Lighthouse Positioning System: Dataset, Accuracy, and Precision for UAV Research

Arnaud Taffanel, Barbara Rousselot, Jonas Danielsson, Kimberly McGuire, Kristoffer Richardsson,
Marcus Eliasson, Tobias Antonsson, and Wolfgang Hönig

***Abstract*— The Lighthouse system was originally developed as tracking system for virtual reality applications. Due to its affordable price, it has also found attractive use-cases in robotics in the past. However, existing works frequently rely on the centralized official tracking software, which make the solution less attractive for UAV swarms. In this work, we consider an open-source tracking software that can run onboard small Unmanned Aerial Vehicles (UAVs) in real-time and enable distributed swarming algorithms. We provide a dataset specifically for the use cases i) flight; and ii) as ground truth for other commonly-used distributed swarming localization systems such as ultra-wideband. We then use this dataset to analyze both accuracy and precision of the Lighthouse system in different use-cases. To our knowledge, we are the first to compare two different Lighthouse hardware versions with a motion capture system and the first to analyze the accuracy using tracking software that runs onboard a microcontroller.**

I. Introduction

Research and development for aerial swarms often require reproducible laboratory settings, at least in the early stages. Frequently, a motion capture system is used to mimic GPS indoors [1]. However, these positioning systems are quite expensive and introduce a centralized bottleneck, which make them difficult to use in classrooms, low-budget labs, or for research on fully distributed and resilient systems. The Valve Lighthouse system (LH) was first introduced to revolutionize virtual reality, but has also found its use in robotics [2–4]. It is about one order of magnitude cheaper than motion capture systems, easier to transport, and allows fully distributed operation as robots can compute their positions without a central computer.

When flying many robots in a small volume, the accuracy of the position tracking becomes very important. The accuracy of the LH has been determined before using the official bulky tracking device and centralized software [5–10]. We are interested in quantifying the performance using small custom trackers (similar to [11]) with open-source software (similar to [12]). In contrast to existing work, our open-source software is designed to run in real-time onboard small UAVs — we rely on the Crazyflie 2.1.[1] To this end, we provide a dataset that includes Lighthouse data (raw and onboard processed), motion capture ground truth in different scenarios (stationary, manual movement, flight) using the two available hardware versions of LH. We also contribute our analysis of the data to quantify the precision (also known as jitter) and accuracy of the system.

II. Lighthouse positioning system

In this section we first introduce the hardware of the Lighthouse (SteamVR) base stations[2] and the Lighthouse sensor board[3]. Afterwards, we specify the Lighthouse positioning system algorithms for estimating the Crazyflie's position by using the crossing beam method (C.B.) or by using an extended Kalman filter (EKF).

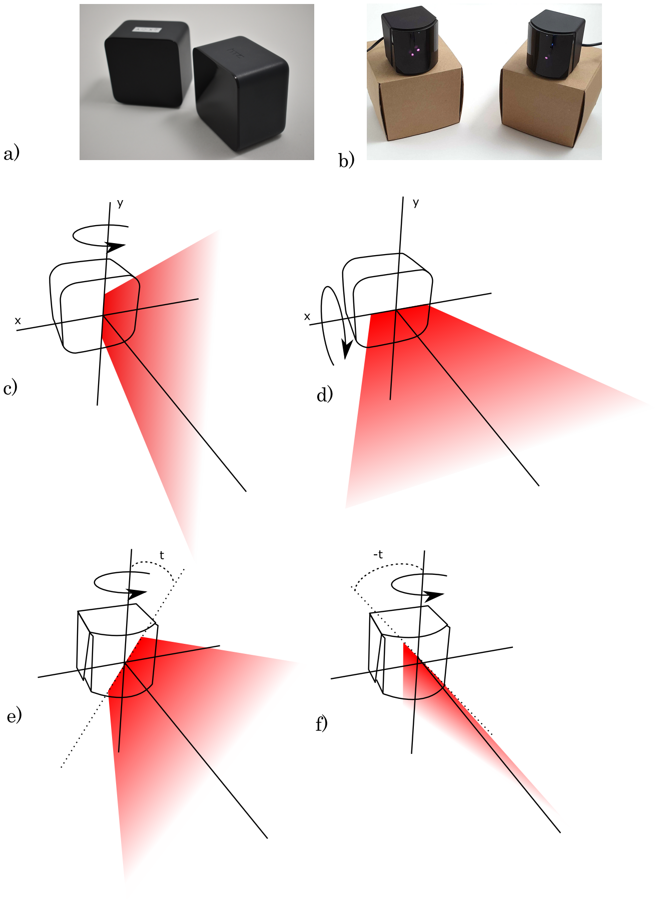

Fig. 1. Pictures and operational principle of the two Lighthouse positioning system versions. LH1 is shown in a) and has two rotating drums creating IR light planes that are perpendicular to each other c) and d). LH2 is shown in b) and has just one rotating drum with the two sweep planes angled at a tilt angle t shown in e) and f).

