HTC Vive: Analysis and Accuracy Improvement

Miguel Borges[⋆⋄], Andrew Symington[†], Brian Coltin[†], Trey Smith[‡], Rodrigo Ventura[⋆]

***Abstract*— HTC Vive has been gaining attention as a cost-effective, off-the-shelf tracking system for collecting ground truth pose data. We assess this system's pose estimation through a series of controlled experiments where we show its precision to be in the millimeter magnitude and accuracy to range from millimeter to meter. We also show that Vive gives greater weight to inertial measurements in order to produce a smooth trajectory for virtual reality applications. Hence, the Vive's off the shelf algorithm is poorly suited for robotics applications such as measuring ground truth poses, where accuracy and repeatability are key. Therefore we introduce a new open-source tracking algorithm and calibration procedure for Vive which address these problems. We also show that our approach improves the pose estimation repeatability and accuracy by up to two orders of magnitude.**

I. Introduction

The HTC Vive is a consumer headset and accompanying motion capture system designed for virtual reality (VR) applications [1]. Motion capture describes the process of estimating absolute position and orientation — or pose — in real-time, and has many applications in film, medicine, engineering [2], and notably robotics.

The Vive system is comprised of *lighthouses* that emit synchronized light sweeps, and *trackers* that use photodiodes to measure light pulse timings as a proxy for estimating the horizontal and vertical angles to lighthouses. The trackers fuse angle measurements from a bundle of rigid photodiodes together to estimate the pose using a technique similar to Angle-of-Arrival [3]. The tracker also has access to motion data from an incorporated Inertial Measurement Unit (IMU) to maintain a smooth and continuous trajectory.

The Vive system provides a compelling means of obtaining ground truth data for roboticists: it is more affordable than competing technologies, it is straightforward to set up and use, and a Robotics Operating System (ROS) driver already exists for integration with an ecosystem of robotic tools.

For the reasons above, Vive was chosen as a source of ground truth for testing the Astrobee robots (Fig. 1). Astrobees [4] are free-flying robots that will be deployed to the International Space Station in 2018 to be used as a research platform for free-flying robots[1]. Ideally, the system should exhibit error in the millimeter range in order to be able to benchmark Astrobee's localization algorithms [5], to test Astrobee's robotic arm and to improve on the available tracking systems (VisualEyez with an in-house developed pose estimation algorithm and QR codes with overhead camera).

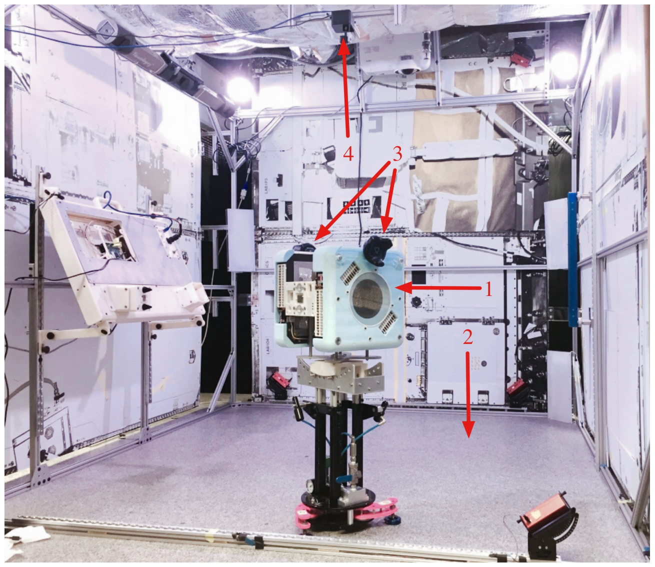

Fig. 1. Astrobee (1) shown on a level granite surface (2), which is used to simulate a 2D microgravity environment. The prototype has trackers mounted to its port and starboard sides (3), and a single lighthouse (4) mounted overhead.

