Accurate Absolute Localization Methods for the HTC Vive Motion Capture System

Miguel Borges*, Andrew Symington† and Rodrigo Ventura*,
*Instituto Superior Técnico
†NASA Ames Research Center

***Abstract*—Most of the high accuracy off-the-shelf full pose (position and orientation) ground truth systems are extremely expensive for a user or researcher with limited resources. Nevertheless, there is a platform that has been gaining attention as a cost-effective alternative for collecting ground truth pose data: HTC Vive (designated by Vive from now on). This system has been designed however for Virtual Reality applications, focusing mainly on low-latency and the smoothness of the trajectories instead of prioritizing the pose estimation's accuracy. Hence, the Vive's baseline algorithms are poorly suited for robotic applications, where accuracy and repeatability are key. The objective of our work is to improve the performance of Vive for robotic applications through the adaptation of different estimation methods tuned for accuracy and the development of an automated method to estimate the optimal setup of the system. In a series of controlled experiments we compare Vive to a high-accuracy motion capture system, where we show that the position's maximum error between the systems is in the low centimeter magnitude and that the average achieved error is in the low to mid millimeter magnitude.**

***Index Terms*—Localization; Ground Truth; Accuracy; Robotics; Virtual Reality; Low-cost Tracking.**

I. Introduction

There are multiple ground truth systems available for roboticists capable of estimating full poses accurately enough for diverse applications as is the case of the motion capture[1] systems. These are however prohibitively expensive and only well funded research laboratories or private companies are able to purchase them. There are more affordable alternatives but they tend to focus on a particular application.

Nowadays, with the massification of consumer electronics, technology is becoming more affordable and one of the industries that benefited significantly from this, is Virtual Reality (VR), where accurate pose estimation is also required to track the users. One of the platforms that has been gaining attention and that made VR go mainstream is HTC Vive. The requirements that VR imposes on Vive makes it a compelling way of collecting ground truth pose data for roboticists and it has inclusively been reported as being able to achieve sub-millimeter precision[2]. Vive is also easy to set-up, affordable when compared to competing technologies and a Robot Operating System (ROS) driver is already available, allowing access to its baseline algorithms.

[1]Motion capture consists of estimating the absolute position and orientation (or pose) of an object or subject in real time.

[2]www.roadtovr.com/analysis-of-valves-lighthouse-tracking-system-reveals-accuracy/ (last visited on 01/05/2019)

Vive's motion capture resorts to base stations that emit infrared infrared pulses that are detected by the tracked devices. These devices use the photodiodes to detect the light pulses and also resort to an Inertial Measurement Unit (IMU) to reduce latency and smoothen the trajectories. The IMU's high sampling rate prevents motion sickness in the VR experience at the cost however of deteriorating the accuracy and repeatability of its pose estimation. These constraints mean that the sub-millimeter precision is only achievable in a static state. When the devices are in a dynamic state the poses often drift, with the error reaching 90 cm for the position.

Our objective is to use Vive as a cost-effective ground truth localization system. We aim at improving the accuracy and repeatability of the pose estimation by decreasing the weight of the inertial measurements and increasing the robustness to outliers. The goal is to reduce the gap between Vive's accuracy and the one of a state-of-the-art motion capture system. We also intend to adapt different algorithms and methods to compare the performance achieved by each one.

We address this problem by implementing a set of methods to improve the system's accuracy by decreasing the influence of the inertial data and also by implementing a stricter outlier rejection policy. These algorithms include in the system's model the parameters that correct the distortion in the lighthouse's ideal model, and are based on the following methods: Least-Squares (LS), Extended Kalman Filter (EKF), Iterated Extended Kalman Filter (IEKF), Unscented Kalman Filter (UKF) and a fusion between Pose Graph Optimization (PGO) and Moving Horizon Estimation (MHE). In addition to these tracking algorithms, we implement a pair of procedures to address the environment calibration. In the end we compare the implemented algorithms against the motion capture system OptiTrack. We also study Vive's Cramer-Rao Lower Bound (CRLB) to estimate the system's precision bound and to obtain the optimal lighthouse placement.

We consider the contributions of our work to be the following: automated method for the placement of base stations; study of the minimum variance achievable by Vive in different conditions; formulation and development of multiple algorithms applied to this localization system; comparison between the implemented algorithms in a simulated environment; accuracy analysis of the implemented algorithms against a high accuracy motion capture system; ROS open-source software platform with the developed algorithms.

II. Existing Systems

In order to collect full-pose ground truth data, there are already solutions that address this particular problem roboticists face. They usually involve however compromising in the cost or in the accuracy.

A. Motion Capture Systems

The most widely spread systems to acquire high-accuracy poses usually rely on cameras that track reflective markers illuminated by infrared light sources. VICON in one of the platforms that applies this technology and has a claimed accuracy of up to 76 μm [1]. OptiTrack[3], Qualisys[4] and Motion Analysis Corporation[5] also resort to reflective markers to track their targets with sub-millimeter performance. VisualEyez [2] and OptoTrak Certus[6] employ a different tracking technique that is able to achieve precisions in the millimeter magnitude or even lower. Their solution uses a three-camera system to track a set of active LED markers.

The issue with the described motion capture systems is that they are prohibitively expensive for general adoption. As an example, a setup with OptiTrack equipment will cost between \$4,700 and and \$147,849, making it out reach for the average consumer or researcher.

The alternatives to these systems are usually tuned for specific applications and tend to not be as reliable. EthoVision is an integrated general purpose video tracking of activity, movement and interaction of animals [3]. [4] describes a tracking system with the goal of tracking small and fast moving objects such as flying insects. In [5], the authors developed a pan-tilt camera-based marker-tracking system. X Vision, in [6], manipulates images to do real-time tracking. [7] proposes a tracking system based on color coded markers and a video camera. [8] proposes GTvision and GTlaser, two ground truth systems that rely respectively on cameras to detect markers and lasers to estimate the pose of the robot, with a claimed accuracy in the centimeter range. SwisTrack, in [9], is also a tracking platform that was designed for a broader set of applications. It provides the user the ability to chose between multiple methods and it is able to achieve millimetric precision, but only for a small workspace.

B. Virtual Reality Platforms

The VR industry has been gaining momentum as more affordable systems become available. Currently, there are multiple affordable platforms in the market [10], being Oculus Rift and HTC Vive among the most used.

The Oculus Rift [11] is a platform created mainly for gaming applications. Rift resorts to the constellation (a base-station with a built-in camera) to detect the cluster of infrared LEDs on both the headset and the controllers and, identically to the Perspective-n-Point (PnP) problem (see [12–15]), compute their poses. This system also uses an IMU to smoothen the experience and increase the refresh rate. Oculus has also released recently Oculus Go, Oculus Rift S and Oculus Quest, which do not require a fixed base station to track their targets.

The HTC Vive platform is very similar to the Rift platform, but instead of using infrared LEDs, it has photodiodes that detect the light pulses from the lighthouses (base-station). Using time-differences and a rigid cluster of photodiodes, it is able to estimate the pose the tracked devices. Vive also resorts to inertial measurements (acceleration and angular velocity) to smoothen the user's experience. HTC and Valve have also recently released the HTC Vive Pro, which changes the functioning principles of the lighthouses and reduces at the same time the number of large moving parts.

Unfortunately, both Rift and Vive were developed for the consumer market, therefore the technical details, algorithms and protocols are not publicly available.

