International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2017)
R. Lindeman, G. Bruder, and D. Iwai (Editors)

Real-time Ambient Fusion of Commodity Tracking Systems for Virtual Reality

J. Fountain[1] and S. P. Smith[2]

School of Electrical Engineering and Computing, The University of Newcastle, Australia
[1]jake.fountain@uon.edu.au, [2]shamus.smith@newcastle.edu.au

Abstract

Cross-compatibility of virtual reality devices is limited by the difficulty of alignment and fusion of data between systems. In this paper, a plugin for ambiently aligning the reference frames of virtual reality tracking systems is presented. The core contribution consists of a procedure for ambient calibration. The procedure describes ambient behaviors for data gathering, system calibration and fault detection. Data is ambiently collected from in-application self-directed movements, and calibration is automatically performed between dependent sensor systems. Sensor fusion is then performed by taking the most accurate data for a given body part amongst all systems. The procedure was applied to aligning a Kinect v2 with an HTC Vive and an Oculus Rift in a variety of common virtual reality scenarios. The results were compared to alignment performed with a gold standard OptiTrack motion capture system. Typical results were 20cm and 4° of error compared to the ground truth, which compares favorably with the accepted accuracy of the Kinect v2. Data collection for full calibration took on average 13 seconds of in-application, self-directed movement. This work represents an essential development towards plug-and-play sensor fusion for virtual reality technology.

CCS Concepts

•***Computing methodologies*** → ***Tracking;*** *Camera calibration;* •***Computer systems organization*** → *Real-time system architecture;* •***Software and its engineering*** → ***Software libraries and repositories;***

1. Motivation and Problem Definition

The rapidly improving quality and availability of head mounted displays (HMDs) has seen a large increase in public interest in virtual reality (VR). HMDs allow for a wide variety of immersive experiences unavailable to other display platforms. However, HMDs typically block out the real world, leaving the user feeling disembodied in the virtual space. Without tracking of the user's body and hands, interactions within the environment are limited. Furthermore, without visual body and hand representation within the virtual environment, states of presence and immersion are impeded [CB16, SNB*17]. If the virtual environment is multi-user, communication can also be impeded by inaccurate or limited body or hand tracking [GWFM17].

Tracking systems for achieving high fidelity body and hand tracking are prohibitively expensive. For example, OptiTrack, Vicon, and similar gold-standard motion capture systems can have costs in the range of thousands to hundreds of thousands of US dollars. Many low cost devices exist as alternatives, but often do not provide the required tracking quality and range for many purposes. Some examples include Leap Motion, Microsoft Kinect, Perception Neuron, Oculus Rift, HTC Vive and PlayStation VR. These devices cost from one hundred to a few thousand US dollars, but suffer from issues such as limited tracking volume, occlusion, drift and low accuracy (for example, see [RGTR16] for details on the Kinect v2). This paper presents work towards creating highly accessible fusion software which allows for the combination of low cost tracking systems. The objective is to bridge the quality gap between commodity and gold-standard tracking systems. By minimizing the need for user configuration, the software aims to decrease required technical expertise and increase access to high-quality tracking for businesses, research laboratories and hobbyists.

The contributions of this paper revolve around the design of a state machine for ambient calibration, including:

1. A method for managing calibration data with minimal computational overhead
2. A method for determining calibration tasks between tracking systems with dependent sensors
3. Stable fault detection for when a calibration no longer describes the setup accurately

An evaluation was performed on the task of aligning the skeleton tracked by a Microsoft Kinect v2 with a user's true body pose for avatar representation and interaction within a VR system. The term 'VR system' will be used in this paper to mean a HMD with tracked controllers for each hand such that each of the head and hands is

DOI: 10.2312/egve.20171331

tracked with 6 degrees of freedom (6DoF; position and rotation). The VR systems tested were the Oculus Rift with Touch controllers and the HTC Vive with wand controllers. The proposed method requires only in-application self-directed movement from the user to achieve alignment. This is shown with an experiment simulating three common virtual reality scenarios. Section 2 outlines previous work in sensor fusion and alignment for virtual and augmented reality technologies. Section 3 describes the central contributions of this paper. Section 4 details an experiment performed to evaluate accuracy, speed and computation requirements.

2. Related Work

Alignment of two 3D sensor systems $\underline{\mathrm{S}}$ and $\underline{\mathrm{Q}}$ involves determining the transform $\mathbf{Y} : \underline{\mathrm{Q}} \rightarrow \underline{\mathrm{S}}$. It is typically assumed that the two systems are affine representations of the real world and that $\mathbf{Y}$ only has rotational and translational components. Scale components are assumed to be known and corrected for before transformation by $\mathbf{Y}$. In this case, we say $\mathbf{Y}$ is in the 3 dimensional *Special Euclidean Group*, or $\mathbf{Y} \in SE(3)$. The solution to this problem has many solutions, but in this work Arun et al.'s technique is used [AHB87]. The complete model for two 6DoF tracking systems is given by

$$\mathbf{A}_t \mathbf{X} = \mathbf{Y} \mathbf{B}_t \tag{1}$$

Here, S_t and Q_t are the coordinate frames measured at time t by the sensor systems $\underline{\mathrm{S}}$ and $\underline{\mathrm{Q}}$ respectively. Each measurement yields a result $\mathbf{A}_t \in SE(3)$ for $\underline{\mathrm{S}}$ and $\mathbf{B}_t \in SE(3)$ for $\underline{\mathrm{Q}}$. There are two unknown elements of $SE(3)$ to determine, $\mathbf{X} : Q_t \rightarrow S_t, \mathbf{Y} : \underline{\mathrm{Q}} \rightarrow \underline{\mathrm{S}}$ representing the sensor rigid connection and the reference frame relationship respectively [FS16]. This type of system can be solved with the well established hand-eye calibration developed for robotics systems [Sha13].