The first contribution of this paper is an analysis of Vive's static precision and dynamic precision and accuracy. We show that although the original system has a sub-millimeter precision with the trackers in a static state, when that state is dynamic, the precision worsens by one order of magnitude. We also show experimentally that the accuracy of this system can vary from a few millimeters up to a meter in a dynamic situation.

We attribute the high error to the closed-source fusion algorithm giving higher weight to the inertial measurements, thus minimizing jitter to the VR user. Motivated by this and by not having access to the source code of Vive's algorithms, the second contribution of this paper is a set of algorithms for Vive that improve on the accuracy and stability while providing an open-source platform that is easy for the user to change. These algorithms are used to compute the trackers' poses and for the calibration procedure that relates the lighthouses with the user's workspace. We show that our tracking methods, although less smooth, are able to outperform Vive's built-in algorithms in accuracy by up to two orders of magnitude.

[⋆]Institute for Systems and Robotics - Lisboa, Instituto Superior Técnico, Universidade de Lisboa
[†]SGT Inc., NASA Ames Research Center
[‡]NASA Ames Research Center
[⋄]`miguel.r.borges@tecnico.ulisboa.pt`

[1]Astrobee flight software available open source at `https://github.com/nasa/astrobee`

II. Motion Capture

Vive is one of many motion capture systems available on the market. Examples of other systems include VICON, OptiTrack and VisualEyez. VICON and OptiTrack use cameras to track reflective markers illuminated by infrared light sources. VICON quotes accuracy levels of up to 76 μm and precision (noise) of 15 μm in a four camera configuration [6]. OptiTrack claims that its system can achieve accuracy levels of less than 0.3 mm for robotic tracking systems. VisualEyez uses a three-camera system to track active LED markers, and is reported to have millimeter-level precision [7]. The key issue with these tracking systems is that they are prohibitively expensive for general use.

Reliable motion capture is an essential component of an immersive VR experience. As the technology grows in popularity so the cost of equipment falls. We are now at a point where off-the-shelf VR devices are providing a feasible alternative for motion capture in the context of robotics. Examples of VR systems that offer motion capture include Oculus Rift [8] and HTC Vive.

HTC Vive's pose estimation has a similar working principle to the Angle of Arrival (AoA) localization techniques [9] used in Wireless Sensor Networks (WSN). AoA based localization resorts to arrays of antennas to estimate the angle of the received signal. From this interaction between multiple nodes, it is possible to estimate their location. Vive's trackers however, estimate the angle of the lighthouse through a time difference, as is explained in the next section.

In [1], the authors evaluate Vive's accuracy, however they focus on research with the headset. They do not mention how the trackers and controllers behave. These two devices are more appealing for roboticists. It is therefore unclear how Vive behaves as a ground truth tracking system for robotic applications.

III. Problem Description

We intend to use Vive as a means of obtaining Astrobee's ground truth. Our desired accuracy is one order of magnitude greater than the robot's current localization algorithm, which is expected to have an accuracy in the centimeter range.

As a tracking system, Vive's desired outputs are the poses of the trackers. In order to compute this, the system has as inputs inertial measurements (from the IMU in each tracker) and light data (from the photodiodes in the trackers).

The light data may be of two types — a synchronization flash or an infrared sweeping plane (red line in Fig. 2). Both these light signals are emitted by the lighthouse (the fixed base station). Every cycle starts with the synchronization pulse, modulated with low-rate metadata containing calibration parameters for correcting the sweeping plane, followed by an infrared rotating planar laser. The tracker's photodiodes detect both these signals and are able to estimate the angle (α in Fig. 2) between the lighthouse's normal vector and the photodiode with the time difference because the laser rotates at a constant angular velocity. This cycle happens for a horizontally and a vertically rotating laser. From both these angles, Vive can estimate the tracker's absolute pose.