III. Vive Study

As there was already a ROS driver available for Vive we decided to use it as an accurate and cost-effective motion capture platform. Another factor that weighted significantly on Vive's favor was the existence of a library that was able to pull raw data from its devices[7]. Vive has multiple devices: the headset, the trackers/controllers and the lighthouses, but as our objective is robot tracking, we will only address the trackers/controllers and lighthouses, although the headset is very similar to the trackers and controllers, as these devices all rely on light and inertial data for tracking. The difference between the trackers and the controllers is related to the former being attached to objects while the latter is hand-worn. In this section we will only mention the trackers although the controllers principles are identical. We have also included a set o right-handed reference frames to each device: the lighthouse has only the lighthouse frame l while the trackers have the head frame (geometric center of the tracker), the light frame t (where the positions of the photodiodes are expressed) and the imu frame i (coincident with the IMU sensor).

The inertial data (linear acceleration and angular velocity) originates from the accelerometer and gyroscope in the trackers' microelectromechanical system (MEMS) IMU. The high frequency of this sensor allows the system to reduce the latency and increase the smoothness of the global trajectory, improving the VR experience. Inexpensive IMU sensors are however generally noisy, which may lead to the loss of accuracy [16].

The interaction between the lighthouses and the trackers allows the estimation of an absolute pose independently, unlike with the IMU data. The lighthouses emit two types of light beams: a synchronization pulse (from an array of LEDs) and a sweeping planar laser (from lasers attached to rotors). The lighthouse's internal configuration can be seen in figure 1, where it's possible to verify that the rotors are arranged orthogonally to adequately constrain the poses.

[3]www.optitrack.com/motion-capture-robotics/ (last visited on: 15/07/2019)

[4]www.qualisys.com/applications/sports/running/ (last visited on: 15/07/2019)

[5]www.motionanalysis.com/system/osprey/ (last visited on: 25/05/2018)

[6]www.ndigital.com/msci/products/optotrak-certus/ (last visited on: 15/07/2019)

[7]The libsurvive library is available at www.github.com/cnlohr/libsurvive

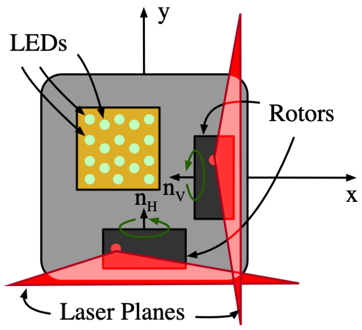

Fig. 1: Internal configuration of the lighthouse with n_H and n_V being the its rotors' rotation axes and x and y representing the axes of the lighthouses' frame.

The trackers resort to photodiodes to detect the infrared light beams and, taking into account that the lasers rotate at a constant frequency (60 Hz), it converts the time-differences between the beams to angles. Each photodiode detects four events: start and end of the synch pulse and start and end of the laser sweep. Vive uses the start of the synchronization pulse and center of the laser sweep (obtained by averaging its start and end) to compute the angle α, which represents the azimuth and the altitude of the photodiode, depending on whether it is obtained from the horizontal or the vertical sweep. If we consider both the horizontal and vertical α angles associated to each photodiode, this problem becomes very similar to the PnP problem, where at least four photodiodes are required to estimate a full pose, however five are recommended for a good estimate with Vive[8]. From the start and end of the laser sweep, it is also possible to estimate another angle, β, for which there is however no accurate model available to represent it. In figure 2, the reader can see both α and β.

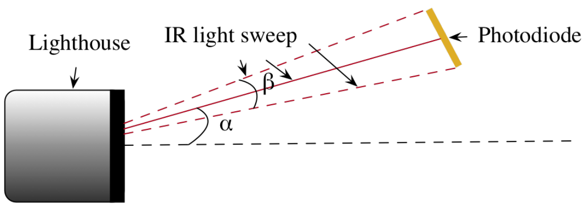

Fig. 2: Detailed sideview of the HTC Vive's working principle.

When there are two lighthouses available instead of one, we can compute the individual positions of the photodiodes using a minimum of three angles α, as long as the sweeping laser planes are not coincident with each other. In this case, to estimate the full pose of the tracker, three photodiodes are required. To be able to use multiple lighthouses, they can be configured differently to select different sampling frequencies and synchronization modes. The lighthouses also encode data in the synchronization pulse's duration, which includes the axis of that sweep, to which lighthouse the sweep corresponds, and by subdividing information through multiple pulses, the serial identifier of the lighthouse and also its correction parameters.

The well known model for respectively the horizontal and vertical α, considering ideal lighthouses, is the following:

$$\alpha_H = \operatorname{atan}(x) \tag{1a}$$

$$\alpha_V = \operatorname{atan}(y) \tag{1b}$$

where x and y are respectively the quotient between ${}^{l}\mathbf{P}_p$'s x and z coordinates, and y and z coordinates, with ${}^{l}\mathbf{P}_p$ being the position of photodiode p in lighthouse l's frame. If we however have into account the correction parameters[9], the model changes to the following:

$$\alpha_H = \operatorname{atan}(x) - \phi_l - \tan(\tau_l) y - c_l y^2 - m_r \sin[\phi_r + \operatorname{atan}(x)] \tag{2a}$$

$$\alpha_V = \operatorname{atan}(y) - \phi_l - \tan(\tau_l) x - c_l x^2 - m_r \sin[\phi_r + \operatorname{atan}(y)] \tag{2b}$$

where ϕ_l, τ_l and c_l are respectively the the offset, tilt and curvature associated to the laser and ϕ_r and m_r correct for the alignment offsets between the laser, mirror and lens[10].

Each tracker also has a set of factory calibration parameters written in its firmware, as is the case of the photodiodes' positions, the IMU's parameters and internal frame transforms.

IV. Approach

As Vive's algorithms are closed-source, we developed our methods to try achieve a better tracking performance in robotic applications and implemented them in ROS platform with a modular architecture. Besides the frames already mentioned, we now also use a world frame w and the vive frame v. The vive frame is used an intermediary frame to relate the lighthouses.

A. *Algorithm for Pose Estimation*

The Algorithm for Pose Estimation (APE) tackles Vive's localization problem in real-time and is based on a non-linear LS approach to the problem, where only the raw light data is used. This method has already been published in [17]. We have however developed a second iteration of this method, where the light model includes now the lighthouse's laser's distortion parameters, therefore replacing the model (1) with (2). To distinguish both methods, we will refer to the first as APE1 and to the new one as APE2.

[8]https://www.youtube.com/watch?v=xrsUMEbLtOs&list=WL&index=15&t=186s (last visited on 26/06/2019)

[9]https://github.com/cnlohr/libsurvive/wiki/BSD-Calibration-Values (last visited on 05/05/2019)

[10]https://www.reddit.com/r/Vive/comments/5s23ha/lighthouse_factory_calibration_values/ddcd7d8/ (last visited on 10/07/2019)

B. Kalman Filter

The Kalman Filter (KF) uses measurements over time to estimate parameters by computing their joint probability distribution [18]. We base our KF's implementation on [19]'s model, using both light and inertial data now.