2.1. Homogeneous Depth Camera Co-Registration

Müller et al. [MIGL17] demonstrated a gait analysis system consisting of six fused Kinect v2 cameras networked across several computers. Registration of the Kinect v2 sensors was performed using mutually visible fiducial markers and Procrustes analysis. This process of explicit calibration achieves high accuracy such that the point clouds from each Kinect can also be unified. The result was a system which performed comparably to a gold standard motion capture system for the purpose of analyzing walking gaits. However, this alignment procedure requires a custom designed visual marker and a manual procedure which requires time and expertise. Additionally, this method cannot be used with systems which cannot track the marker, such as the HTC Vive which uses laser based tracking. Even for systems which could track the marker, such as the camera based tracking of the Oculus Rift or Playstation VR, custom software would need to be written for each system to account for lens distortion and other camera properties.

Rietzler et al. presented a real time framework for combining skeleton tracking data extracted from multiple depth cameras [RGTR16]. Their work is available as open source software called FusionKit. The software allows for multiple networked computers to send skeleton data from the Microsoft Kinect v2 to a central computer for registration and fusion. Identification of sensor dependencies is performed on a per-user basis, with skeletons matched before registration by joint configuration and length, or matched after registration by distance in the global coordinate frame. Registration is performed ambiently using Iterative Closest Point (ICP) algorithm performed on the skeleton joints generated by movement of a user within the mutual tracking space of two sensors. Our approach extends this approach to include inhomogeneous cases such as depth cameras with a VR system.

2.2. Inhomogeneous Tracking System Alignment

Czesak et al. created a system for full body tracking using three commodity tracking devices (Oculus Rift DK2, Leap Motion, Kinect v2) [CMC*16]. However, the system performed no closed-loop calibration and simply used each sensor system to track mutually exclusive body sections. There is ample room for improvement over this model. Destelle et al. demonstrates a procedure for fusing inertial measurement unit (IMU) data with optical data from the Microsoft Kinect v1 to produce a hybrid skeletal tracking solution with accuracy rivaling that of expensive gold standard optical tracking solutions [DAO*14]. However, part of the fusion process involves meticulous alignment of the IMU reference frames with one another, and alignment of the IMUs with respect to the Kinect reference frame. We aim to automate such procedures.

2.3. Ubiquitous Tracking

Ubitrack is an open source software system for modular fused real time 6DoF tracking for the purpose of augmented and virtual reality (AR/VR) applications [PHW*11]. Ubitrack features network infrastructure for modular sensor systems distributed across multiple computers. Ubitrack uses a Sensor Relation Graph (SRG) for data queries. This involves user configuration of sensor relationships to enable graph-search based queries for data in a given reference frame. However, calibration is not ambient and the user is required to perform specific motions with calibration objects. The configuration of Ubitrack is difficult and aimed exclusively at researchers and experts. The Ubitrack system is complex, containing code for numerous hardware drivers as well as the core fusion algorithms. This can be undesirable for rapid development due to increased complexity.

Society of Devices Toolkit (SoD-Toolkit) [SAC*15] is another open source ubiquitous tracking platform supporting a wide array of devices such as Microsoft Kinect and Leap Motion. However, the system does not provide high quality articulated body tracking, but rather focuses on providing coarse scale multi-user information with multiple tablet devices. Jester is an open source human skeletal sensor fusion layer for virtual environments [Sch14]. Jester defines a middle-ware architecture for abstracting the hardware and software layers in a virtual environment, while also providing support for fusion of sensors with support for basic per-joint filters such as the Kalman filter. Jester also requires explicit calibration of systems.

The work in this paper aims to improve accessibility and ease of use compared to the discussed systems by ambiently aligning sensors systems during typical use. Changes in the state of sensor

systems is common due to the trade off between stability and reconfigurability in sensor installation. Fault detection allows the system to function continuously without the need for manual re-calibration if a sensor drifts or is displaced. This is all done in real time. The following section describes our implementation and techniques.

3. Implementation of Ambient Calibration

We addressed the shortcomings of the systems discussed in Section 2 by developing a lightweight plugin for skeleton fusion targeting modern game engines. By using the game engine as a middleware, the proposed plugin does not handle device drivers and hardware configuration. By offloading driver and hardware abstraction, we were able to develop a focused system which can be easily integrated into other C++ compatible software platforms. Presently, only Unreal Engine 4 is supported, with future support for other game engines planned. This section describes our central contribution: calibration procedures with fault detection to allow for ambient fusion of inhomogeneous tracking data. The proposed procedures calibrate sensor systems automatically based on ambiently aquired sensor data. The advantage of the proposed system over conventional methods is that no setup is required by the user. Also, if the system configuration is disturbed, recovery is automatic and doesn't require effort from the user. This is all done in real time at VR compatible timescales.