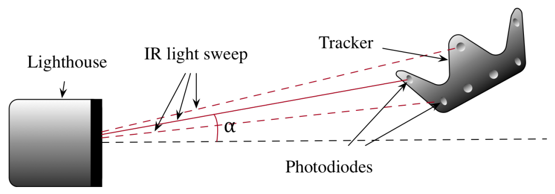

Fig. 2. Side view of HTC Vive working principle with α being the angle between a sweeping plane and the normal vector to the lighthouse.

The inertial data is composed of linear accelerations and angular velocities. But since this comes from a consumer grade IMU, the measurements are very noisy.

There are multiple frames associated with this problem. Both the lighthouse and the tracker have their own frames, l and t respectively. For clarity, the lighthouse is represented as l_i instead of l in Fig. 3, where i is its index. An auxiliary frame vive, v, is selected to always be coincident with one of the lighthouses' frames — chosen during the calibration procedure. This procedure allows the system to relate the poses of the lighthouses, in the case multiple lighthouses are being used. It also allows the user to choose the world frame w. The final output of Vive is a rigid-body transform between a tracker frame and the world frame, represented by a red arrow in Fig. 3.

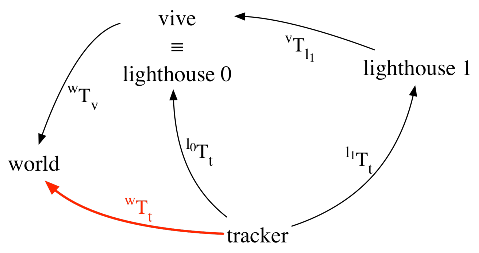

Fig. 3. Frames involved in Vive's pose estimation.

We use the standard notation for transforms from [10], where $^{a}T_{b}$ is a rigid-body transform from frame b to a (or pose of b in frame a). We will also be mentioning $^{a}P_{b}$ and $^{a}\tilde{P}_{b}$, which are the position of frame b relatively to a in cartesian and homogeneous coordinates, respectively.

IV. Vive Analysis

In order to evaluate the performance of Vive as ground truth tracking system, we performed two sets of tests where we measured the system's precision and accuracy. For these experiments, two lighthouses were attached to the top of consecutive walls of a two meter cubic workspace with a granite level ground surface. The lighthouses were facing 45° downwards in order to maximize our working volume.

A. Stationary Performance

First, we assess how precise the system is in a static configuration. We placed a tracker on a still surface and recorded the returned poses. The position's standard deviation recorded is always less than 0.5 mm, as we show in Table I of section VI, meaning that Vive achieves sub-millimeter precision in a static configuration. We will see however, that the performance degrades with the trackers in motion.

B. *Dynamic Performance*

In order to evaluate the performance of the system in the mentioned state, we took advantage of the fact that Astrobee is in a support that floats in a granite level surface with compressed air thrusters [11]. While the robot floats across the granite surface, the height of any part is constant because the surface has been precisely machined to be leveled. This surface is used to simulate a confined 2D zero-gravity environment. As we show in Table II in section VI, we evaluate the deviation between tracker's sample points and a plane fit to its trajectory. We also provide an orientation estimation evaluation in the same table.

The results show how unstable the system is and how the accuracy can vary from 1 mm to 43 mm and even 802 mm in the worst case. In order to use Vive to benchmark Astrobee's localization algorithms and also to evaluate its robotic arm's grasp, our desired accuracy is in the millimeter magnitude. We suspect that in order to improve the experience for the VR user, this system gives the inertial data a high weight in order to reduce the jitter. This pushed us towards developing our own algorithms, tuned for robotic applications.

V. Tracking Algorithms

The new software platform, in Fig. 4, is composed of two main ROS nodes, the Bridge and the Server. The Bridge uses deepdive[2] to pull light and IMU data from the tracker through a USB connection and then sends it to the Server. The Server then passes that data to the Calibrator or APE (Absolute Pose Estimator), depending on its current state. These states can be *Calibrating* — determining the relative rigid-body transforms between the lighthouses and the world frame using the Calibrator — or *Tracking* — real time pose solving using the APE.