The inertial model resorts to a state composed of the position, velocity and orientation (in the quaternion form) of the tracker's imu frame in the vive frame, to the bias of the measured angle velocity and to the gravity vector. To describe the process the following state-space model is used:

$$\begin{bmatrix} {}^v\dot{\mathbf{P}}_i \\ {}^v\dot{\mathbf{V}}_i \\ {}^v\dot{\mathbf{q}}_i \\ {}^i\dot{\mathbf{b}}_\omega \\ {}^v\dot{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} {}^v\mathbf{V}_i \\ -\mathbf{R}({}^v\mathbf{q}_i)\left({}^i\mathbf{a} -{}^i\mathbf{b}_a + \mathbf{v}_a\right) - {}^v\mathbf{g} \\ \frac{1}{2}\boldsymbol{\Omega}\left({}^v\mathbf{q}_i\right)\left({}^i\boldsymbol{\omega} - {}^i\mathbf{b}_\omega + \mathbf{v}_\omega\right) \\ \mathbf{v}_b \\ \mathbf{v}_g \end{bmatrix} \tag{3}$$

where the vectors $\mathbf{v}_a$, $\mathbf{v}_\omega$, $\mathbf{v}_b$ and $\mathbf{v}_g$ represent noise, ${}^i\mathbf{a}$ and ${}^i\omega$ are the measured linear acceleration and angular velocity and ${}^i\mathbf{b}_a$ is the linear acceleration's bias. Functions $\mathbf{R}(\cdot)$ and $\boldsymbol{\Omega}(\cdot)$ are the same as the ones used in [19]. For the KF to be able to handle the described model, we convert it to discrete time by integrating it over time and adding the previous state.

The light data used by the KF resorts to the model in (2) with added noise (w) to each measurement and with the poses in the inertial frame converted using the following transform:

$${}^l\mathbf{P}_p = {}^l\mathbf{R}_v\left[\,\mathbf{R}\left({}^v\mathbf{q}_i\right)\left({}^i\mathbf{R}_t\;{}^t\mathbf{P}_s + {}^i\mathbf{P}_t\right) + {}^v\mathbf{R}_i\right] + {}^l\mathbf{P}_v \tag{4}$$

We consider $\mathbf{f}(\cdot,\cdot,\cdot)$ the continuous-time inertial model, which receives as arguments the state, the inertial measurements and the inertial noise and $\mathbf{h}(\cdot,\cdot)$ to be the light model, which receives as arguments the state and the light noise.

1) Extended Kalman Filter: Since Vive's model is non-linear, we had to resort to its adaptations. The EKF (see [18]) uses a first order linear approximation of the system's model to estimate the parameters. This filter can be subdivided into two steps, the prediction step, in algorithm 1 (used here for the inertial data) and the update step, in algorithm 2, that resorts to the light data with the model in (2). To initialize the filter we use a simplified version of APE2. In order to increase the stability of the method, we only accept measurements from photodiodes in the lighthouses' Field of View (FoV) and we also use the quadratic error implemented with APE. We use $\mathbf{x}$ to represent the state, $\mathbf{S}$ for its covariance and $\mathbf{Q}$ and $\mathbf{R}$ for the noise of respectively the inertial and light measurements.

Algorithm 1: Extended Kalman Filter - Predict

```
$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1} + \int_{t_{k-1}}^{t_k} \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}\right) dt$
$\mathbf{A}_{k-1} = \frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{v})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_{k-1},\ \mathbf{u}=\mathbf{u}_{k-1},\ \mathbf{v}=\mathbf{0}}$
$\mathbf{G}_{k-1} = \frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{v})}{\partial \mathbf{v}}\Big|_{\mathbf{x}=\mathbf{x}_{k-1},\ \mathbf{u}=\mathbf{u}_{k-1},\ \mathbf{v}=\mathbf{0}}$
$\hat{\mathbf{S}}_{k|k-1} = \mathbf{A}_{k-1}\ \hat{\mathbf{S}}_{k-1}\mathbf{A}_k^\top + T_s^2\ \mathbf{G}_{k-1}\ \mathbf{Q}_{k-1}\ \mathbf{G}_{k-1}^\top$
```

Algorithm 2: Extended Kalman Filter - Update

```
$\tilde{\mathbf{C}}_k^j = \frac{\partial h(\mathbf{x},\mathbf{w})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\tilde{\mathbf{x}}_k^j,\ \mathbf{w}=\mathbf{0}}$
$\mathbf{W}_k = \hat{\mathbf{S}}_k\ \mathbf{C}_k^\top \left(\mathbf{C}_k\ \hat{\mathbf{S}}_k\ \mathbf{C}_k^\top + \mathbf{R}_k\right)^{-1}$
$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_k + \mathbf{W}_k\ \left(\mathbf{z}_k - \mathbf{C}_k\hat{\mathbf{x}}_k\right)$
$\hat{\mathbf{S}}_{k|k} = \left(\mathbf{I} - \mathbf{W}_k\ \mathbf{C}_k\right)\hat{\mathbf{S}}_k$
```

2) Iterated Extended Kalman Filter: The IEKF (see [20]) follows the exact same approach as the EKF, with the exception of the update step, that becomes the one in algorithm 3. The IEKF has the advantage of linearizing the model around an estimate of the updated variable. It performs particularly well when the system's measurement model is fully observable.

Algorithm 3: Iterated Extended Kalman Filter - Update

```
$j = 0,\ \tilde{\mathbf{x}}_k^j = \hat{\mathbf{x}}_k$
do
    $\tilde{\mathbf{C}}_k^j = \frac{\partial h(\mathbf{x},\mathbf{w})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\tilde{\mathbf{x}}_k^j,\ \mathbf{w}=\mathbf{0}}$
    $\tilde{\mathbf{W}}_k^j = \hat{\mathbf{S}}_k\ \tilde{\mathbf{C}}_k^{j\ \top}\left(\tilde{\mathbf{C}}_k^j\ \hat{\mathbf{S}}_k\ \tilde{\mathbf{C}}_k^{j\ \top} + \mathbf{R}_k\right)^{-1}$
    $\tilde{\mathbf{x}}_k^{j+1} = \tilde{\mathbf{x}}_k^j + \tilde{\mathbf{W}}_k^j\left(\mathbf{z}_k - \mathbf{h}\left(\tilde{\mathbf{x}}_k^j\right)\right)$
    $\epsilon = ||\tilde{\mathbf{x}}_k^{j+1} - \tilde{\mathbf{x}}_k^j||_2$
    $j = j + 1$
while $\epsilon < \delta$;
$\mathbf{C}_k = \tilde{\mathbf{C}}_k^j$
$\mathbf{W}_k = \hat{\mathbf{S}}_k\ \mathbf{C}_k^\top\left(\mathbf{C}_k\ \hat{\mathbf{S}}_k\ \mathbf{C}_k^\top + \mathbf{R}_k\right)^{-1}$
$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_k + \mathbf{W}_k\ \left(\mathbf{z}_k - \mathbf{C}_k\hat{\mathbf{x}}_k\right)$
$\hat{\mathbf{S}}_{k|k} = \left(\mathbf{I} - \mathbf{W}_k\ \mathbf{C}_k\right)\hat{\mathbf{S}}_k$
```

3) Unscented Kalman Filter: The UKF (see [21]) extends EKF's and IEKF's state by including the inertial and light model's noise. It uses the exact same light model as the other two filters and also the same restrictions to increase stability. Identically to the EKF and IEKF, it also has a prediction step, in 4, and an update step, in (5). We use κ as a tuning variable, ${}^a\mathbf{x}_k$ and ${}^a\hat{\mathbf{S}}_k$ to represent the extended state and its covariance, n for the size of this new state and $[\cdot]_i$ to extract the i^{th} column from its argument. The unscented transform allows this filter to better estimate the effect of a non-linear transform in a random variable.