3.1. Software Implementation

The plugin is written in C++ using only the standard C++11 libraries and the Eigen mathematics library [GJ10]. Figure 1 summarizes the structure of the system. There are two central software modules within the system - the *Calibrator* and the *Fusion Graph*. The Calibrator is responsible for aligning different sensor systems and detecting faults in calibration, representing the major contributions of this paper. More information on the calibrator is given later, in Section 3.2. The Fusion Graph models an articulated skeleton with support for sensor fusion. It is structured as a directed graph with nodes representing 3D affine transformations. Each node inherits the transform of its parent and is updated based on the fusion of the latest measurements assigned to the node. Currently, the Fusion Graph simply uses pre-defined priorities to select the best tracking result available for each node in the fusion step. However, more sophisticated fusion will be implemented in the future.

3.2. The Calibrator

For each iteration of the application, the Calibrator might receive a number of measurements. Each measurement corresponds to a sensor from a single sensor system. More than one measurement from a single sensor can be collected each frame. The Calibrator stores new measurements for a pair of connected sensors only if both of their measurements differ from their previous recorded measurements such that the sum of the position change (m) and the angular change (rad) is greater than $r = 0.075$. That is, if the position difference is more than 7.5cm or the angular difference is more than about 4.3° for both connected sensors. This reduces redundant information and removes the dependency of the calibration on the timing of the user's movements. Calibration between S and Q is then performed when M measurement pairs (S_t, Q_t) are available. The evaluation in this paper uses $M = 100$. The values for M and r used were determined by trial and error to trade off calibration quality against the time taken to collect data. Larger M values monotonically result in longer data gathering and calibration computation times, but give higher accuracy and reliability. Smaller r values monotonically result in faster gathering times, but lower data variety and hence lower quality.

The Calibrator determines which systems can be aligned from the sensor measurements available to it. This is done continuously in real time using a data structure called the *system-node table*. A *system* is a label corresponding to a single reference frame. For example, Vive, Rift, Kinect, OptiTrack, etc. A *node* is a label corresponding to a real world object, such as 'Left Hand', 'Right Hand', 'Head', 'Box 1', etc. A *sensor* is a label corresponding to a single sensor within a given system, usually simply an integer. Each sensor is tagged with exactly one node and one system, but each system and node can have many sensors and corresponding measurements. The system-node table maps one system and one node to their corresponding sensors, and the measurements corresponding to those sensors.

Each frame, the Calibrator decides if calibration is viable between each unordered pair of systems (S,Q). Calibration is performed as follows:

1. For each node (row in the system-node table):
   a. Check if S and Q have corresponding measurements
   b. If there are more than m corresponding measurements for the node, store them for calibration later
2. If more than M measurements stored in total, perform calibration as described in [AHB87, Sha13, PHBK06]
3. Clear measurements which will not be used for further calibration

In this way, the measurements are sorted and analyzed in real time with only a small overhead. Typical values for the parameters M and m were $M = 100$ and $m = 4$. The update frame-rate is set in configuration. The algorithm operates asynchronously relative to each sensor system. Interpolation is used to synchronize measurements sampled at times with small differences. However, two

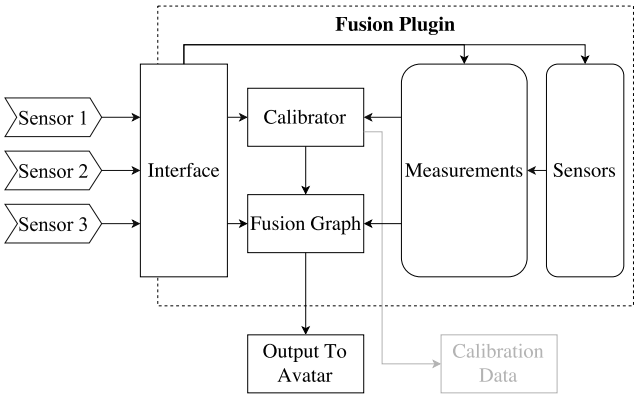

Figure 1: *Software architecture of the fusion plugin.*

measurements received at the same time are naively considered synchronous. We leave ambient latency compensation techniques (e.g. [HSK09]) to future work since it is easily implemented as a pre-processing step to improve overall performance of the system.

3.3. Calibration State Machine

The calibration of each pair of systems occupies one of three calibration states:

- **Uncalibrated (U)** - no calibration information available. The system is either still gathering data regarding the two tracking systems or the systems cannot be calibrated because they share no dependent data.
- **Refinement (R)** - partial data is available for the calibration. The system will continue to refine the result until convergence is achieved.
- **Calibrated (C)** - the calibration has converged and the system is no longer adjusting calibration. Fault detection is now running to detect a systematic error between the two systems.

A state machine is maintained for each pair of systems $(\underline{S},\underline{Q})$. Every time a calibration is performed between two systems, their calibration state is updated according to the state transition diagram in Figure 2.

The transition conditions are computed from the mean validation error E of the latest calibration. Computation of E depends on the type of calibration. For example, $E = \frac{1}{N}\sum_{t=1}^{N} ||\mathbf{A}_t\mathbf{X} - \mathbf{Y}\mathbf{B}_t||$ with the Frobenius norm for a complete 6DoF hand-eye calibration or $E = \frac{1}{N}\sum_{t=1}^{N} ||\mathbf{Y}\mathbf{b}_t - \mathbf{a}_t||$ for position only point cloud alignment. Additionally, a quality measure $q : \mathbb{R} \to [0,1]$ is computed from the error value to create a bounded metric of calibration performance

$$q(E) = \frac{1}{1+(E/s)^2} \tag{2}$$

where s is a tunable scale parameter, fixed at runtime. For 6DoF calibration error, $s = 1$ was used, while $s = 0.05$ was used for position only calibration error. This accounts for the different magnitudes of the different norms. The quality measure $q(E)$ is monotonically decreasing with E, with $q(E) = 1$ indicating a perfect solution.