All authors are with Bitcraze AB, Sweden.
Email: `firstname@bitcraze.io` or `all@bitcraze.io`

Dataset and code are available at: https://github.com/bitcraze/positioning_dataset.

[1] https://store.bitcraze.io/products/crazyflie-2-1

[2] https://www.vive.com/eu/accessory/base-station2/, ... /basestation/

[3] https://www.bitcraze.io/products/lighthouse-positioning-deck/

A. Hardware

The Lighthouse base stations contain drums which rotate infrared (IR) light planes that are captured by the IR receivers on the Lighthouse sensor board mounted on the robot (so-called deck). Currently, there are two versions:

- V1 (Fig. 1a) has two rotating drums with each carrying one sweep (Fig. 1c and d). The rotating drums of two base stations are synced with each other by a sync cable or visual line of sight. This paper will refer to this version as LH1.
- V2 (Fig. 1b) has one rotating drum with two planes each at a different angle (Fig. 1e and f). Depending on the channel configuration of the base station, the drums rotate at a slightly different rate. Thus, light planes of different base stations may periodically collide. This paper will refer to this as LH2.

Table I shows the the tilt angles (t_p) of the IR light planes and the drum's rotation matrices (R_d) for both LH1 and LH2.

TABLE I
CHARACTERISTICS OF LH1 AND LH2 WITH I BEING AN IDENTITY MATRIX.

Version:	LH1		LH2	
Light Plane:	1st	2nd	1st	2nd
t_p	0	0	$-\pi/6$	$\pi/6$
R_d	I	$\begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & -1 & 0\end{bmatrix}$	I	I

The Lighthouse deck, see Fig. 2, contains 4 sensors that are able to detect passing IR light planes from both LH1 and LH2. The received signals are decoded by an FPGA, which transmits the timestamps, pulse-width, and, for LH2, the decoded base station synchronization time to the Crazyflie. On the STM32F4 chip of the Crazyflie, the data is processed to plane sweep angles, which are used for the position estimation either by the C.B. method (Section II-B) or the EKF (Section II-C). There are also calibration models for LH1 and LH2 that are implemented in the firmware of the Crazyflie[4], which will not be covered in detail in this paper.

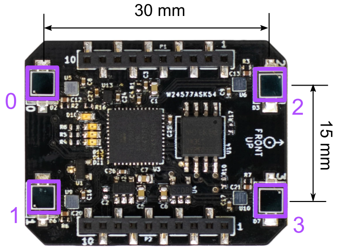

Fig. 2. Lighthouse deck with 4 TS4231 IR receivers.

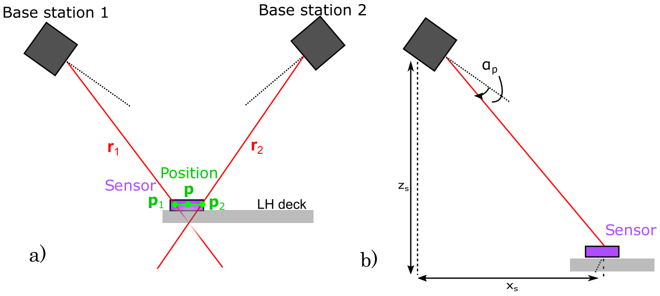

Fig. 3. The two position estimation methods, where a) depicts the crossing beam method (C.B.) and b) depicts state variables necessary for the EKF's sweeping beam measurement model.