C. Pose Graph Optimization with Window

One issue with APE is that it bundles all the light measurements into one single pose although each one has a different timestamp. The filters on the other hand are more sensitive to outliers. The Pose Graph Optimizer with Sliding Window (PGOSW) addresses both these issues by fusing PGO (see [22]) and MHE (see [23]). Since we only use the inertial data to constrain consecutive poses, the PGOSW resorts to a reduced version of the filter's state $\mathbf{x}$, which includes only

Algorithm 4: Unscented Kalman Filter - Predict

```
${}^{a}\mathcal{X}^{0}_{k-1} = {}^{a}\hat{\mathbf{x}}_{k-1},\ \mathcal{W}^{0} = \frac{\kappa}{n+\kappa}$
${}^{a}\mathcal{X}^{i}_{k-1} = {}^{a}\hat{\mathbf{x}}_{k-1} \pm \left[\sqrt{(n+\kappa)^{a}\hat{\mathbf{S}}_{k-1}}\right]_{i},\ \mathcal{W}^{i} = \frac{\kappa}{2(n+\kappa)}$
${}^{\mathbf{x}}\mathcal{X}^{i}_{k|k-1} = \mathbf{f}\left({}^{\mathbf{x}}\mathcal{X}^{i}_{k-1},\ \mathbf{u}_{k-1},\ {}^{\mathbf{v}}\mathcal{X}^{i}_{k-1}\right)$
$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=0}^{2n} \mathcal{W}^{i}\ {}^{\mathbf{x}}\mathcal{X}^{i}_{k|k-1}$
$\hat{\mathbf{S}}_{k|k-1} =$
$\sum_{i=0}^{2n} \mathcal{W}^{i}\left({}^{\mathbf{x}}\mathcal{X}^{i}_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}\right)\left({}^{\mathbf{x}}\mathcal{X}^{i}_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}\right)^{\top}$
```

Algorithm 5: Unscented Kalman Filter - Update

```
${}^{a}\mathcal{X}^{0}_{k} = {}^{a}\hat{\mathbf{x}}_{k},\ \mathcal{W}^{0} = \frac{\kappa}{n+\kappa}$
${}^{a}\mathcal{X}^{i}_{k} = {}^{a}\hat{\mathbf{x}}_{k} \pm \left[\sqrt{(n+\kappa)^{a}\hat{\mathbf{P}}_{k}}\right]_{i},\ \mathcal{W}^{i} = \frac{\kappa}{2(n+\kappa)}$
$\mathcal{Z}^{i}_{k} = \mathbf{h}\left({}^{\mathbf{x}}\mathcal{X}^{i}_{k},\ {}^{\mathbf{w}}\mathcal{X}^{i}_{k}\right)$
$\hat{\mathbf{z}}_{k} = \sum_{i=1}^{2n} \mathcal{W}^{i}\mathcal{Z}^{i}_{k}$
$\mathbf{P}^{vv}_{k} = \mathbf{R}_{k} + \sum_{i=0}^{2n} \mathcal{W}^{i}\left(\mathcal{Z}^{i}_{k} - \hat{\mathbf{z}}_{k}\right)\left(\mathcal{Z}^{i}_{k} - \hat{\mathbf{z}}_{k}\right)^{\top}$
$\mathbf{P}^{xz}_{k} = \sum_{i=0}^{2n} \mathcal{W}^{i}\left({}^{\mathbf{x}}\mathcal{X}^{i}_{k} - \hat{\mathbf{x}}_{k}\right)\left(\mathcal{Z}^{i}_{k} - \hat{\mathbf{z}}_{k}\right)^{\top}$
$\mathbf{W}_{k} = \mathbf{S}^{xz}_{k}\ \mathbf{S}^{vv-1}_{k}$
$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k} + \mathbf{W}_{k}\left(\mathbf{z}_{k} - \hat{\mathbf{z}}_{k}\right)$
$\hat{\mathbf{S}}_{k|k} = \hat{\mathbf{S}}_{k} - \mathbf{W}_{k}\ \hat{\mathbf{S}}^{vv}_{k}\ \mathbf{W}^{\top}_{k}$
```

the position, velocity and orientation. We used the following model:

$$\begin{bmatrix} {}^{v}\dot{\mathbf{P}}_{i} \\ {}^{v}\dot{\mathbf{V}}_{i} \\ {}^{v}\dot{\mathbf{q}}_{i} \end{bmatrix} = \begin{bmatrix} {}^{v}\mathbf{V}_{i} \\ -\mathbf{R}({}^{v}\mathbf{q}_{i})\left({}^{i}\mathbf{a} - {}^{i}\,\mathbf{b}_{a} + \mathbf{v}_{a}\right) - {}^{v}\mathbf{g} \\ \frac{1}{2}\mathbf{\Omega}\left({}^{v}\mathbf{q}_{i}\right)\left({}^{i}\boldsymbol{\omega} -\ {}^{i}\mathbf{b}_{\omega} + \mathbf{v}_{\omega}\right) \end{bmatrix} \tag{5}$$

This model is however in continuous time and needs to be converted to a discrete model. For that, we integrate it over the sampling time and add the result to the previous state. For the light model, the PGOSW uses (2) with the transform in (4).

This methods computes a series of consecutive poses and returns the last one as being the most recent. The cost function that defines this algorithm follows:

$$c_{PGOW} = \eta_{w} d_{w}^{2}\left(\mathbf{x}_{N-W}, \mathbf{x}'_{N-W}\right) + \sum_{k=N-W}^{N} d_{l}^{2}\left(\mathbf{x}_{k}, \boldsymbol{\alpha}_{k}\right) \tag{6}$$
$$+ \eta_{i} \sum_{k=N-W}^{N-1} d_{i}^{2}\left(\mathbf{x}_{k}, \mathbf{x}_{k+1}, \mathbf{u}_{k}\right)$$

where W is the size of the estimation window, η_w and η_i are weighting factors, $\mathbf{x}'_{N-W}$ is the previous estimate for state $\mathbf{x}_{N-W}$, $\boldsymbol{\alpha}_k$ are the light angles, $\mathbf{u}_k$ are the inertial measurements and $d_w^2(\cdot,\cdot)$, $d_l^2(\cdot)$ and $d_i^2(\cdot,\cdot)$ are functions that relate the current first pose with its former estimate, a pose with the light data and two poses with the inertial measurements. These distances are obtained as follows:

$$d_{w}^{2}(\mathbf{x},\mathbf{x}') = ||{}^{v}\mathbf{P}_{i} - {}^{v}\mathbf{P}'_{i}||_{2}^{2} + ||{}^{v}\mathbf{V}_{i} - {}^{v}\mathbf{V}'_{i}||_{2}^{2} + \angle^{2}\left(\mathbf{R}({}^{v}\mathbf{q}_{i})^{\top}\,\mathbf{R}({}^{v}\mathbf{q}'_{i})\right) \tag{7a}$$

$$d_{l}^{2}\left(\mathbf{x}, \boldsymbol{\alpha}\right) = \sum_{p=0}^{M}\left(\alpha_{p} - h_{p}\left({}^{v}\mathbf{P}_{i},\ {}^{v}\mathbf{q}_{i}\right)\right)^{2} \tag{7b}$$

$$d_{i}^{2}\left(\mathbf{x}_{k+1}, \mathbf{x}_{k}, \mathbf{u}_{k}\right) = d_{w}^{2}\left(\mathbf{x}_{k+1}, \mathbf{x}_{k+1|k}\right) \tag{7c}$$

where α_p is the light angle measurements associated to sensor p, $h_p(\cdot,\cdot)$ is the mentioned light model, $\angle(\cdot)$ is the norm of the axis-angle representation of the argument $\mathbf{x}_{k+1|k}$ is a predicted pose, obtained using the tracker's former pose $\mathbf{x}_k$, inertial data and the process's discretized inertial model (identical to the KFs).