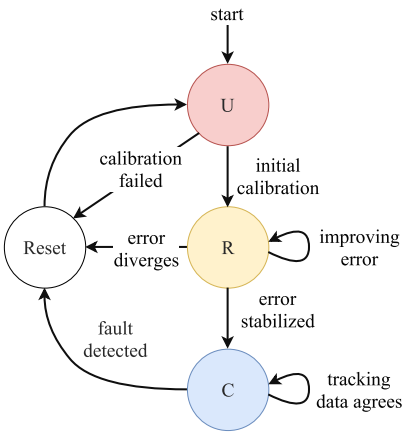

Figure 2: *State machine for calibration between two systems. States: Uncalibrated (U), Refining (R), Calibrated (C).*

The state of the calibration of $(\underline{S},\underline{Q})$ is defined by the tuple $(\mathbf{Y},E,N,q) \in SE(3) \times \mathbb{R} \times \mathbb{N} \times [0,1]$. Here, $\mathbf{Y}$ is the 4×4 homogeneous matrix $\mathbf{Y} : \underline{S} \to \underline{Q}$, E is the mean validation error, N is the number of samples which have been used to obtain $\mathbf{Y}$ and q is the quality of the calibration. After each calibration operation, the calibration state $(\mathbf{Y},E,N,q)$ is updated using the new calibration result $(\mathbf{Y}',E',N',q(E'))$ by interpolation weighted by N and N':

$$\mathbf{Y} \leftarrow \text{slerp}\left(\mathbf{Y},\mathbf{Y}',\frac{N'}{N+N'}\right) \tag{3}$$

$$E \leftarrow \frac{NE+N'E'}{N+N'} \tag{4}$$

$$q \leftarrow \frac{Nq+N'q(E')}{N+N'} \tag{5}$$

$$N \leftarrow N+N' \tag{6}$$

Here, $\text{slerp}(\mathbf{X},\mathbf{Y},\alpha) \in \text{SE(3)}$ is the spherical linear interpolation of $\mathbf{R}_\mathbf{X}$ to $\mathbf{R}_\mathbf{Y}$ by α and the linear interpolation from $\mathbf{x}$ to $\mathbf{y}$ by $\alpha \in [0,1]$. This type of interpolation update guarantees eventual convergence of the results, leading to a stable calibration result while accurately weighting any erroneous results. Table 1 describes the transition logic which is followed whenever a new calibration result is computed. Here, the signed change in quality $\Delta q(E)$ is defined as the change in quality which occurred due to the update in Equation 5. The state diagram in Figure 2 is used to update the state of the calibration.

Table 1: *Transition conditions for the state machine shown in Figure 2. Each of the hard-coded values here are configurable in reality - these are the values that were found to work well.*

Transition name	Condition
Initial calibration	$q(E) > 0.5$
Calibration failed	$q(E) \leq 0.5$
Error stabilized	$(\Delta q(E) < 0.01) \wedge (q(E) > 0.90)$
Improving error	$\Delta q(E) > 0.01$
Error diverges	$\neg$(Error stabilized) $\wedge$ $\neg$(Improving error)
Fault detected	See Section 3.4
Tracking data agrees	$\neg$(Fault detected)

3.4. Fault Detection

Fault detection is responsible for detecting a change in configuration in the sensor network, such as movement or drift of a reference frame. When the Calibrator is in the Calibrated state, new calibrations are performed as usual (Section 3.2). However, the update step given by Equations 3-6 is not performed. Instead a *background calibration* result $\mathbf{Y}_B \in \text{SE(3)}$ is tracked. The background calibration result is updated in two steps based on each new calibration result $\mathbf{Y}'$. First, a decay step:

$$\mathbf{Y}_\beta := \text{slerp}\left(\mathbf{Y}_B,\mathbf{I}_{4,4},\beta\right) \tag{7}$$

where $\mathbf{I}_{4,4}$ is the 4×4 identity matrix and $\beta \in [0,1]$ is the decay rate ($\beta = 0.1$ was used in the evaluation). Secondly, an exponential filter

step is used to compute the new value of $\mathbf{Y}_B$

$$\mathbf{Y}_B \leftarrow \text{slerp}\left(\mathbf{Y}_\beta, \mathbf{Y}', \gamma\right) \tag{8}$$

where $\gamma \in [0,1]$ was the exponential gain ($\gamma = 0.25$ was used in the evaluation). After each calibration, $\mathbf{Y}_B$ is compared to the calibration result $\mathbf{Y}$ computed just before transitioning from the Refining state to the Calibrated state. If $\mathbf{Y}_B^{-1}\mathbf{Y}$ describes a rotation of angle greater than θ or displacement more than d, a fault is considered to be detected, and calibration is reset. In the evaluation, $\theta = 5^\circ$ and $d = 10$cm was used. This system is designed to be robust to noise and false positives. The cost of this design is that fault detection can take longer than calibration from the Uncalibrated state.

4. Evaluation

To evaluate the proposed solution, the plugin was used to calibrate a Kinect v2 skeleton tracking camera with two different VR systems. The VR systems track the users hands and head with 6DoF while the Kinect measured just body joint positions. The systems tested were the HTC Vive with wand controllers and the Oculus Rift with Touch controllers. The Kinect v2 provides measurements of the user's entire body, but at a coarse scale compared to the VR tracking systems. The mean error in Kinect v2 tracking compared to ground truth of gold standard tracking has been reported to be around 23cm for the hands and 5cm for the head [RGTR16]. This error should be kept in mind when considering the results in the following sections.