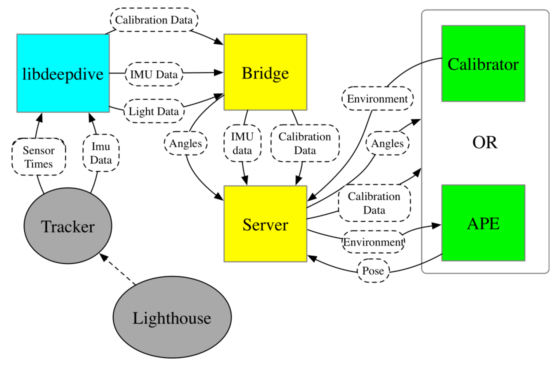

Fig. 4. Diagram of the system.

[2]deepdive is available open source at https://github.com/asymingt/deepdive

A. *Pose Tracking*

The APE is our algorithm that estimates full poses of a tracker in real-time, using only light data. It finds the best fit between a pose and the available data with a non-linear least-squares method that incorporates the model of the system. Inertial data is also recorded, however we did not use it because first we are looking for a satisfactory result. The correction parameters modulated in the synchronization pulses are also not used for the same reason as the inertial data. Including the IMU data and the sweeping laser correction parameters in the algorithm will therefore be left for future work.

The model of this system is a function that returns a horizontal and a vertical angle based on the relative position between a tracker's photodiode and the lighthouse. This position is obtained through a series of rigid-body frame transforms that convert the photodiode's coordinates from the tracker's frame to the lighthouse's:

$$^{l}\tilde{P}_p = {}^{l}T_v \; {}^{v}T_t \; {}^{t}\tilde{P}_p \tag{1}$$

where p is the photodiode, l the lighthouse, v the vive frame and t the tracker. The position of the photodiode ${}^{t}\tilde{P}_p$ is already known from the start due to a factory calibration.

In order to obtain the relative horizontal and vertical angle between a photodiode and the lighthouse's normal vector, we have to use the three photodiode's coordinates separately as in (2). The ${}^{l}P_p^x$, ${}^{l}P_p^x$ and ${}^{l}P_p^x$ terms are the x, y and z coordinates of ${}^{l}P_p$ and the top expression is for the horizontal axis while the bottom one is for the vertical one. This expression is a formalization of what was explained in a previous section with Fig. 2.

$$h\left({}^{l}P_p\right) = \begin{cases} \arctan\left(\frac{{}^{l}P_p^y}{{}^{l}P_p^z}\right) \\ \arctan\left(\frac{{}^{l}P_p^x}{{}^{l}P_p^z}\right) \end{cases} \tag{2}$$

In order to compute the pose of the tracker we use a sum of of squared-differences between the photodiode's recorded angles (α_p) and the estimated angles. This is a non-convex optimization problem however, using an optimizer [3], we are able to get results quickly enough to achieve real-time tracking. The cost function is the following:

$$f_{APE} = \sum_{l=1}^{M} \sum_{p=1}^{N} \left[h_{p,l}\left({}^{v}T_t\right) - \alpha_p\right]^2 \tag{3}$$

where $h_{p,l}\left(\cdot\right)$ is function (2) with (1) as the input argument after converting it to cartesian coordinates.

Our cost function uses the data from all the lighthouses at the same time in order to increase the stability of the solution, however for the horizontal axis, we have to negate the recorded angles ($-\alpha_p^{horizontal}$) due to the rotation direction of the lighthouse's laser.

Our algorithm also takes advantage of Vive's high sampling rate by initializing the optimizer at the last computed

[3]Ceres-Solver is available at http://ceres-solver.org

pose as a means of making the estimation process faster. For the first estimation done by the algorithm, we use an arbitrary starting pose in front of one lighthouses, to make sure it doesn't converge to a pose behind it.

In order to prevent outliers and old data from influencing the estimation we included the following restrictions: all measured angles with magnitude greater than 60 degrees are rejected, there must be at least 4 measured angles from the most recently detected lighthouse and all samples older than 50 ms are not used in the case they are not from the most recently detected lighthouse, otherwise the APE skips the estimation of this pose.