B. Crossing Beam Method

The crossing beam method is developed by the open-source community[5] for LH1 and has been extended by us to work for LH2 on the Crazyflie. The method needs two base stations to be in direct line of sight with the Lighthouse deck on the Crazyflie (see Fig. 3a). The intersection of the two light planes of the base station (b) to one of the Lighthouse deck's sensor (s) results in a ray ($\boldsymbol{r}'_{b,s}$) that points towards the Lighthouse deck. The ray's origin and direction are then converted to the global coordinate system ($\boldsymbol{r}_{b,s}$) by using the rotation matrix (R_b) and translation ($\boldsymbol{t}_b$) of the base station, i.e., with slight abuse of notation:

$$\boldsymbol{r}_{b,s} = R_b \cdot \boldsymbol{r}'_{b,s} + \boldsymbol{t}_b. \tag{1}$$

Thus, the rays of the two base stations that are perceived by sensor s in global coordinates are $\boldsymbol{r}_{1,s}$ and $\boldsymbol{r}_{2,s}$. The position $\boldsymbol{p}_s = (x_s, y_s, z_s)^\top$ of sensor s is then calculated by first solving[6] the following optimization problem

$$\boldsymbol{p}_{1,s}, \boldsymbol{p}_{2,s} = \underset{\boldsymbol{p}_1 \in \boldsymbol{r}_{1,s},\ \boldsymbol{p}_2 \in \boldsymbol{r}_{1,s}}{\arg\min} \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|_2 \tag{2}$$

to compute the two points $\boldsymbol{p}_{1,s}$ and $\boldsymbol{p}_{2,s}$ on the two rays that are the closest to each other. Then, we compute $\boldsymbol{p}_s$ and estimate the error δ using

$$\boldsymbol{p}_s = \frac{\boldsymbol{p}_{1,s} + \boldsymbol{p}_{2,s}}{2} \text{ and } \delta = \|\boldsymbol{p}_{1,s} - \boldsymbol{p}_{2,s}\|^2, \tag{3}$$

respectively. Here, the position of the sensor ($\boldsymbol{p}_s$) is estimated by using the middle between the two closest points. Afterwards, the average of the estimated position of all the sensors ($s = 0 \ldots 3$) is used as a measured position. For our analysis, we rely on the estimated error (δ) to filter the measured data.

C. EKF Lighthouse Measurement model

When using the EKF implementation on the Crazyflie [13], each IR light plane p can be used as a measurement. This enables the Crazyflie to fly with only one base station. This section explains the measurement model that is used for our implementation.

The observation model (see Fig. 3b) maps the relevant part of the EKF state, the estimated x_s, y_s, and z_s components of the sensor position in the global coordinate frame, to the

[4] https://github.com/bitcraze/crazyflie-firmware

[5] https://github.com/ashtuchkin/vive-diy-position-sensor/

[6] http://geomalgorithms.com/a07-_distance.html#Distance-between-Lines

measured observations α_p:

$$\alpha_p = \arctan \frac{y_s}{x_s} + \arcsin \frac{z_s \tan t_p}{r_s}, \tag{4}$$

$$\text{where } r_s = \sqrt{x_s^2 + y_s^2}. \tag{5}$$

The measurement model is linearized by the Jacobian:

$$\boldsymbol{g}_p = \left(\frac{\partial \alpha_p}{\partial x_s}, \frac{\partial \alpha_p}{\partial y_s}, \frac{\partial \alpha_p}{\partial z_s} \right) \tag{6}$$

$$\boldsymbol{g}_p = \left(\frac{-y_s - x_s z_s q_p}{r_s^2}, \frac{x_s - y_s z_s q_p}{r_s^2}, q_p \right) \tag{7}$$

$$\text{with: } q_p = \frac{\tan t_p}{\sqrt{r_s^2 - (z_s \tan t_p)^2}} \tag{8}$$

This Jacobian is first rotated to the coordinate system of the base stations with the drum's rotation matrix (R_d, see Table I) and the rotation matrix of the orientation of the base station itself (R_b):

$$\boldsymbol{g}'_p = (R_d \cdot R_b^{-1})^{-1} \cdot \boldsymbol{g}_p = R_b \cdot R_d^{-1} \cdot \boldsymbol{g}_p. \tag{9}$$

With the measurement model, the characteristics of Table I, and the measured sweep angles of each sensor, the EKF is able to estimate the position of the Crazyflie.

III. Data Collection Setup

In this section we introduce our flight area configuration with the LH base stations and the ground truth measurement, as well as the hardware configuration of the Crazyflie used for the data collection.

