker object would lead to errors in the registration and in the fusion itself. There are two different system states that have to be considered separately. In a fully registered system every sensors' data can be transformed into a unified coordinate system and can therefore be compared directly based on the registered position, using some amount of threshold (defaulting to 10 cm for skeletons). If the registration of a sensor is unknown, a different approach has to be chosen. Since a usable fusion is impossible in a unregistered system, a simple one user scenario is active as long as no registration was calculated. If one sensor detects more than one skeletons at one time, both skeletons are ignored.

An alternative approach matches users without a registration and enables multi-user handling for registration (but not fusion). It considers the users pose and body instead of the not

available global position and allows user handling in an unregistered system. Two features are used to decide whether two skeletons belong to the same user. The first one compares the lengths of the bones. The second criterium for user mapping is the user's pose. For comparing the pose the relative skeleton (excluding the main joint) is interpreted as a point cloud to compute the rotation and translation between them. Since the direction vectors of the relative skeleton's joints do not consist of absolute positions, the translation of the point clouds has to be 0 using a fully accurate tracking device. Using the Kinect as tracking device, the translation can be interpreted as an error which has to be under a proposed threshold of 2 cm. This principle is described in more detail in the following section. This second approach allows for example to perform multi-user registrations, although with the downside of much longer durations since the unregistered multi-user approach is more restrictive. As it is usually possible to perform a single-user registration, we use this approach to user handling as the default method due to better results and shorter registration durations.

Similar to the correspondence finding for skeletons, markers have to be mapped to a unique marker, too, which is done by comparing their registered positions.

Extrinsic Registration

In order to calculate the extrinsic registrations between the sensors, one sensor is defined as main sensor. The main sensor provides the globally used coordinate system to which each other sensor is registered. A modified ICP approach is used, which uses the tracked skeleton over time as point clouds to calculate the relative rotation and translation iteratively. Since correspondence finding is not necessary due to the already established relation given by the joint and the missing scale factor, the algorithm can be simplified. The new registrations are updated in background after defined time steps. Using the human skeleton supersedes the use of an calibration object, but has drawback of using inherent inaccurate data of the Kinect skeleton tracker which greatly depends on the tracking situation.

Different approaches have been proposed in the related work to overcome this problem. The first one is to use only the best tracked joint [1]. Since the used torso joint (from OpenNI) is most of the time only moving in two dimensions, this approach only provides accurate rotations around the x and y axis. The second approach [18] includes the estimation of the groundplane in addition to the best joint. Since the Kinect groundplane estimation is not always reliable or sometimes even unavailable, we based our registration process on multiple joints. As this also includes inaccurate joint data, we developed a filter to remove badly tracked joints before they are added to the point cloud.

In the following we define a point cloud as a set of tuples with the first items (P) including the data of sensor 1 (S1) and the second items (Q) the matching data of sensor 2 (S2). Each rotation and translation pair is computed by an algorithm that minimizes the euclidean distance of P and Q [3]. To find the rotation and translation the covariance matrix H is computed (see equation 1). The rotation (R) can be computed by multiplying the matrix V^* by U transposed (U^T) using the Singular Value Decomposition of H ($H = U\Sigma V^*$). The 3D vector of the translation (T) is computed as shown by multiplying R with the centroid of P and adding the centroid of Q.

$$H = \sum_{i=0}^{n} (a_i - centroid_A) \cdot (b_i - centroid_B) \qquad (1)$$

The whole registration process is split in two steps. The first one is done in each frame, where the data of each sensor pair is compared and filtered to add the result to the registration point cloud of the respective sensor pair. The second step is repeated in certain intervals and follows the principle of a simplified ICP with the result beeing the relative rotation and translation of the sensors.

For the filter, the data of each sensor pair is added to a point cloud as relative bone vectors (excluding the main joint). The optimal rotation and translation are used to filter the data. Since no absolute data can be found in the point clouds the translation should be zero and its magnitude can be interpreted as an error value which can arise by comparing different poses or bone lengths which are both caused by tracking errors (or by comparing different users, which should not be the case during a single user registration or a working user handling). If the translation's magnitude is too high, the matching joints with the maximal euclidean distance after applying the computed optimal rotation are filtered and a new registration is calculated using the remaining joints. This iterative approach is illustrated in figure 7. The remaining joints are added to the registration point cloud using their absolute position. The simplified ICP runs after defined time frames. The current registration is used in every further step for filtering the point cloud based on the euclidian distance of two matching positions after applying the registration.