In order to increase the robustness of PGOSW it includes the same steps as APE, where it only accepts light data in the lighthouse's FoV and uses the squared error threshold to reject poor estimates. The cost function used by the optimizer[11] was also robustified with the Cauchy loss function.

D. Algorithm for Environment Estimation

So far it was assumed that we knew the poses of the lighthouses with respect to the vive frame. There is however the need of a procedure that estimates these transforms. In [17] we have already introduced a method to obtain these transforms, the Algorithm for Environment Estimation (AEE). Here we also introduce a second iteration of this procedure, where is uses the new light model in (2). It also computes the gravity vector in the environment using the following expressions

$${}^{v}\mathbf{g} = -{}^{v}\mathbf{R}_{l}\ {}^{l}\mathbf{R}_{t}\ {}^{t}\mathbf{R}_{i}\left(\frac{1}{N}\sum_{k=1}^{N}\ {}^{i}\mathbf{a}_{k}\right) \tag{8}$$

where ${}^{i}\mathbf{a}_k$ is an individual accelerometer sample. To distinguish both methods, we will refer to them as AEE1 and AEE2.

E. Calibration Refiner

The AEE however only functions with the trackers in a stationary state, therefore the workspace will be calibrated with the same number of poses as the number of trackers available. The Calibration Refiner (CR) addresses that issue by using moving trackers to improve on the original calibration.

This method uses a cost function very similar to the one of the PGOSW:

$$c_{ref} = \sum_{k=1}^{N} d_{l}^{2}\left(\mathbf{x}_{k},\ {}^{v}\mathbf{T}_{l}\right) + \gamma_{i}\sum_{k=1}^{N-1} d_{i}^{2}\left(\mathbf{x}_{k+1}, \mathbf{x}_{k}\right) \tag{9}$$

where $\mathbf{T}$ represents a rigid-body transform. In this case the optimized variables are the states $\mathbf{x}_k$ and the poses of the

[11]Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm in the Ceres-Solver library, available at http://ceres-solver.org

lighthouses ${}^{v}\mathbf{T}_{l}$. The trackers' poses are however optimized here as a bystander. We also used the Cauchy loss function to reduce the effect of outliers and included a step to verify if an angle α complies with the lighthouse's FoV. Depending on whether the procedure uses (1) or (2), it is referenced as CR1 or CR2 respectively.

F. Precision Limit

To better evaluate the performance achievable by an estimator using Vive, we resort to CRLB, in [24, 25]. As our goal with Vive is using it as a motion capture system, the parameters we evaluate with CRLB are both the position and orientation (in an axis-angle representation), which are represented here by the state $\mathbf{x}$. We use both light and inertial measurements that can be computed as follows:

$$\begin{bmatrix}\boldsymbol{\alpha}\\ \mathbf{a}\\ \boldsymbol{\omega}\end{bmatrix} = \begin{bmatrix}\mathbf{h}(\mathbf{x})\\ -\mathbf{R}^{\top}\left({}^{v}\boldsymbol{\Phi}_{t}\right)\left({}^{v}\dot{\mathbf{V}}_{t} + {}^{v}\mathbf{g}\right)\\ \text{vec}\left[\dot{\mathbf{R}}\left({}^{v}\boldsymbol{\Phi}_{t}\right)\ \mathbf{R}^{\top}\left({}^{v}\boldsymbol{\Phi}_{t}\right)\right]\end{bmatrix} \tag{10}$$

where $\boldsymbol{\alpha}$, $\mathbf{a}$ and $\boldsymbol{\omega}$ are the angle alpha, the linear acceleration and the angular acceleration. $\text{vec}(\cdot)$ is the operator that converts its argument from a skew-symmetric matrix to a vector and ${}^{v}\mathbf{V}_{t}$ and ${}^{v}\boldsymbol{\Phi}_{t}$ are respectively the velocity and orientation of the tracker in the vive frame. The time derivatives of ${}^{v}\mathbf{V}_{t}$ and $\mathbf{R}\left({}^{v}\boldsymbol{\Phi}_{t}\right)$ can be obtained in the following way:

$${}^{v}\dot{\mathbf{V}}_{t_k} = \frac{1}{T_s}\left(\frac{{}^{v}\mathbf{P}_{t_k} - {}^{v}\mathbf{P}_{t_{k-1}}}{T_s} - \frac{{}^{v}\mathbf{P}_{t_{k-1}} - {}^{v}\mathbf{P}_{t_{k-2}}}{T_s}\right) \tag{11a}$$

$$\dot{\mathbf{R}}\left({}^{v}\boldsymbol{\Phi}_{t}\right) = \frac{\mathbf{R}\left({}^{v}\boldsymbol{\Phi}_{t_k}\right) - \mathbf{R}\left({}^{v}\boldsymbol{\Phi}_{t_{k-1}}\right)}{T_s} \tag{11b}$$

We also add gaussian noise to all the measurements in (10), with the following standard deviation: $0.002°$ for α if there are no outliers or $0.311°$ if there are and $0.024\ m/s^2$ for the linear acceleration and $0.040°/s$ for the angular velocity. This will result in a random variable for each measurement with an associated gaussian probability density function (PDF).

As we assume that these PDFs are independent, we can multiply them to create a joint PDF and use the following expression to obtain the CRLB associated to the system:

$$V_{CRLB}\left(\mathbf{x}_1, ..., \mathbf{x}_N\right) = -\left(E\left[\frac{\partial^2 \log p\left(\mathbf{Z}|\mathbf{x}_1, ..., \mathbf{x}_N\right)}{\partial(\mathbf{x}_1, ..., \mathbf{x}_N)^2}\right]\right)^{-1} \tag{12}$$

where $p\left(\mathbf{Z}|\mathbf{x}_1, ..., \mathbf{x}_N\right)$ is the joint PDF. This procedure is extremely slow and since the variance is small for the measurements, we can ignore the expected value and instead replace the measurements with their mean value. For static situations we can also partially compute the argument in the inversion operation in (12), and extend the information matrix with repeated elements to reduce the computation time.

As computing the CRLB is still very time consuming, for its other application, we ignore the inertial measurements, as they would be of minimal influence. In this application, we study the optimal placement for the lighthouses and its influence on the precision.