A. Flight Area and Hardware Setup

All experiments are conducted in a $7\,\mathrm{m} \times 7\,\mathrm{m} \times 3\,\mathrm{m}$ flight space that is equipped with a motion capture system (6 Qualisys Miqus M3)[7] and has no natural light that might interfere with the LH or motion capture (Fig. 4). Note that the Crazyflie flight area ($1.5\,\mathrm{m} \times 1.5\,\mathrm{m} \times 1.5\,\mathrm{m}$) was placed such that it flew in the middle of the LH's systems field of view and that it was measurable by the motion capture system for ground truth.

Both LH and motion capture operate using infrared light, which makes using both of them simultaneously difficult and has motivated previous studies to use an industrial serial manipulator to generate ground truth data rather than a motion capture system [6]. When using a motion capture system in its default mode, where an IR ring around the cameras is active, the LH sensors cannot perceive the beams from the LH base stations, as they use the same spectrum. We found that this interference can be avoided entirely by using active markers: instead of using the classical retro-reflective markers, we rely on IR LEDs as markers and disable the IR ring around the motion capture cameras. As long as the radiation cone of the IR LEDs does not include the LH sensors, the LH perception is unimpaired.

In practice, we use the following commercially available off-the-shelf hardware[8]. A Crazyflie 2.1 with the extension boards (so-called "decks") mounted on top: SD-card deck, Lighthouse deck, and an active marker deck (Fig. 5). The active marker deck has to be above the LH deck to keep the IR LED radiation cone outside the LH sensors. The micro SD card deck is used to log the raw data that we receive onboard the Crazyflie in a space-efficient, binary encoding, where each datapoint is annotated with a high-precision microsecond timestamp. The Crazyflie communicates with a PC over a custom radio (Crazyradio PA) only for high-

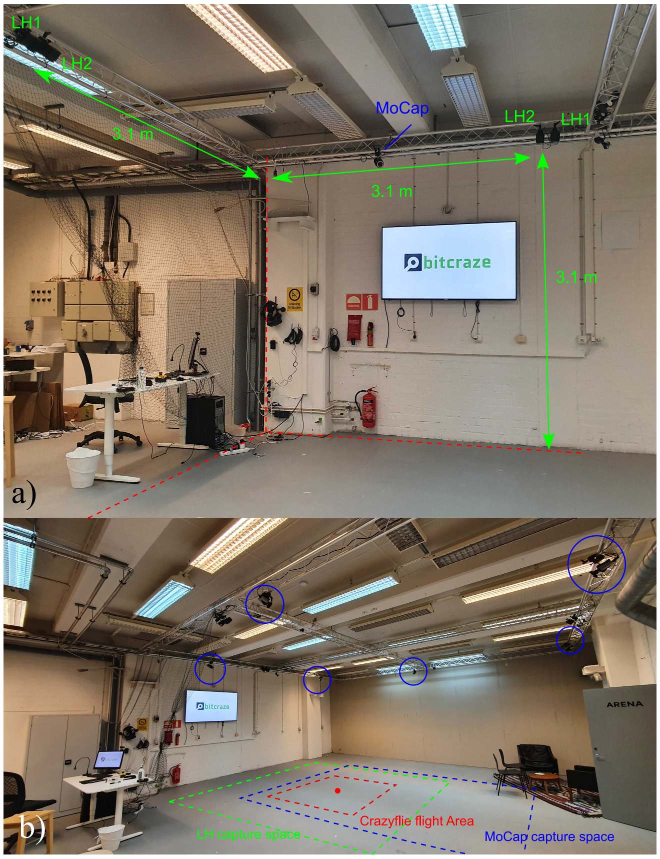

Fig. 4. Photos and indications of a) the closeup of the Lighthouse setup in the flight area and b) of the full flight area with Qualisys motion capture (MoCap) cameras.

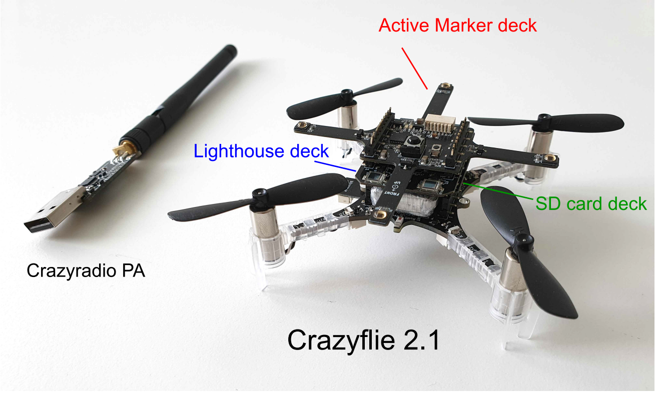

Fig. 5. Picture of the Crazyflie used for data collection with (top to bottom) an active marker deck, a Lighthouse deck and a uSD-card deck, which were kept together with long male pin headers.