If no registration can be calculated between a sensor (S) and the main sensor (M) – e.g. due to no overlapping tracking space, the registration handling tries to register S to any other sensor that is already registered to M. As soon as the rotation and translation between S and a registered Sensor is known, both values are transformed by the already known registrations. Using this mechanism we ensure to register sensors even if they have no overlapping space to the main sensor.

Beside the skeleton based point cloud, the registration can also be done capturing one marker over time. Only one marker is allowed at the same time since the correspondence finding for markers is only working in a fully registered state, which makes this variant very sensitive to erroneous marker data, although potentially delivering more precise registrations.

Fusion

The fusion of skeletal sensor data is a two step process. In the first step the joints of each skeleton are rated by our heuristics. The second part is the fusion itself.

Heuristic Models: Weighing Joints and Markers

We implemented the best performing two heuristic models as described in our heuristic evaluation, the linear model using a gamma distribution and the non linear gradiant boosting model.

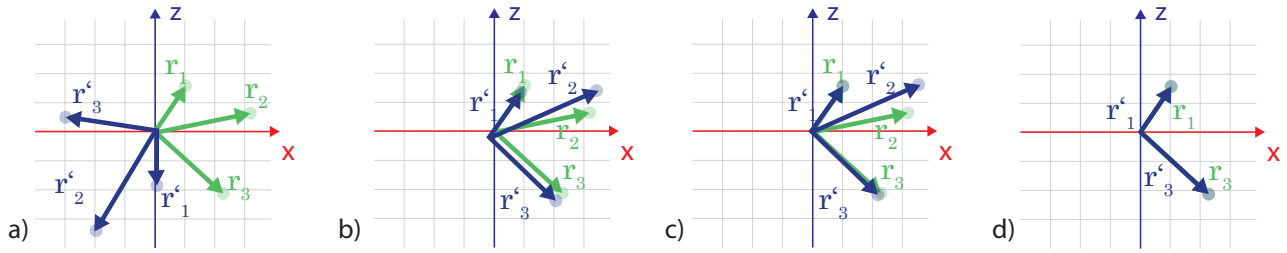

Figure 7. Illustration of the relative skeleton based filter. a) Reduced example of a relative skeleton using three points. b) Finding and applying the optimal rotation and translation – the origin of the blue skeleton is moved from the center which indicates an error. c) The blue skeleton is only rotated. Obviously r_2 and r'_2 don't match and are removed from the point cloud. d) The point clouds can be mapped without or minimal translation. The remaining joints would be added to the registration point cloud.

Each heuristic model predicts an error value for each joint of each skeleton of each sensor seperately. All errors regarding the same joint of all matching skeletons are normalized as weights for the fusion step.

Data Models

We implemented the described data models for the fusion. The *joint model* is based on the absolute skeleton and is described by absolute position data. We propose to use a *bone model* (based on the relative skeleton) for fusion. We also implemented an improvement strategy based on previous knowledge and the strict assumption about the human skeleton that the length of a bone may not vary over time, as already proposed by Yeung et al.[23].

Fusion Approaches

We implemented three different vector based fusion approaches inspired by the related work which handle both data models. The first is to take the best rated joint or bone as the final result (called *best*). The second uses a *weighted middle* of all available data. The third approach is to exclude *outliers* from this average. The predicted error is used to filter all joints with a higher error than the measured median. In a second step, outliers regarding the variances of the distance to the weighted middled position of a joint or bone are ignored for the fusion.

The elemination of outliers follows two steps – the first considering only predicted error values and the second considering positions. All joints having a higher predicted error than a threshold based on the average are excluded first. In a second step the predicted error values of the remaining joints are normalized and inverted as weights (w). The weighted middle (see equation 2) of all remaining joints is used for the second step. All joints with a higher euclidean distance to the middle than the average of all remaining joints are further excluded.