V. Results

In this section we provide the results obtained with the methods presented in the previous section. These not only evaluate Vive's precision and accuracy but also compute the optimal placement for the system's lighthouses. We used always the following parameters for the methods: for APE, the threshold constant $\gamma_\tau = 10^{-5}$; for the filters, the threshold constant $\gamma_\tau = 5\times 10^{-5}$, the inertial and light noise covariance matrices $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} = 10^{-6}\mathbf{I}$; for the PGOSW, a window $W = 7$, the weights $\eta_w = 7\times 10^{-4}$, $\eta_i = 1.0$ and the threshold constant $\gamma_\tau = 5\times 10^{-5}$; for the CR, an inertial weight $\gamma_i = 10.0$. These were chosen in order to improve the accuracy with the real system, with a small exception for the PGOSW, that resulted from a compromise between the solving time and the accuracy.

A. Lighthouse Placement

As described in section IV-F, we use the Vive's CRLB to obtain the optimal placement for the lighthouse's using Vive. As a tracker's photodiodes are distributed asymmetrically on its surface, we use a single photodiode and analyze its position's CRLB. This approach leads to the same comprehensive results.

When we are using a one lighthouse configuration, we are not able to compute the position of a photodiode. For a two lighthouse configuration that is already possible if they are not coincident. Besides that, using two lighthouses leads to a better precision when compared to using only one of them since there are more observations available. We therefore analyze the two ideal lighthouse setup.

To define the best configuration we considered the photodiode as the center of a spherical coordinate system and changed the azimuth (γ) and altitude (λ) of one of the lighthouses while maintaining the other static, keeping always both at distance of $1\ m$ from the photodiode. The obtained results are in figure 3, where we we can see from the summed variances that the optimal configuration consists of $\gamma = 90°$ and the λ taking any value or $\lambda = 90°$ and γ taking any value. This can be explained by the fact that the largest variance is in the estimate along the lighthouse's z axis, thus by arranging the lighthouses in order for both their z axes to be orthogonal improves the performance. Using a bigger domain for figure 3 would lead to identical results, due to the problem's symmetry. Although there is an interval of values where the summed variance is optimal, when we move in this interval we are actually improving the precision of one of the position's elements at the cost of the other. If we have into consideration the trace of the Fisher's information matrix however, we should choose $\lambda = 90°$ and $\gamma = 45°$ or $\gamma = 90°$ and $\lambda = 45°$.

Larger distances of the photodiode from the lighthouse also results in a worse precision. Therefore, the positioning of the lighthouses should be chosen in order to fill in the requirements while keeping the tracking volume as small as

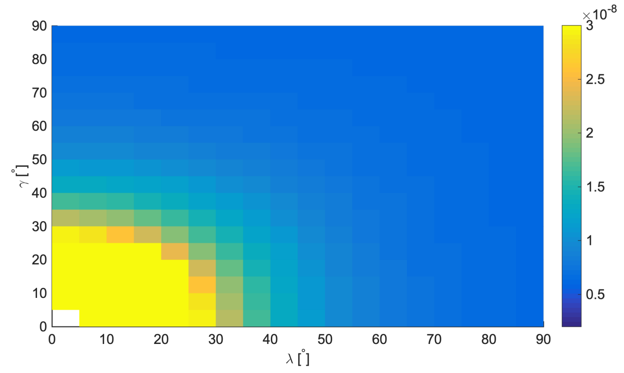

Fig. 3: Trace of the variace for the photodiode's position with different values of λ and γ.

possible. This can be see in table I, where d represents the distance from the photodiode to the lighthouses. To obtain the results in this table we used a simulator to prove that the optimal placements result in an improved precision.

d $[m]$	γ $[°]$	λ $[°]$	σ_P $[mm]$	σ_θ $[°]$
1	90	45	0.0075	0.0037
2	90	45	0.0112	0.0044
1	45	90	0.0107	0.0148
1	180	0	0.0358	0.0159
1	45	45	0.0221	0.0146
1	0	45	0.0237	0.0177

TABLE I: Standard deviation (square-root of the variance matrix) for the tracker's position and orientation using different lighthouse placements and APE to obtain the poses.

B. Accuracy Study

As was proven in [17], it is possible to improve Vive's accuracy as a motion capture system for robotic applications. The next step consists of comparing the performance of Vive using the pose estimation algorithms presented in section IV between themselves and against a state-of-the-art motion capture system.

To better control the performance comparison between the multiple methods, we use the simulator already mentioned in subsection V-A, that adds gaussian noise to the measurements, in order to make it closer to reality. The occlusion model for the photodiodes however only takes into account if the photodiode is facing the lighthouse.

For the first simulation, we resort to the ideal lighthouse (leading to APE1 and APE2 being the same) and to the optimal environment used to obtain figure 3 but with a distance of 2 m between the tracker and the lighthouses. With the tracker in a stationary state, we compute the square-root of the covariance matrix's trace for the position and orientation, leading to the results in table II. Analyzing the results, we can verify that no algorithm is on top of the CRLB. This can be explained by the optimizers precision limit for APE and PGOSW, by the non-linearities in the model for the filters [25] and by the fact that these methods were tuned to deal with the real system, where there is the presence of outliers and drifts. We used five thousand light samples and ten thousand inertial samples to estimate the standard deviation associated to the tracker's pose. Nevertheless, the best performing methods are PGOSW and APE.

	σ_P $[mm]$	σ_θ $[°]$
CRLB	0.0056	0.0012
APE	0.0112	0.0044
EKF	0.0193	0.0101
IEKF	0.0193	0.0101
UKF	0.0170	0.0066
PGOSW	0.0101	0.0023

TABLE II: Precisions results for the tracker's position and attitude for the implemented algorithms with a distance of two meters between tracker and lighthouses.

For the second simulation, we analyze the performance of Vive with the trackers in motion with the same optimal configuration as before. The trajectory used to test consisted in a tracker rotating around a central point. The circumference created from the trajectory has a 1 m radius and a duration of 20 s, with the tracker spinning at 0.05 Hz. In figure 4, we can see that APE associating multiple light samples to a single pose does not benefits it anymore. This effect will only increase for higher velocities. Besides that, PGOSW's performance is also degraded by the optimization algorithm's attempt to place the new estimate too close to the previous one. The filters all have a similar performance, with the UKF being able to the outperform the other two due to handling better the non-linearities in the system in the orientation.

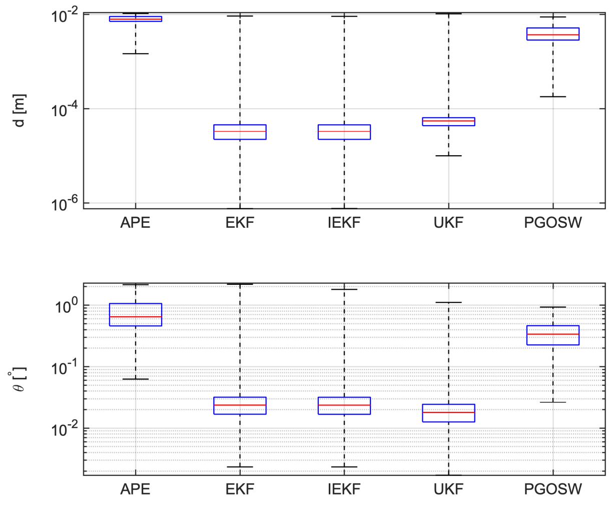

Fig. 4: Position and orientation error for the simulated trajectory.