[7] https://www.qualisys.com/cameras/miqus/

[8] https://www.bitcraze.io

TABLE II
DATASET SCENARIOS. C.B. STANDS FOR THE CROSSING BEAM METHOD.

LH	Estimation	Setting	# Datasets	$\sum$ Duration [s]
LH1	C.B.	Stationary	5	59
LH1	C.B.	Ext. Motion	5	603
LH1	C.B.	Flight	5	416
LH1	EKF	Stationary	5	59
LH1	EKF	Flight	4	436
LH2	C.B.	Stationary	5	59
LH2	C.B.	Ext. Motion	5	603
LH2	C.B.	Flight	4	440
LH2	EKF	Stationary	5	59
LH2	EKF	Flight	4	437

level commands, such as start/stop of logging, as the radio bandwidth is insufficient to transfer the data in real-time.

B. Data Collection Preparation

Before conducting experiments, we calibrate the motion capture system using the vendors calibration kit and software. We estimate the pose of the two base stations using the CfClient software, which computes the poses after placing the Crazyflie at the origin using `solvePnP` in OpenCV.[9] Since this method only relies on four points in one plane, it is expected to be less accurate than a typical motion capture calibration that relies on hundreds of points. We do not attempt to align the coordinate systems, as this would not be possible with high precision. Instead, we propose to compute the transformation in postprocessing, as described in Section IV.

We consider all valid combinations of LH hardware version (LH1 and LH2), state estimation method (crossing beam and EKF), and scenarios (stationary, external motion, flight), see Table II. For a description of the hardware and state estimation, see Section II. In the stationary scenario we place the Crazyflie at different positions and collect data for 10 consecutive seconds. This is in particular useful to compute an estimate of the noise of the estimation, e.g., jitter. In the external motion or movement scenario, we mount the Crazyflie on a long rod and move it around manually inside the capture space, similar to how a calibration wand is moved. Here, higher velocities might be reached and there is no known dynamics model that can be applied. We use various different sweeping patterns as well as movement velocities. For the flight scenario, we fly the Crazyflie using the LH system as feedback using a sweeping pattern and randomly sampled position setpoints with two desired velocities ($0.25\,\mathrm{m/s}$ and $0.5\,\mathrm{m/s}$). Since the EKF requires a known dynamics model, the combination of EKF and external motion is not considered.

We operate the motion capture system at a fixed sample frequency of $300\,\mathrm{Hz}$. Since we only consider positions and not orientation for this work, we do not use the rigid body tracking integrated in the Qualisys software, but instead compute the mean of the position, $\boldsymbol{p}^{\mathrm{MC}}$, of the four 3D markers created by the active marker deck (the geometry of the deck is such that the mean reflects the geometric center). If there were not exactly 4 markers detected, we report NaN. Each datapoint is timestamped with the Qualisys camera timestamp, t^{MC}, and the resulting time series

$$\mathcal{D}^{\mathrm{MC}} = \{(t_i^{\mathrm{MC}}, \boldsymbol{p}_i^{\mathrm{MC}}) \mid i = 1, \ldots\} \tag{10}$$

is written to a `mocap<id>.npy` file.

On the Crazyflie, we record the gyroscope and acceleration at $100\,\mathrm{Hz}$, the raw LH angles and the estimated position (by either the EKF or crossing beam method) as event streams. Each datapoint is timestamped with the onboard microsecond timer and the resulting time series is written to a `log<id>` file. For brevity, we only introduce notation for the estimated position events:

$$\mathcal{D}^{\mathrm{CF}} = \{(t_i^{\mathrm{CF}}, \boldsymbol{p}_i^{\mathrm{CF}}) \mid i = 1, \ldots\}. \tag{11}$$

All files are compact binary representations and we provide Python scripts to parse and analyze the data.

In order to enable synchronization between the motion camera clocks and the Crazyflie clock, we record the onboard microsecond timestamp when the IR LEDs on the active marker deck are turned on and off, respectively. These two events are sufficient to synchronize the clocks (offset and drift).

IV. Data Analysis

We now discuss how the collected data can be postprocessed for analysis and the performance metrics that we are considering in this paper.