The cost function (3) is fairly complex and the optimizer may sporadically diverge or converge into a local minimum. In order to prevent wrong estimations, we included one more verification before providing the solution to the user: it checks the cost function's cost and if it is bigger than a threshold linearly related with the number of observed angles, the algorithm rejects the pose and waits for new data.

These constraints improve APE's stability but they also lead to ignoring poses (loss of tracking) on the edges of the workspace, where most of the photodiodes are occluded from the lighthouse.

All the poses are estimated by the algorithm in the vive frame instead of the world frame. The vive frame is an auxiliary frame between the lighthouses' frames and the world frame. The poses in the world frame are computed through ROS because the world frame is determined prior to the pose estimation, as is mentioned in subsection V-B.

B. Calibration Procedure

When the Vive system is installed, the lighthouses are individually mounted wherever it is convenient for the user, so the registration from lighthouse to lighthouse and from lighthouse to the world frame of interest is initially unknown. Therefore we created a procedure that addresses this issue.

Our calibration procedure consists of a concatenation of rigid-body transforms (4) that leads to the relative poses of the lighthouses. It assumes that the trackers are static leading to a more accurate process.

$$^{w}T_l =^{w} T_b \, ^{b}T_t \, ^{t}T_l \tag{4}$$

For Astrobee tracking, as in many robotic applications, we have multiple trackers rigidly mounted to the robot chassis, pointing in different directions, to provide improved tracking coverage. We use the combined tracker information to estimate the position of the robot's body frame (b). The user should specify the mounting geometry of the trackers on the robot as body-to tracker relative poses $^{b}T_t$. The user also registers the world frame by taking Vive measurements with the robot body frame fixed at a known pose $^{w}T_b$.

In the time interval between the start and end of the data acquisition, our calibrator records light data at 30 or 60 Hz (depending on the lighthouse's mode). After completing the data acquisition, it starts by computing an initial estimate of the relative pose between the trackers and the lighthouse, using the following cost function:

$$\hat{f}_{Cal} = \sum_{p=1}^{N} \left[h_p\left(^{l}T_t\right) - \alpha_p\right]^2 \tag{5}$$

where h_p is function (2) using as its input:

$$^{l}\tilde{P}_p = \, ^{l}T_t \, ^{t}\tilde{P}_p \tag{6}$$

This estimate is used to initialize the final cost function, where we compute, simultaneously, the pose of each lighthouse in the vive frame, but this time with all the trackers at the same time, as in:

$$f_{Cal} = \sum_{t=1}^{K}\sum_{l=1}^{M}\sum_{p=1}^{N} \left[h_{p,l,t}\left(^{w}T_l\right) - \alpha_p\right]^2 \tag{7}$$

where the function $h_{p,l,t}$ is similar to (2), however the input is $^{w}T_l$ which can be obtained from the rigid-body transform $^{l}T_w$. To obtain the input of the original function, we resort to the following expression:

$$^{l}\tilde{P}_p = \, ^{l}T_w \, ^{w}T_t \, ^{t}\tilde{P}_p \tag{8}$$

After having all the lighthouses' poses computed, the procedure chooses the vive frame as one of the lighthouses' frames and converts the lighthouses' poses to this new auxiliary frame. We decided to use this frame in order to preserve the frame hierarchy in the original ROS driver.

VI. Results

In order to evaluate Vive's performance, we designed two experiments where we assess the system in different situations. Since we do not have access to Vive's *baseline* algorithm's (from now on referenced as baseline) input data, in order to compare it with our *proposed* algorithm (from now on referenced as proposed), we will have to use different datasets collected in similar conditions. We used however, different lighthouse configurations for each dataset. We used two set-ups (shown in Fig. 5): for the baseline algorithms, we used the adjacent walls configuration, which provided a position standard deviation of 5 mm against 93 mm and a maximum deviation of 28 mm against 6271 mm of the other configuration; the proposed algorithm performed well for the first lighthouse set-up (lighthouses on opposing walls), eliminating the need to reconfigure their locations.