For each weighted middle approach the fused vector (V_f) using the not excluded joint vectors ($V_1 \dots V_n$) is computed according to equation 2.

$$V_f = \sum_{i=0}^{n} w_n \cdot V_i \qquad (2)$$

The fusion of marker data can only be done using the absolute marker position which is therefore the only model in use.

Rigid Body Tracking

A rigid body consists of n (where $n > 2$) markers (M), each of them having $n-1$ connections (C) to the other markers. The distance of each connection c in C has to be different to each other connection. Therefore a rigid body marker is distinctly defined by $n-1$ connections.

The euclidean distances between each marker is calculated every frame and matched to one or more connections, using a fixed threshold while the deviation is remembered as a confidence value. The result is a number of tuples consisting of a connection and a sensor marker. In an optimal case only one sensor marker would fulfill the condition of matching the $n-1$ connections defined by the rigid body marker. If there are more than one matching markers, the confidence value is used for the final decision.

Data Output

The FusionKit provides several output modalities for different use cases. The REST interface can be used for infrequent requests while the UDP streaming and the WCF interface were designed for continuous tracking. These flexible output modalities allow an arbitrary amount of clients and easy connection of other systems. If no live data is required, it is also possible to capture whole sequences either by a remote call to the REST interface or by the Fusion Service UI itself, which can be saved to different file formats afterwards. At the moment, we support the common motion capture data file format Biovision Hierarchy (BVH) as well as an own file format being a JSON representation of the FusionKit's internal data model.

SYSTEM EVALUATION

To evaluate the accuracy of our toolkit, we conducted a second study which was split in two steps. The first one was to find the optimal settings and the second to evaluate the accuracy of the FusionKit itself.

Apparatus

The sensor setup was the same as shown in figure 1. Three participants (2 male, 1 female, age 24 - 27, height 1.69m - 1.88m, skinny to normal build) were instructed to perform random natural movements within the tracking area of the multi-Kinect and OptiTrack system. This rather unrestricted random task instruction was chosen since the performance should be evaluated not under consideration of any specific use case, but as representative as possible for any kind of movements.

The participants mostly performed common movements like walking, moving arms and pointing, as well as jumping or kneeling. Each recording lasted for about two minutes and therefore consisted of roughly 3600 frames and thus samples per joint from the Kinects (capturing at 30 fps), and about 14400 frames / samples for the Optitrack system (capturing at 120 fps). Each provided error value is interpreted as the euclidean distance between the FusionKit joint and the OptiTrack one.

Feature Evaluation

Some of the features developed for the FusionKit provide alternative implementations. The heuristics for example were on one hand designed using a linear gamma distributed model and a non-linear boosting model both using our full set of heuristics. We also implemented two data models and three different fusion approaches. To optimize our system we compared the different alternatives and measured the accuracy of the FusionKit using optimal settings.

Heuristics

We compared the influence of the error prediction on the fusion results using the two heuristic models. All calculations were done using the bone model with outlier fusion. While the error prediction of the boosting model performed around 0.5 cm better then the linear one, the error after the fusion was reduced by 0.8 cm. The joints that were optimized most using the boosting model were the knee joints (around 1.4 cm) and the hips (around 1.3 cm). The overall error using the boosting model was 12.4% less then using the linear model.

Fusion

We analyzed the accuracy of the two data models (joint and bone model) that were fused via three different fusion approaches (best, weighted, outlier). Consequently 6 different approaches are compared in their mean and per joint error. Tukey post-hoc tests were used for further validation. Each significance is stated according to the 5% level.

Every fusion approach was significantly more accurate than the Kinect with the least error. The weighted fusion turned out to be the worst one – no matter which data model was used. Though the outlier approach was little more accurate than the take best semantic, the differences were not significant.

The bone model was the significantly most accurate data model, independent of the fusion approach with a mean error of around 2.6 cm compared to 3.3 cm using the joint model when fused with the outlier approach.