A. Spatiotemporal Alignment

The dataset contains data from two different clock sources (motion capture clock and Crazyflie clock) and two different coordinate systems (motion capture and LH). For any quantitative analysis we need to align the data in both space and time.

For the temporal alignment, let t_s^{CF} and t_f^{CF} be the time of the Crazyflie clock when the IR LEDs on the active marker deck were enabled and disabled, respectively. Let t_s^{MC} and t_f^{MC} be the time of the motion capture clock when 3D markers were first and last detected, respectively. We can then rescale the time for each recorded Crazyflie event i to

$$\hat{t}_i^{\mathrm{CF}} = (t_i^{\mathrm{CF}} - t_s^{\mathrm{CF}}) \frac{t_f^{\mathrm{MC}} - t_s^{\mathrm{MC}}}{t_f^{\mathrm{CF}} - t_s^{\mathrm{CF}}}. \tag{12}$$

The compensation for the clock drift is important, as the Crazyflie does not have a high-precision crystal.

Next, we compute the matching ground truth position, in the motion capture reference frame, for each Crazyflie event i using linear interpolation:

$$\hat{\boldsymbol{p}}_i^{\mathrm{MC}} = \mathrm{interp}(\hat{t}_i^{\mathrm{CF}}, \{t_i^{\mathrm{MC}} - t_s^{\mathrm{MC}} \mid \forall i\}, \{\boldsymbol{p}_i^{\mathrm{MC}} \mid \forall i\}). \tag{13}$$

Since the sampling rate of the motion capture is roughly one order of magnitude higher than the LH events, this is a good approximation. Note that $\hat{\boldsymbol{p}}_i^{\mathrm{MC}}$ may be NaN, if there was no motion capture data available around time $\hat{t}_i^{\mathrm{CF}}$.

[9] https://docs.opencv.org/4.5.1/d9/d0c/group__calib3d.html

We now have a temporally aligned dataset $\tilde{\mathcal{D}} = \{(\hat{t}_i^{\text{CF}}, \boldsymbol{p}_i^{\text{CF}}, \hat{\boldsymbol{p}}_i^{\text{MC}}) \mid i = 1, \ldots\}$, where the two positions come from a different reference frame and the transformation between the reference frames is unknown. It is easy to estimate the transformation numerically given this aligned dataset. We rely on an approach that uses singular value decomposition (SVD) [14] and that has been shown empirically to produce high-quality and robust results [15]. The approach requires as only input $\tilde{\mathcal{D}}$ (ignoring the temporal component) and outputs a rotation matrix R and translation vector $\boldsymbol{t}$, which transforms points from the LH frame to the motion capture frame. Finally, we can compute our spatiotemporally aligned dataset

$$\begin{aligned} \hat{\boldsymbol{p}}_i^{\text{CF}} &= R\boldsymbol{p}_i^{\text{CF}} + \boldsymbol{t} \quad \forall i \\ \hat{\mathcal{D}} &= \{(\hat{t}_i^{\text{CF}}, \hat{\boldsymbol{p}}_i^{\text{CF}}, \hat{\boldsymbol{p}}_i^{\text{MC}}) \mid i = 1, \ldots\}. \end{aligned} \tag{14}$$

In practice, we found that the motion capture system has some additional latency of unknown source. To mitigate this, we consider different offsets t_s^o and t_f^o in the range of $\pm 100\,\text{ms}$ and repeat the spatiotemporal alignment procedure described earlier. We then pick the offsets that minimize the Euclidean error $\sum_i \|\hat{\boldsymbol{p}}_i^{\text{CF}} - \hat{\boldsymbol{p}}_i^{\text{MC}}\|_2$.

B. Data Filtering

The dataset contains periods of time where either no LH data was received or the Crazyflie could not be observed with the motion capture system. The first one largely occurs during the external motion, since the LH capture space is smaller than the motion capture space. Other reasons for missing data include interference between the two base stations of LH2 that occurs periodically, as well as a partial occlusion of the LH sensors due to the active marker deck mounted on top. Missing data from the motion capture system occurs if not all 4 markers are visible. Thus, we exclude datapoints where we had no motion capture ground truth.

For the crossing beam estimation, we exclude datapoints where we did not receive both light planes from both base stations on all four sensors (by design of the crossing beam method). We also exclude datapoints where the crossing beam δ is greater than $0.1\,\text{m}$. This variable is computed onboard as the error of the ray intersection approximation.