We also used a simulated dataset to test the performance of the calibration procedures, using the same optimal lighthouse placement. To compare the estimated environment to its ground truth, we used the L2 norm of the error in the position difference of the lighthouses, the error in the orientation difference of the lighthouses and the L2-norm of the estimated gravity and actual gravity. The results in table III show us that both calibration procedures have a very similar performance, which can be explained by the fact that CR spends most of the optimization time improving the poses of the tracker in the vive frame. This method is therefore more useful when environment estimation obtained with AEE is not ideal. The gravity estimation is identical in both methods as the CR does not recompute it.

	$\Delta P\ [mm]$	$\Delta\theta\ [°]$	$\Delta g\ [mm]$
AEE	0.280645	0.008	69.64
AEE + CR	0.34	0.009	69.64

TABLE III: Position, orientation and gravity error resultant from the calibration procedures.

The final step consists of evaluating how Vive performs against the high-accuracy motion capture system OptiTrack using the developed methods and algorithms. We do not compare against Vive's baseline algorithms because, since we do not have access to their raw data, we would be comparing different datasets, which was already done in [17]. This comparison is done based on a set of tests where we move a Vive controller according to a certain trajectory, while OptiTrack tracks a cluster of markers attached to the controller.

Since we are not able to compare the raw poses returned by OptiTrack and Vive, due to the tracked object not being the same and the reference frames being different, we have to resort to a calibration procedure to convert the poses' frames. As this is identical to the hand-eye calibration problem (see [26, 27]) we apply the same approach using a non-linear optimizer[12]. Using five recorded datasets to assess the calibration, we obtained an acceptable accuracy for the transforms, with an average difference between the centroids of the trajectories of $2.4\ mm$, a mean rotation between the trajectories of $0.7°$ (both with the Kabsch algorithm, in [28]), and a geodesic L2 mean (see [29]) of the orientation differences of $0.4°$. The recorded datasets used in this calibration were similar to both figures 5 and 6.

We started first by only using the AEE to calibrate the environment and to compare the results between the different tracking methods. We collected five datasets and aggregated them to improve the readability, with the results visible in figure 7. Each of the datasets contained around thirty seconds of data from both Vive and OptiTrack.

We also used the CR method to optimize the calibration estimated with AEE. It starts with the calibration environment estimated to obtain the results in figure 8. We used the exact same datasets to arrive at these results and also aggregated the data to improve readability.

As the standard deviation associated to each method is high when compared to the performance differences between the methods, we resorted to statistical hypothesis testing. For that we used the typical significance level of 5% and considered

[12]Levenberg-Marquardt algorithm in the Ceres-Solver library, available at http://ceres-solver.org

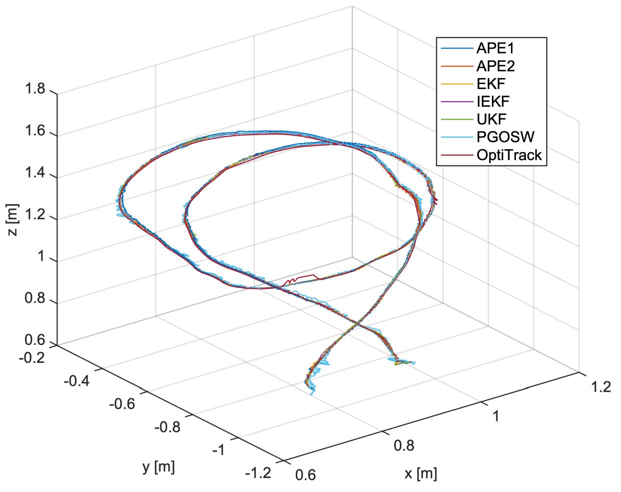

Fig. 5: Estimated trajectories for dataset 1.

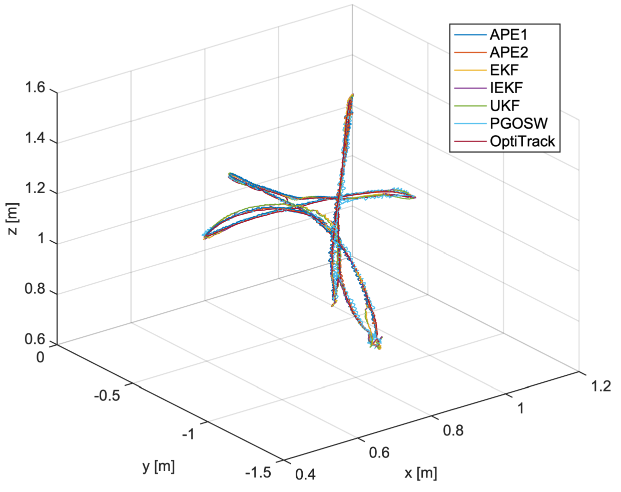

Fig. 6: Estimated trajectories for dataset 3.

the alternative hypothesis to be either the mean error of two methods being different or the mean error of one being larger than the other.

Comparing both the results from AEE and CR with the statistical hypothesis testing, we can say that APE1 is the only method among the implemented where the hypothesis of the means being equal can be rejected. Comparing the maximum error between the correspondent methods leads to the same conclusions. This is explained by the fact that the CR improves the performance in situations where the calibration obtained with AEE is not great, which is exactly what happens with APE1 and AEE1, where the correction parameters are not included.

Applying the statistical hypothesis testing to the comparison between the average behavior of the corresponding methods,

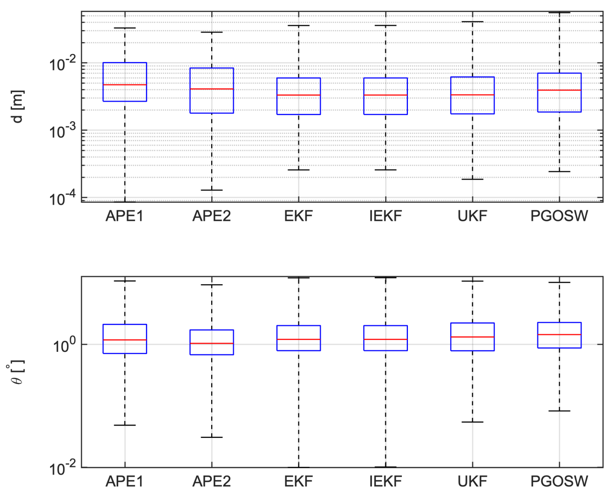

Fig. 7: Aggregated position and orientation error using AEE for the calibration.

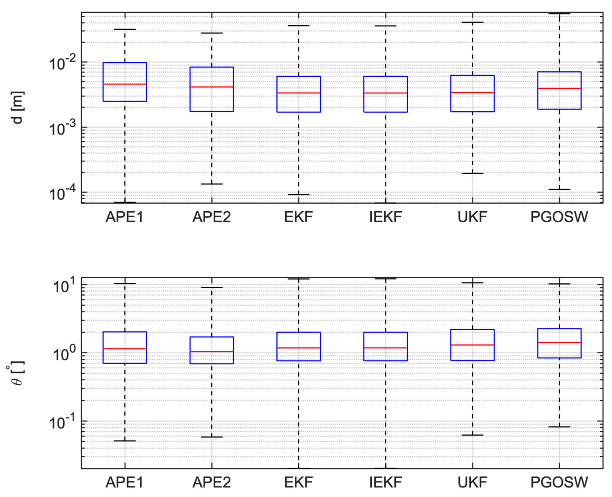

Fig. 8: Aggregated position and orientation error using AEE and CR for calibration.

we reach the conclusions that using the correction parameters improves the results, as the mean of APE2's error can be stated as lower than APE1 with the statistical significance level of 5%, that the filters outperform the others in the position estimation, and that the APE2 has the best performance in the orientation estimation. For the orientation estimation the EKF and IEKF, on average slightly outperform the UKF. We can also state that on average the PGOSW is the worst estimator for the orientation due to its noise sensitivity.