4.1. Method

The Kinect was placed approximately 1.5 meters off the ground. A user performed tasks modeled around modern room scale VR applications. For example, walking, handling virtual objects and shooting (pointing) at distant virtual targets. The actions were performed while facing the body no further than 90° away from the Kinect. After around 10-50 seconds (depending on the task), the system computed the alignment as described in [AHB87]. The resulting transform $\mathbf{Y}$ was then compared to the actual location of the Kinect as measured by an OptiTrack motion capture system. The actions performed to collect the data included the following VR tasks:

- **Sorting** (Figure 3a,b) - the user is tasked with picking up virtual items from a shelf walking a short distance to place them on a virtual platform of the same color. This simulates applications such as *Job Simulator* by Owlchemy Labs.
- **Pointing** (Figure 3c) - the user is tasked with pointing at targets placed up to 90° either side of the user and up to 90° elevation. This task simulates applications where the user stands still but uses their hands, such as *Robo-Recall* by Epic Games.
- **Walking** (Figure 3d) - the user walked around the tracking space while facing toward the Kinect while moving arms slowly up and down. This represents an 'ideal' calibration scenario with large amounts of movement around the tracking space.

4.2. Results

For each movement type, the task was performed until the Calibrated state was reached. The error was then recorded before resetting calibration and starting again. Each trial was repeated 10 times and the results are shown in Figure 4 and summarized in Table 2. The durations required to gather data for calibration are also noted in Table 2. Errors were computed based on ground truth Kinect pose measured using a gold-standard OptiTrack motion capture system (see Figure 5). The OptiTrack system was used to compute a mapping from the VR tracking space to the Kinect tracking space. First, the mapping $\mathbf{V} \in SE(3)$ was computed using a hand-eye calibration [Sha13] between an OptiTrack rigid body marker and a VR controller. Next, a marker was placed on the Kinect and calibrated such that it coincided in orientation and position to the true center and orientation of the Kinect. The OptiTrack system then measures the mapping $\mathbf{K} \in SE(3)$ between the OptiTrack system and the marker on the Kinect. If the transform between the marker on the Kinect and the Kinect tracking space is $\mathbf{I} \in SE(3)$, the system equation is found by forming a loop in Figure 5 giving $\mathbf{IKV} = \mathbf{Y}$. We make the approximation that $\mathbf{I}$ is equal to the identity transform, and thus the calibration error is given by the matrix $\mathbf{E}(\mathbf{Y}) := \mathbf{KVY}^{-1}$. The error matrix $\mathbf{E}(\mathbf{Y})$ will be equal to the identity when the calibration $\mathbf{Y}$ is perfectly accurate. The final error values reported in Table 2 are decomposed into the magnitude of the translation of $\mathbf{E}(\mathbf{Y})$ and the magnitude of the angle of $\mathbf{E}(\mathbf{Y})$ when decomposed into angle-axis form.

4.3. Discussion

The mean error in Kinect v2 tracking compared to ground truth of gold standard tracking has been reported to be around 23cm for the hands and 5cm for the head [RGTR16]. The calibration results (Figure 4 and Table 2) compare favorably with the expectation of the hand tracking error, but not the head tracking error. This is likely explained by the fact that less data from the user's head is used in calibration than data from the arms simply because people tend to move their arms more than their head. This is a limitation with the problem of ambient calibration itself - you cannot control the user's actions. Future work will improve performance in this domain by incorporating prior information about relative reliability of different sensors, as is well established in the literature (e.g. [RGTR16]). Subjectively, the skeleton tracked by the Kinect matches well with the real body position, even with 22.2cm and 5.25° error. Figure 6 shows such a typical calibration result visualized from first person and third person perspectives. The avatar aligns well with the real body seen through the Vive's pass-through camera. It should be noted that the pass-through camera is offset slightly below and in front of the user's eyes, and this introduces some error in where the body appears with the pass-through camera. More distant objects are less affected by this error, such as the feet. Our method is agnostic to device and tracking method, though it does not achieve the same level of accuracy as Müller et al., who use time consuming and fault sensitive fiducial based calibration [MIGL17].

Worse results were observed in the pointing task. This is expected since a smaller variety of data is collected compared to the other tasks. In particular, the head remained mostly stationary and thus the hand trackers recorded the majority of data for calibration. The walking task and the sorting task performed similarly, suggesting that it is important to utilize the more accurate head tracking of the Kinect. However, it is necessary to use the hand data to sam-