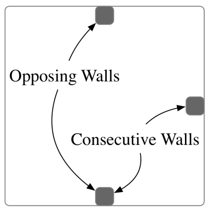

Fig. 5. Configuration of the lighthouses in the granite surface's workspace.

A. Static State Results

We started with a stationary state comparison between algorithms, as described in section IV, where we evaluate the estimated pose's standard deviation (maximum standard deviation of the 3D position and standard deviation of the angle resorting to an axis-angle representation of the orientation). In this experiment, the tracker was static and at a distance of 1-2 m from the lighthouses. Table I contains the results we obtained for each of the approximately 30 s dataset.

TABLE I
STANDARD DEVIATION OF THE POSE IN A STATIC STATE.

Algorithm	Dataset	σ_{Position} [mm]	$\sigma_{\text{Orientation}}$ [°]
Baseline	s1	0.417	0.00300
	s2	0.151	0.00586
	s3	0.260	0.000476
	s4	0.214	0.0023
	s5	0.168	0.00687
Proposed	s6	4.960	0.010
	s7	0.0875	0.000216
	s8	1.149	0.000476
	s9	10.052	0.0447
	s10	0.851	0.0030

Comparing the average position's standard deviation from Vive's built-in algorithms (0.242 mm) and from our algorithms (3.419 mm) we can clearly conclude that the former outperforms the latter in a stationary experiment. These results are explained by the fact that our algorithm does not use the correlation between consecutive poses. The inertial measurements also help the baseline algorithm with this correlation. However, this is only valid for a static situation, and the trackers will not be in a constant pose while tracking the robot.

B. Dynamic State Results

The second experiment consists of tracking with the same algorithms but with the trackers in motion, as described in section IV. As Astrobee floats in a perfectly flat surface, it's trajectory should be a perfect plane.

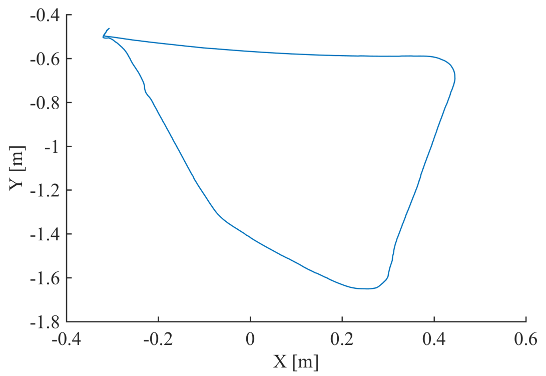

Fig. 6. XY view of the path generated with tracker 3 from dataset d5.

In this set of experiments (Table II), we assess the distance between the trajectory generated by each tracker and a plane fit to the estimated poses and also the angle between the plane's normal vector and the same vector attached to the tracker's frame in the first instant. We will compare the precision (through the standard deviation) of both algorithms again but for this different situation and now we'll also include an accuracy assessment (through the plane's maximum deviation — ϵ_{max} — and average deviation — $\bar{d}$). For these tests, trackers 1 and 2 were mounted on the sides of Astrobee, except for dataset d4, where they were attached to the top of the robot, as was tracker 3 when it was used. We include in the results table a reference to the location of the trackers on the robot for each dataset (s for starboard, p for port and t for top). All the datasets have a duration of 40-120 s and Astrobee performed a trajectory similar to the one in Fig. 6, manually controlled, with an average linear velocity of 1-6 cm/s.

TABLE II
DEVIATION FROM THE FITTED PLANE.