For the EKF estimation, we exclude datapoints where we did not receive at least one light plane for at least one LH sensor from both base stations. We noticed that the filter tends to drift during these time periods, which skews the results otherwise and makes them difficult to compare to the crossing beam method that relies on much more aggressive data filtering.

C. Performance Metrics

We are interested in quantifying precision and accuracy of the LH system. For precision, we compute the root mean square of the position (also often termed as jitter):

$$P = \sqrt{\frac{1}{|\hat{\mathcal{D}}|} \sum_{i=1}^{|\hat{\mathcal{D}}|-1} \|\hat{\boldsymbol{p}}_i^{\text{CF}} - \hat{\boldsymbol{p}}_{i+1}^{\text{CF}}\|_2^2}. \tag{15}$$

TABLE III
PRECISION/JITTER AND SAMPLE FREQUENCY.

	LightHouse 1		LightHouse 2		MoCap
	C.B.	EKF	C.B.	EKF	
Freq. [Hz]	30 ± 2.4	N.A.	34 ± 18	N.A.	300 ± 0.1
Jitter [mm]	0.6	3.9	0.3	0.7	0.1

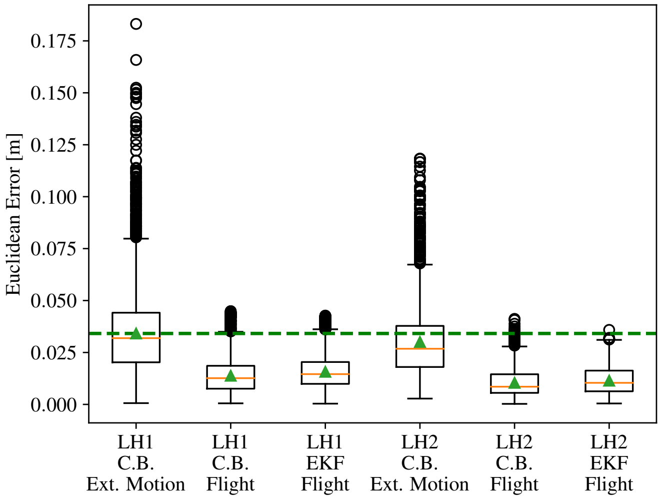

Fig. 6. Accuracy of Lighthouse 1 and 2 in different scenarios. LH2 achieves better accuracy compared to LH1 in all cases. The flight scenarios have a better accuracy due to the lower velocity and focus on the volume with excellent LH coverage. EKF and crossing beam state estimators achieve a similar accuracy during our flight tests.

For the accuracy, we consider the Euclidean error for each datapoint

$$A_i = \|\hat{\boldsymbol{p}}_i^{\text{CF}} - \hat{\boldsymbol{p}}_i^{\text{MC}}\|_2, \tag{16}$$

where interesting statistics include the mean $\bar{A} = \frac{1}{|\hat{\mathcal{D}}|} \sum_i A_i$ and maximum error $A_{\text{max}} = \max_i A_i$.

V. Results

We first analyze the precision and sample frequency by considering the data in the stationary setting, see Table III.

For the crossing beam state estimation, the sample frequency is around $30\,\text{Hz}$, which is sufficient for non-aggressive flight maneuvers. However, in the LH2 case, this frequency has a very large standard deviation which is caused by the periodic interference of the two base stations. An initial investigation showed that this is mostly caused by overly conservative filtering of the raw sensor data and can be improved with a software upgrade in the future. Note that the provided dataset contains the raw sensor data as well and could be used to test less conservative filtering techniques. For the EKF estimation, the sample frequency is arbitrary and data from LH is considered in the filter as soon as it is observed.

The precision is significantly lower for LH2 compared to LH1. Moreover, the precision is worse when using the EKF compared to the crossing beam state estimation for both LH1 and LH2. This indicates that the jitter induced by the IMU measurements or by the process noise is higher than the jitter induced by the LH. Tuning the EKF, for example with the use of this dataset, might improve the result.

The accuracy in various settings is shown using box and whisker plots in Fig. 6. The mean and median Euclidean error is in all cases in the low centimeter ($2\,\mathrm{cm}$ to $4\,\mathrm{cm}$) range. There are some outliers, but during flight they are in the worst case about $5\,\mathrm{cm}$. Larger outliers can be observed for the external motion case (up to $18\,\mathrm{cm}$) with the crossing beam state estimator. These are caused by faster motions as well as operation close to the boundary of the volume that is observable by the LH base stations. Overall, LH2 has a higher accuracy compared to LH1 in all scenarios, which we attribute to the simplified mechanics or improved factory calibration. The two state estimators, EKF and crossing beam, have a similar accuracy during our flight tests.