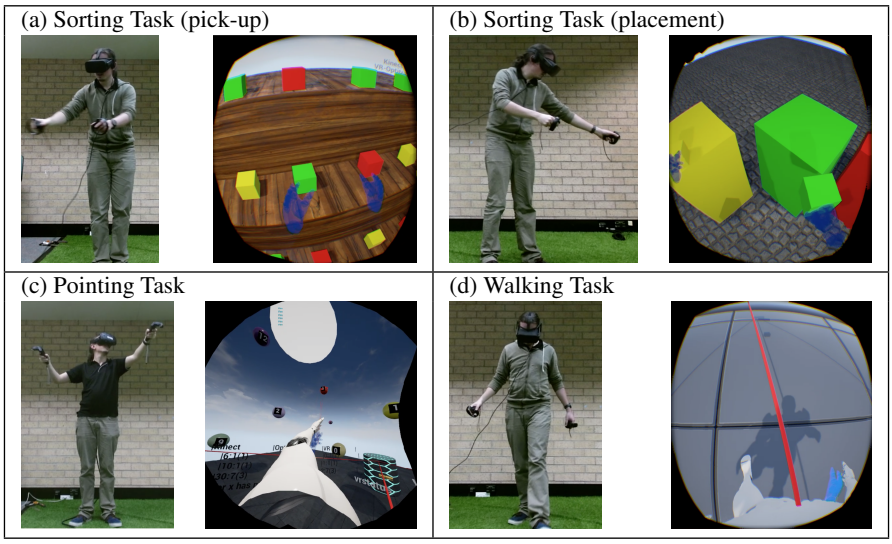

Figure 3: *The three tasks used to assess the performance of the ambient calibration. In the sorting task, the user must sort the cubes (a) into their respective colors (b) on the platforms a few steps away. In the pointing task, the user must point to a series of targets while standing in place (c). The walking task involves the user stepping around the tracking space while facing the Kinect and slowly raising and lowering their arms (d).*

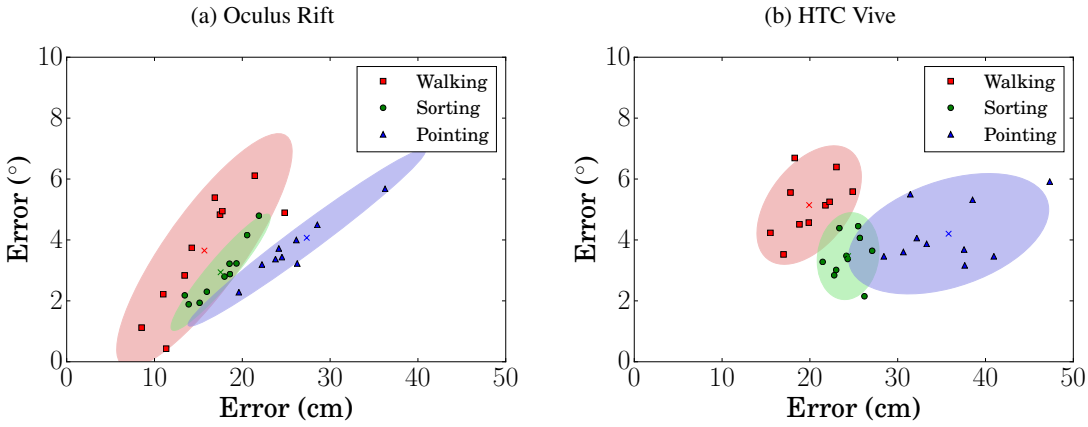

Figure 4: *Distributions of errors for Rift (a) and Vive (b) for three different ambient calibration scenarios. Confidence ellipses (95%) are shown as visual aids.*

ple outside the plane the head usually moves within (crouching of the user is rare). The walking task likely performed best due to the lower redundancy in data collected.

Figure 7 demonstrates an instance of ambient calibration, fault detection and recovery. The ground truth pose of the Kinect is shown as a function of time alongside the positional and rotational calibration error of the system compared to ground truth. The calibration states of the system are indicated by background color of the graph. A fault of about 8° in the yaw position of the Kinect is detected and corrected within 40 seconds. Calibration remains stable otherwise. The time taken to calibrate and detect faults is well within the desired range for ambient calibration. The actual mean computation time required for the calibration operations was measured to be at most 2.64 ($\pm$ 0.30) ms. Analysis was performed on a Windows 10 PC with an Intel Xeon E5-1650 v3, 3.50GHz, 6 cores. However, this amount of compute time was only required on the frames where calibration was triggered, or once every 5 seconds or so. This amount is much less than the typical render budget of 11ms allocated for an application running at 90Hz. Also, multi-threading could be used to perform the calibration during the span several frames if necessary. During data collection, for 3 systems (Kinect, OptiTrack and a VR system) computation requirements were measured to be about 0.3ms for processing measurements and updating the Fusion Graph state (see Section 3.1 and Figure 1).

The Kinect cannot distinguish between the cases of the user facing the device and facing away from the device. The proposed tech-

Table 2: *Summary statistics for errors of ambient calibration procedure compared to ground truth. Values are reported in 'mean ± standard deviation' format.*

		Position Error (cm)	Angle Error (°)	Time (sec)
Walking	**Rift**	15.7 ± 4.7	3.65 ± 1.8	13 ± 2
	Vive	19.9 ± 2.8	5.15 ± 0.9	11 ± 2
Sorting	**Rift**	17.5 ± 2.7	2.93 ± 0.9	13 ± 3
	Vive	24.3 ± 1.7	3.47 ± 0.7	15 ± 3
Pointing	**Rift**	27.34 ± 6.4	4.07 ± 1.4	34 ± 4
	Vive	35.8 ± 5.4	4.2 ± 0.9	40 ± 16

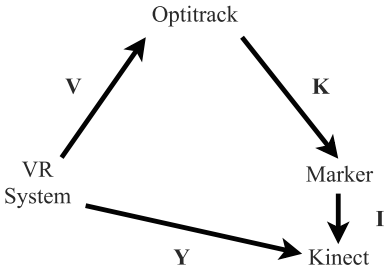

Figure 5: *To measure the accuracy of the ambient calibration* **Y** *between the Kinect and the VR systems, an external gold-standard OptiTrack motion capture system was used to measure* **K**.