Algorithm	Dataset	Tracker	σ [mm]	ϵ_{max} [mm]	$\bar{d}$ [mm]
Bsl.	d1	1 (s)	1.08	2.36	0.90
	d1	2 (p)	1.51	6.77	2.02
	d1	3 (t)	0.74	2.96	0.93
	d2	1 (s)	33.44	802.57	43.25
	d2	2 (p)	7.73	74.51	8.63
	d3	1 (s)	0.76	3.32	1.12
	d3	2 (p)	2.24	28.79	3.39
	d4	1 (t)	71.721	270.628	150.371
	d4	2 (t)	26.629	106.627	48.589
Prop.	d5	3 (t)	1.14	5.05	0.90
	d6	3 (t)	0.39	5.21	0.39
	d7	1 (s)	2.11	22.80	2.94
	d7	2 (p)	1.09	12.40	1.07
Algorithm	Dataset	Tracker	σ [°]	ϵ_{max} [°]	$\bar{d}$ [°]
Baseline	d1	1 (s)	0.02	0.50	0.01
	d1	2 (p)	0.02	0.47	0.02
	d1	3 (t)	0.01	0.26	0.01
	d2	1 (s)	0.80	58.31	0.11
	d2	2 (p)	0.11	4.7	0.08
	d3	1 (s)	0.01	0.26	0.01
	d3	2 (p)	0.09	4.24	0.02
	d4	1 (t)	0.04	0.64	0.01
	d4	2 (t)	0.04	0.69	0.01
Proposed	d5	3 (t)	0.11	2.14	0.06
	d6	3 (t)	0.36	4.98	0.17
	d7	1 (s)	1.05	11.63	0.32
	d7	2 (p)	0.29	4.89	0.13

The obtained plane distances show that the baseline algorithm can be unstable when compared to our algorithm. For the position's worst case scenario, the baseline is outperformed by the proposed algorithm in precision by a factor of 15 while for accuracy by a factor of at least 36 (and up to 50). Both algorithms however have similar results for the best case. For the orientation evaluation, the baseline algorithms outperform the proposed ones due to using the gravitational acceleration. The tests we present only address two degrees of freedom. Nevertheless, the orthogonality between the lighthouse's fields of view leads to similar results for the other degrees.

The inertial measurements make the baseline algorithm behave differently from the proposed one. This can be observed in Figs. 7 and 8, with the evolution of the plane's distance for every sample with both algorithms. Although dataset d5 is longer, d3 has more samples due to incorporating IMU measurements.

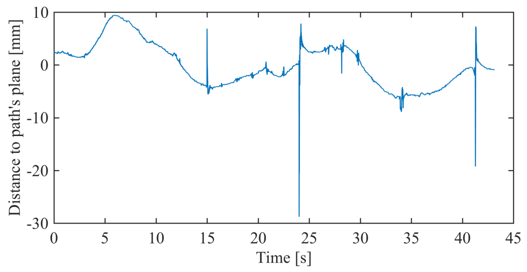

Fig. 7. Height of tracker 2 in the path generated with the baseline algorithm, for dataset d3.

Analyzing the distance to the best-fit plane, represented in Fig. 8, we can see that although it is smooth throughout most of the samples, it has some spikes. We suspect that these spikes might be related to rogue reflections perturbing the data or to a sudden loss of measurements.

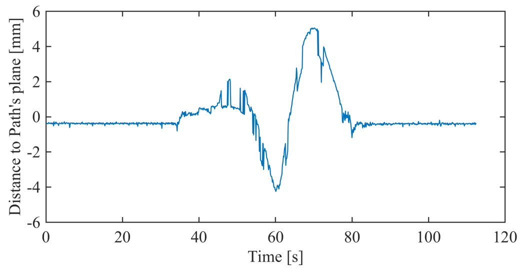

Fig. 8. Height of tracker 3 in the path generated with the proposed algorithm, for dataset d5.

Examining figure (Fig. 8) we notice some noise that resembles a step, which can be explained by the movement of the tracker changing the photodiodes that detect the infrared laser. We also suspect that the predominating offset in the same figure is related to not including the error parameters broadcasted by the lighthouses mentioned in section III. The inclusion of these parameters will be part of our future work.