Considering the accuracy by joint, the bone model improved the accuracy of every joint except the head, wrists and elbows significantly. The largest improvement could be observed on the knee joints (left: 2.4 cm and right: 2.3 cm), followed by the feet (both 2.1 cm) and ankles (left: 1.0 cm and right: 0.9 cm).

System

Since the bone model proved to be the most accurate one, reducing the error by 21% compared to the joint model we used this model for the final system. The model is fused by the outlier fusion approach, which was, though not significant, a little more accurate than the take best approach. Each joint is weighted based on the error predicted by the non linear boosting model.

	FusionKit		Tracked Joints	
Joint	**Error (cm)**	**SD (cm)**	**Error(cm)**	**SD (cm)**
Head	2.66	2.65	2.79	2.46
Neck	1.43	1.94	2.795	2.61
Spine Shoulder	1.28	1.71	2.04	2.59
Shoulder l/r	1.27	1.90	5.51	4.33
Elbow l/r	3.32	2.31	12.70	8.95
Wrist l/r	4,37	2.64	16.13	14.51
Spine Mid	1.06	1.92	2.37	2.40
Spine Base	1.90	1.74	1.78	2.52
Hip l/r	2.54	2.19	6.13	2.93
Knee l/r	3.04	2.14	9.67	4.49
Ankle l/r	2.74	2.74	9.76	6.12
Foot l/r	3.60	2.59	13.87	7.65
Mean	**2.64**	**2.41**	**8.38**	**8.41**

Table 1. Mean error per joint of the fused skeleton with the best performing settings of the FusionKit and using only tracked joints.

We compared the best performing settings to two reference measurements, the first being the best performing single Kinect, and the second being a simplified fusion approach using the mean position of all joints having the *tracked* state. Compared to the best performing single Kinect, the error was reduced by 9.4 cm in mean (around 78%) and compared to the tracked joints by 5.7 cm (around 68%). The error per joint of the FusionKit result and the tracked joints is illustrated in table 1.

As our results indicate, the accuracy of fused skeletal data heavily depends on different factors, most of all by the weighting strategy and the used model.

CONCLUSION

In this paper we presented an analysis of factors influencing the Kinect v2 tracking accuracy. With these results, we were able to create different statistical models which allow to predict the magnitude of tracking errors based on the tracking situation. The offset of our predicted error and the measured one was around 3.2 cm in mean. By applying these models to multi-Kinect fusion we were able to build a toolkit named FusionKit. The proposed toolkit realizes fusion of multiple depth-cameras for marker-less skeleton as well as marker and rigid-body tracking.

Using our accurate prediction of errors using a statistical non linear boosting model including a large set of heuristics, as well as a data model following kinematic principles, we could reduce the error compared to a single Kinect by 78% resulting in a mean error of around 2.6 cm.

Our architecture allows to process data in real-time at flexible framerates and is therefore suitable for a variety of HCI applications like VR, gestural interaction or real-time motion capturing.

We hope that through the release of this software as an open source toolkit, a lot of users are enabled to use motion tracking technology for their work in diverse scientific and industrial use cases, without having to implement custom depth camera solutions on their own.