nique does not take into account this shortcoming and so calibration can only be performed facing the Kinect. In future work, this could be overcome by breaking symmetry using the VR system. Additionally, the calibration model doesn't account for the non-zero rigid transform which naturally must exist between two rigidly linked sensors. The Kinect tracks the wrist position, whereas the reported center position of the controller is not on the wrist position, but rather on the controller itself. A similar model describes the head: the VR headset center is not the same point as the head point measured by the Kinect. These offsets vary depending on the devices involved and the physiology and behaviour of the user, but could be considered constant over a session of usage. Therefore the approach of configuring offsets manually is highly inconvenient. Point cloud alignment used in this paper doesn't account for these differences and instead assumes the same point is measured from both reference frames [AHB87]. This explains the discrepancy in results between the Rift and the Vive seen in Figure 4. The Vive wand controllers are much larger than the Touch controllers, and so are more likely to have a larger rigid offset from the Kinect position. This problem of automatically determining individual offsets requires further research. The Rift results (Figure 4a) feature significant correlation between rotational error and positional error. This is due to the technique used for calibration; calculation of the rotation transform is performed first, and the positional error depends on the accuracy of the resulting transform. The Vive shows less correlation in angular and positional error, likely due to the previously described rigid link factor which masks the correlation.

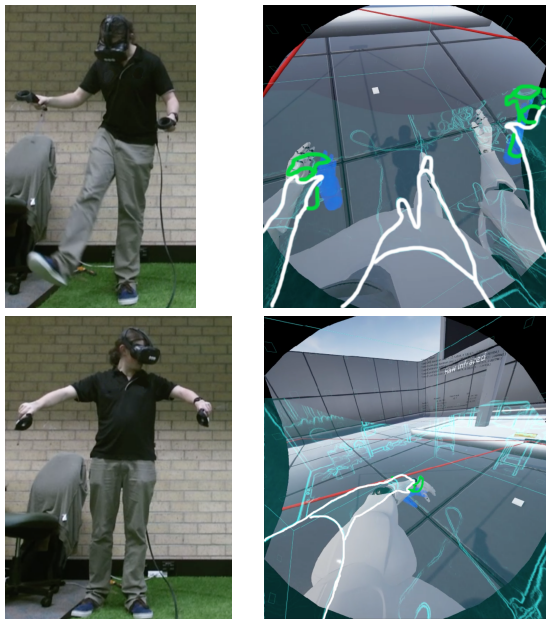

Figure 6: *An example of the typical results for calibration with walking activity. An outline of the user's real body from the Vive's passthrough camera is shown on the right, overlayed with the virtual scene rendered from user perspective. This calibration took 16 seconds, and had an error of 22.2cm and 5.25°.*

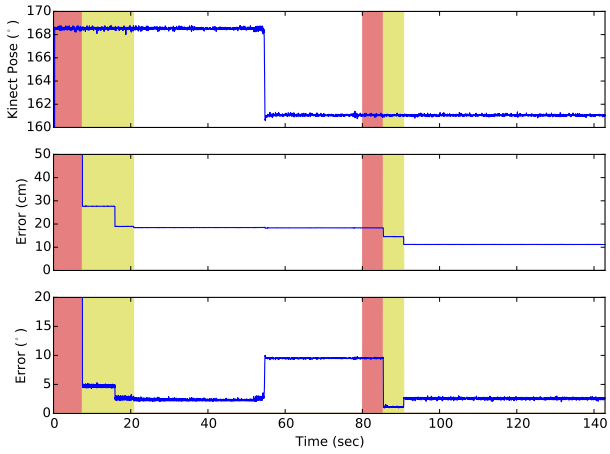

Figure 7: *An example calibration trace with calibration states (Figure 2) overlayed in color (Red = 'Uncalibrated', Yellow = 'Refining', White = 'Calibrated'). The kinect is moved at the 55 second mark, giving it a rotation 8 degrees from its original configuration. The error is corrected after about 35 seconds.*

4.4. Future Work

Motion capture of an articulated system such as the human hand or human body can utilize inverse kinematics and contextual constraint information to infer the poses of untracked joints. Early examples of such research have applied inverse kinematics tech-

niques to estimate arm pose given hand pose [TB96]. Hand tracking can be performed by tracking the wrist position and each fingertip provided appropriate constraints are applied [AL10]. By searching within prior learned manifold of plausible human poses, markerless motion capture from a single camera is possible, though not in real time [PMBG*11]. Future work will incorporate these ideas.

Other future work will involve improving the accuracy and usability of the ambient calibration algorithms. A method for accounting for rigid offsets will be investigated. Methods for ambiently compensating for latency will be needed for improved accuracy. Methods for identification of dependencies amongst sensors will be incorporated into the system to further reduce the configuration requirements [FS16]. For example, the Kinect symmetry problem could be resolved by matching hand motions to left and right controller motions. More sophisticated skeleton fusion methods will be investigated. On the software side, we hope to simplify the interface, supply example support for more devices and create an open source code release soon. For now, the code can be viewed at http://www.github.com/JakeFountain/Spooky.

5. Conclusion

A method for ambient calibration of sensor systems is presented with open-source code available for Unreal Engine 4. The central contribution is a state machine for gathering data, calibration and fault detection. An evaluation was performed to calibrate two example VR systems with the Kinect v2. It was demonstrated that the resulting calibration has accuracy on the order of what is expected given the accuracy of the Kinect. Typical accuracy was demonstrated to be 20cm and 4° compared to the ground truth with only around 20 seconds of in-application user-directed movement. This work is an important step toward ambient calibration and fusion of real-time sensor systems.