VII. Outdoor Experiments

We also tried our algorithms and platform in an outdoor environment. This test was similar to the one in VI-A, but this time the lighthouses were further from each other (the distance was around 5 m). The sun's radiation interfered severely in the synchronization between lighthouses, but this was easily solved with a synchronization cable. Nevertheless, for two trackers, the worst standard deviation for the position was 13.5 mm and for the orientation was 0.0193°. These results show that both the system and the algorithms have potential for outdoor environments.

VIII. Conclusions

HTC Vive is an affordable solution for localization problems, in indoors and outdoors environments with great accuracy. We demonstrated that it has sub-millimetric precision when the tracker is static. We also provide evidence from tests performed in a controlled environment that the accuracy of Vive with the trackers in motion can range from millimetric up to metric.

We have also contributed a tracking algorithm, a calibration procedure and accompanying open-source software[4] that besides providing a completely transparent experience for the user, also grants accuracy as a means of obtaining ground truth localization data.

We intend to make our algorithms even easier to use and we also plan to experiment with other pose estimators [12] that use the IMU data and include the lighthouses' error parameters in the pose estimator.

Acknowledgement

We would like to thank the Astrobee engineering team and the NASA Human Exploration Telerobotics 2 project for supporting this work. The NASA Game Changing Development Program (Space Technology Mission Directorate), ISS SPHERES Facility (Human Exploration and Operations Mission Directorate) and Fundação da Ciência e Tecnologia (project [UID/EEA/50009/2013] and grant [SFRH/BI/135041/2017]) supported this work.

References

[1] D. Niehorster, L. Li, and M. Lappe, "The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research," *i-Perception*, vol. 8, no. 3, 2017.

[2] F. King, J. Jayender, S. Bhagavatula, P. Shyn, S. Pieper, T. Kapur, A. Lasso, and G. Fichtinger, "An Immersive Virtual Reality Environment for Diagnostic Imaging," *Journal of Medical Robotics Research*, vol. 01, no. 01, p. 1640003, 2016.

[3] P. Kulakowski, J. Vales-Alonso, E. Egea-López, and W. Ludwin, "Angle-of-arrival localization based on antenna arrays for wireless sensor networks," *Computers & Electrical Engineering*, vol. 36, no. 6, pp. 1181–1186, 2010.

[4] M. Bualat, J. Barlow, T. Fong, C. Provencher, and T. Smith, "Astrobee: Developing a Free-flying Robot for the International Space Station," in *AIAA SPACE Conference and Exposition*, 2015, p. 4643.

[5] B. Coltin, J. Fusco, Z. Moratto, O. Alexandrov, and R. Nakamura, "Localization from Visual Landmarks on a Free-flying Robot," in *IEEE/RSJ IROS*, 2016, pp. 4377–4382.

[6] 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.

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

[8] P. R. Desai, P. N. Desai, K. D. Ajmera, and K. Mehta, "A Review Paper on Oculus Rift - A Virtual Reality Headset," *IJEET*, vol. 13, no. 4, 2014.

[9] P. Rong and M. L. Sichitiu, "Angle of Arrival Localization for Wireless Sensor Networks," in *3rd Annual IEEE SECON*, vol. 1, 2006, pp. 374–382.

[10] J. Craig, *Introduction to Robotics: Mechanics and Control*. Pearson, 2005, vol. 3.

[11] D. Miller, A. Saenz-Otero, J. Wertz, A. Chen, G. Berkowski, C. Brodel, S. Carlson, D. Carpenter, S. Chen, S. Cheng *et al.*, "Spheres: A Testbed For Long Duration Satellite Formation Flying In Micro-Gravity Conditions," in *AAS/AIAA Space Flight Mechanics Meeting*, 2000, pp. 167–179.

[12] D. Schinstock, "Gps-aided INS Solution for OpenPilot," Kansas State University, Tech. Rep., 2014.

[4]Our algorithms and platform are available open-source at `https://github.com/nasa/astrobee` together with Astrobee's source code