The mean error does not fully evaluate the performance. To complement it, we can also analyze the maximum error, which shows us how each method deals with the outliers. We can therefore state that the filters and PGOSW are the ones that perform worse around outliers. The APE2, by bundling multiple measurements and weighting down outliers with the Cauchy loss function is able to outperform the other methods in this situation and have therefore a better overall performance in regular tracking conditions. The UKF is also able to outperform the other filters in the maximum error associated to the orientation.

From the achieved results, we can say that if we use the APE2, the maximum error for the position will not be larger than a $3\ cm$, being in average in the low millimeter range, while for the orientation will always be lower than $10°$. If the velocity is higher, this would no longer apply, as APE2's performance would degrade. In this case, the filters are the best alternative, as they smoothen better the trajectories and are able to achieve a maximum error of $4\ cm$ for the position and of $11°$ for the orientation. The PGOSW is the most sensitive to the noise. However its performance could be improved, but at the cost of an unacceptably slow estimation time.

As we encountered outliers in the trajectories returned by OptiTrack, we were forced to remove certain parts from it. If we had included them, the maximum error would increase to over $0.1\ m$ and $80°$ for all the implemented algorithms. This would make us say that Vive is more accurate than OptiTrack, but we must have into account that OptiTrack was configured to track in a large room while Vive used an optimal setup. Nevertheless, we can state that Vive achieves a remarkable accuracy for a system that has a cost 50 times lower.

VI. Conclusion

As we do an analysis of the state-of-the-art ground systems for full poses, we show that in order to have access to accurate pose estimates, an expensive commercial motion capture system is required. If we try to find a more affordable system we find that they are usually tuned for different applications and therefore are not adequate for robot tracking.

Vive is able to address precisely that as cost-effective solution for motion capture in robotics applications. It tends however to smooth severely the recorded trajectories while sacrificing the accuracy. APE1 and AEE1 show that by changing the weights of the inertial measurements and the outlier rejection policy, we can make Vive a better motion capture system for roboticists.

To increase Vive's performance, we studied Vive's precision model with the CRLB and designed a method to automatically choose the best lighthouse placement. We also introduced the lighthouses' distortion correction parameters and implemented a set of new methods (APE2, EKF, IEKF, UKF, PGOSW, AEE2 and CR) in order to evaluate their advantages. We also reintroduced with some methods (all but APE2 and AEE2) the inertial measurements, but with an adjusted weight. In the end we compare Vive's performance using the implemented methods against OptiTrack. This new implementation is able to achieve on average a millimeter magnitude error. The maximum error stayed around $2\ cm$ for APE2 and $5\ cm$ for the EKF, IEKF, UKF and PGOSW.

References

[1] M. Windolf, N. Götzen, and M. Morlock, "Systematic accuracy and precision analysis of the video motion capturing systems - exemplified on the vicon-460 system," *Journal of biomechanics*, no. 12, pp. 2776–2780, 2008.

[2] S. Soylu, A. Proctor, R. P. Podhorodeski, C. Bradley, and B. Buckham, "Precise trajectory control for an inspection class rov," *Ocean Engineering*, vol. 111, pp. 508–523, 2016.

[3] L. Noldus, A. Spink, and R. Tegelenbosch, "Ethovision: A versatile video tracking system for automation of behavioral experiments," *Behavior Research Methods, Instruments, & Computers*, vol. 33, no. 1, pp. 398–414, 2001.

[4] S. Fry, M. Bichsel, P. Müller, and D. Robert, "Tracking of flying insects using pan-tilt cameras," *Journal of Neuroscience Methods*, vol. 101, no. 1, pp. 59–67, 2000.

[5] K. Kurihara, S. Hoshino, K. Yamane, and Y. Nakamura, "Optical motion capture system with pan-tilt camera tracking and realtime data processing." in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2002, pp. 1241–1248.

[6] G. D. Hager and K. Toyama, "X vision: A portable substrate for real-time vision applications," *Computer Vision and Image Understanding*, vol. 69, no. 1, pp. 23 – 37, 1998.

[7] A. Sirota, "Robotracker - a system for tracking multiple robots in real time," Technion Israel Institute of Technology, Tech. Rep., 2004.

[8] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. Sorrenti, and P. Taddei, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Autonomous Robots*, vol. 27, no. 4, p. 353, 2009.

[9] T. Lochmatter, P. Roduit, C. M. Cianci, N. Correll, J. Jacot, and A. Martinoli, "Swistrack - a flexible open source tracking software for multi-agent systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 4004–4010.

[10] C. Anthes, R. Hernandez, M. Wiedemann, and Kranzlmüller, "State of the art of virtual reality technologies," in *IEEE Aerospace Conference*, Big Sky, Montana, United States, 2016, pp. 1–19.

[11] S. LaValle, A. Yershova, M. Katsev, and M. Antonov, "Head tracking for the oculus rift," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 187–194.

[12] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[13] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 2969–2976.

[14] L. Kneip, H. Li, and Y. Seo, "Upnp: An optimal o(n) solution to the absolute pose problem with universal applicability," in *Computer Vision (ECCV)*. Springer, 2014, pp. 127–142.

[15] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, p. 155, 2009.

[16] W. Geiger, J. Bartholomeyczik, U. Breng, W. Gutmann, M. Hafen, E. Handrich, M. Huber, A. Jackle, U. Kempfer, H. Kopmann *et al.*, "Mems imu for ahrs applications," in *IEEE/ION Symposium on Position, Location and Navigation*, 2008, pp. 225–231.

[17] M. Borges, A. Symington, B. Coltin, T. Smith, and R. Ventura, "Htc vive: Analysis and accuracy improvement," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 2610–2615.

[18] R. Ventura, "Derivation of the discrete-time kalman filter," Instituto Superior Técnico, Tech. Rep., 2011.

[19] D. Schinstock, "Gps-aided ins solution for openpilot," Kansas State University, Tech. Rep., 2014.

[20] T. Lefebvre, H. Bruyninckx, and J. Schutter, "Kalman filters for nonlinear systems: a comparison of performance," *International journal of Control*, vol. 77, no. 7, pp. 639–653, 2004.

[21] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068, 1997, pp. 182–194.

[22] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[23] J. Rawlings, "Moving horizon estimation," *Encyclopedia of Systems and Control*, vol. 50, pp. 1–7, 2013.

[24] S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1993, ch. 3, pp. 27–81.

[25] N. G. Branko Ristic, Sanjeev Arulampalam, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004, ch. 4, pp. 67–80.

[26] M. Shah, R. D. Eastman, and T. Hong, "An overview of robot-sensor calibration methods for evaluation of perception systems," in *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*. ACM, 2012, pp. 15–20.

[27] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[28] L. Kavraki, *Geometric methods in structural computational biology*. Rice University, 2009.

[29] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International journal of computer vision*, vol. 103, no. 3, pp. 267–305, 2013.