References

[AHB87] ARUN K. S., HUANG T. S., BLOSTEIN S. D.: Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-9*, 5 (Sept. 1987), 698–700. doi:10.1109/TPAMI.1987.4767965. 2, 3, 5, 7

[AL10] ARISTIDOU A., LASENBY J.: Motion capture with constrained inverse kinematics for real-time hand tracking. In *4th International Symposium on Communications, Control and Signal Processing (ISCCSP)* (Mar. 2010), pp. 1–5. doi:10.1109/ISCCSP.2010.5463419. 8

[CB16] CUMMINGS J. J., BAILENSON J. N.: How immersive is enough? A meta-analysis of the effect of immersive technology on user presence. *Media Psychology 19*, 2 (Apr. 2016), 272–309. doi:10.1080/15213269.2015.1015740. 1

[CMC*16] CZESAK K., MOHEDANO R., CARBALLEIRA P., CABRERA J., GARCIA N.: Fusion of pose and head tracking data for immersive mixed-reality application development. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)* (2016), IEEE, pp. 1–4. URL: http://ieeexplore.ieee.org/abstract/document/7548886/. 2

[DAO*14] DESTELLE F., AHMADI A., O'CONNOR N. E., MORAN K., CHATZITOFIS A., ZARPALAS D., DARAS P.: Low-cost accurate skeleton tracking based on fusion of kinect and wearable inertial sensors. In *22nd European Signal Processing Conference (EUSIPCO)* (Sept. 2014), pp. 371–375. 2

[FS16] FOUNTAIN J., SMITH S. P.: Automatic identification of rigidly linked 6dof sensors. In *IEEE Virtual Reality 2016* (Mar. 2016), pp. 175–176. doi:10.1109/VR.2016.7504710. 2, 8

[GJ10] GUENNEBAUD G., JACOB B.: *Eigen v3*. 2010. URL: http://eigen.tuxfamily.org. 3

[GWFM17] GREENWALD S. W., WANG Z., FUNK M., MAES P.: Investigating social presence and communication with embodied avatars in room-scale virtual reality. In *International Conference on Immersive Learning* (2017), Springer, pp. 75–90. URL: http://link.springer.com/chapter/10.1007/978-3-319-60633-0_7. 1

[HSK09] HUBER M., SCHLEGEL M., KLINKER G.: Temporal calibration in multisensor tracking setups. In *8th IEEE International Symposium on Mixed and Augmented Reality* (2009), IEEE, pp. 195–196. URL: http://ieeexplore.ieee.org/abstract/document/5336465/. 4

[MIGL17] MÜLLER B., ILG W., GIESE M. A., LUDOLPH N.: Improved kinect sensor based motion capturing system for gait assessment. *bioRxiv* (Jan. 2017). URL: http://biorxiv.org/content/early/2017/01/10/098863.abstract, doi:10.1101/098863. 2, 5

[PHBK06] PUSTKA D., HUBER M., BAUER M., KLINKER G.: Spatial relationship patterns: elements of reusable tracking and calibration systems. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality* (2006), ISMAR '06, IEEE Computer Society, pp. 88–97. doi:10.1109/ISMAR.2006.297799. 3

[PHW*11] PUSTKA D., HUBER M., WAECHTER C., ECHTLER F., KEITLER P., NEWMAN J., SCHMALSTIEG D., KLINKER G.: Automatic configuration of pervasive sensor networks for augmented reality. *IEEE Pervasive Computing 10*, 3 (July 2011), 68–79. doi:10.1109/MPRV.2010.50. 2

[PMBG*11] PONS-MOLL G., BAAK A., GALL J., LEAL-TAIXÉ L., MÜLLER M., SEIDEL H.-P., ROSENHAHN B.: Outdoor Human motion capture using inverse kinematics and von mises-fisher sampling. In *2011 International Conference on Computer Vision* (Nov. 2011), pp. 1243–1250. doi:10.1109/ICCV.2011.6126375. 8

[RGTR16] RIETZLER M., GEISELHART F., THOMAS J., RUKZIO E.: FusionKit: a generic toolkit for skeleton, marker and rigid-body tracking. In *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (2016), EICS '16, ACM, pp. 73–84. doi:10.1145/2933242.2933263. 1, 2, 5

[SAC*15] SEYED T., AZAZI A., CHAN E., WANG Y., MAURER F.: SoD-Toolkit: a toolkit for interactively prototyping and developing multi-sensor, multi-device environments. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (2015), ITS '15, ACM, pp. 171–180. doi:10.1145/2817721.2817750. 2

[Sch14] SCHAPANSKY K.: *Jester: a device abstraction and data fusion API for skeletal tracking*. Master's Thesis, California Polytechnic State University, June 2014. 2

[Sha13] SHAH M.: Solving the robot-world/hand-eye calibration problem using the kronecker product. *Journal of Mechanisms and Robotics 5*, 3 (June 2013), 031007–031007. doi:10.1115/1.4024473. 2, 3, 5

[SNB*17] SKARBEZ R., NEYRET S., BROOKS F. P., SLATER M., WHITTON M. C.: A psychophysical experiment regarding components of the plausibility illusion. *IEEE Transactions on Visualization and Computer Graphics 23*, 4 (Apr. 2017), 1369–1378. doi:10.1109/TVCG.2017.2657158. 1

[TB96] TOLANI D., BADLER N. I.: Real-time inverse kinematics of the human arm. *Presence: Teleoperators and Virtual Environments 5*, 4 (Jan. 1996), 393–401. doi:10.1162/pres.1996.5.4.393. 8