VI. Conclusion

We investigate the Lighthouse positioning system for UAV research in comparison to a traditional motion capture system by providing a dataset and analysis of accuracy and precision. We quantify the precision in terms of RMS jitter, which is in most of our settings in the sub-millimeter range (about 5 times larger than the reference motion capture system). We quantify the accuracy in terms of Euclidean error, which is about $1\,\mathrm{cm}$ in the average case with outliers up to $5\,\mathrm{cm}$ during flight, and significantly higher outliers when moving freely. Our results indicate that the LH system is a versatile alternative to traditional motion capture systems: as feedback during flight (EKF mode) or as a ground truth (crossing beam mode) for other, less accurate positioning systems such as ultra-wideband (UWB) based systems. The Lighthouse system's price and fully distributed operation make it in particular interesting for classroom use and distributed swarm research.

The dataset also opens up many exciting avenues for future research. First, it would be interesting to improve the automatic metric of the crossing beam method to filter out bad state estimates. We found that the current approach of using the δ variable does not always correlate with high errors. Second, the precision measurements showed that LH2 has a high standard deviation for the sample frequency, which is likely caused by pessimistic filtering of the raw data. Third, the high precision might indicate that a higher accuracy is possible as well, if better calibration methods are developed. We believe that our dataset can be used for all of these directions as well as for tuning the EKF, as it contains the relevant raw data.

References

[1] J. A. Preiss, W. Hönig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.

[2] M. Hoppe, M. Burger, A. Schmidt, and T. Kosch, "DronOS: A flexible open-source prototyping framework for interactive drone routines", in *International Conference on Mobile and Ubiquitous Multimedia (MUM)*, ACM, 2019, 15:1–15:7.

[3] M. Greiff, A. Robertsson, and K. Berntorp, "Performance bounds in positioning with the vive lighthouse system", in *IEEE International Conference on Information Fusion (FUSION)*, 2019, pp. 1–8.

[4] K. Sletten, "Automated testing of industrial robots using HTC vive for motion tracking", M.S. thesis, University of Stavanger, Norway, 2017.

[5] D. C. 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.

[6] M. S. Ikbal, V. Ramadoss, and M. Zoppi, "Dynamic pose tracking performance evaluation of HTC vive virtual reality system", *IEEE Access*, vol. 9, pp. 3798–3815, 2021.

[7] R. Wu, J. Pandurangaiah, G. M. Blankenship, *et al.*, "Evaluation of virtual reality tracking performance for indoor navigation", in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2020, pp. 1311–1316.

[8] J. Lwowski, A. Majumdat, P. Benavidez, J. J. Prevost, and M. Jamshidi, "HTC vive tracker: Accuracy for indoor localization", *IEEE Systems, Man, and Cybernetics Magazine*, vol. 6, no. 4, pp. 15–22, 2020.

[9] E. Luckett, "A quantitative evaluation of the HTC vive for virtual reality research", Bachelor's Thesis, The University of Mississippi, 2018.

[10] S. M. van der Veen, M. Bordeleau, P. E. Pidcoe, C. R. France, and J. S. Thomas, "Agreement analysis between vive and vicon systems to monitor lumbar postural changes", *Sensors*, vol. 19, no. 17, p. 3632, 2019.

[11] C. Honnet and G. Lopes, "Hivetracker: 3d positioning for ubiquitous embedded systems", in *ACM International Joint Conference on Pervasive and Ubiquitous Computing and International Symposium on Wearable Computers (UbiComp/ISWC)*, R. Harle, K. Farrahi, and N. D. Lane, Eds., 2019, pp. 288–291.

[12] 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 (IROS)*, 2018, pp. 2610–2615.

[13] M. W. Mueller, M. Hehn, and R. D'Andrea, "Covariance correction step for kalman filtering with an attitude", *Journal of Guidance, Control, and Dynamics*, pp. 1–7, 2016.

[14] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, pp. 698–700, 1987.

[15] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: A comparison of four major algorithms", *Mach. Vis. Appl.*, vol. 9, no. 5/6, pp. 272–290, 1997.