REFERENCES

1. Stylianos Asteriadis, Anargyros Chatzitofis, Dimitrios Zarpalas, Dimitrios S. Alexiadis, and Petros Daras. 2013. Estimating Human Motion from Multiple Kinect Sensors. In *Proceedings of the 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications (MIRAGE '13)*. ACM, New York, NY, USA, 3:1–3:6.
2. Nur Aziza Azis, Ho-Jin Choi, and Youssef Iraqi. 2015. Substitutive skeleton fusion for human action recognition. In *2015 International Conference on Big Data and Smart Computing (BigComp)*. 170–177.
3. Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Robotics-DL tentative*. International Society for Optics and Photonics, 586–606.
4. Maurizio Caon, Yong Yue, Julien Tscherrig, Elena Mugellini, and Omar Abou Khaled. 2011. Context-Aware 3D Gesture Interaction Based on Multiple Kinects. In *Proceedings of The First International Conference on Ambient Computing, Applications, Services and Technologies, AMBIENT*. 7–12.
5. Florian Faion, Simon Friedberger, Antonio Zea, and Uwe D Hanebeck. 2012. Intelligent sensor-scheduling for multi-kinect-tracking. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 3993–3999.
6. Jerome H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.
7. Nicholas Gillian and Joseph A Paradiso. 2014. The gesture recognition toolkit. *The Journal of Machine Learning Research* 15, 1 (2014), 3483–3487.
8. T. Hachaj, M.R. Ogiela, and M. Piekarczyk. 2013. Dependence of Kinect sensors number and position on gestures recognition with Gesture Description Language semantic classifier. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*. 571–575.
9. Emanuela Haller, Georgiana Scarlat, Irina Mocanu, and Mihai Trăscău. 2013. Human Activity Recognition Based on Multiple Kinects. In *Evaluating AAL Systems Through Competitive Benchmarking*. Communications in Computer and Information Science, Vol. 386. Springer Berlin Heidelberg, 48–59.
10. Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. 2013. Enhanced computer vision with microsoft kinect sensor: A review. *Cybernetics, IEEE Transactions on* 43, 5 (2013), 1318–1334.
11. Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera *(UIST '11)*. ACM, New York, NY, USA, 559–568.
12. Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. 2014. Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 637–644.
13. S. Kaenchan, P. Mongkolnam, B. Watanapa, and S. Sathienpong. 2013. Automatic multiple Kinect cameras setting for simple walking posture analysis. In *Computer Science and Engineering Conference (ICSEC), 2013 International*. 245–249.
14. Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. 2013. Multisensor Data Fusion: A Review of the State-of-the-art. 14, 1 (2013), 28–44.
15. Alexandros Kitsikidis, Kosmas Dimitropoulos, Stella Douka, and Nikos Grammalidis. 2014. Dance analysis using multiple kinect sensors. *VISAPP2014* (2014).
16. Matteo Munaro, Alberto Basso, Andrea Fossati, Luc Van Gool, and Emanuele Menegatti. 2014. 3D reconstruction of freely moving persons for re-identification with a depth sensor. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 4512–4519.
17. Matteo Munaro, Filippo Basso, and Emanuele Menegatti. 2016. OpenPTrack: Open source multi-camera calibration and people tracking for RGB-D camera networks. *Robotics and Autonomous Systems* 75 (2016), 525–538.
18. Michael Otto, Philipp Agethen, Florian Geiselhart, and Enrico Rukzio. 2015. Towards ubiquitous tracking: Presenting a scalable, markerless tracking approach using multiple depth cameras, In In Proc. of EuroVR 2015. *In Proc. of EuroVR 2015 (European Association for Virtual Reality and Augmented Reality), Best Industrial Paper award* (2015).
19. Christian Schönauer and Hannes Kaufmann. 2011. Wide Area Motion Tracking Using Consumer Hardware. In *ACM Advances in Computer Entertainment Technology conference (ACE 2011), Lisbon, Portugal*, Vol. 8.
20. Qifei Wang, G. Kurillo, F. Ofli, and R. Bajcsy. 2015. Evaluation of Pose Tracking Accuracy in the First and Second Generations of Microsoft Kinect. In *Healthcare Informatics (ICHI), 2015 International Conference on*. 380–389.
21. B. Williamson, J. LaViola, Tim Roberts, and Pat Garrity. 2012. Multi-kinect tracking for dismounted soldier training. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*. 1727–1735.
22. Lin Yang, Longyu Zhang, Haiwei Dong, Abdulhameed Alelaiwi, and Abdulmotaleb El Saddik. 2015. Evaluating and improving the depth accuracy of Kinect for Windows v2. *Sensors Journal, IEEE* 15, 8 (2015), 4275–4285.

23. Kwok-Yun Yeung, Tsz-Ho Kwok, and Charlie CL Wang. 2013. Improved Skeleton Tracking by Duplex Kinects: A Practical Approach for Real-Time Applications. *Journal of Computing and Information Science in Engineering* 13, 4 (2013), 041007.

24. Zhengyou Zhang. 2012. Microsoft Kinect Sensor and Its Effect. *MultiMedia, IEEE* 19, 2 (Feb 2012), 4–10.