Indoor vehicle-in-the-loop simulation of unmanned micro aerial vehicle with artificial companion.

Antal Hiba[1,*], Viktor Kortvelyesi[1], Albert Kiskaroly[2], Omkar Bhoite[1], Patrik David[1], and Andras Majdik[2]

[1]*Computational Optical Sensing and Processing Laboratory*
Institute for Computer Science and Control (SZTAKI), ELKH, Budapest Hungary
[2]*Machine Perception Research Laboratory*
Institute for Computer Science and Control (SZTAKI), ELKH, Budapest Hungary
[*]email: hiba.antal@sztaki.hu

***Abstract*—Vehicle-in-the-loop simulation is an extension of the well-known hardware-in-the-loop technique, where a vehicle moves in real space while a simulator generates input for on-board sensors real-time. VIL frameworks designed for unmanned aerial vehicles have many open challenges. This paper introduces an indoor VIL for micro aerial vehicles in a drone arena equipped with precise positioning system and reliable wireless communication. This indoor setup stands as a prototype and predecessor for other outdoor VIL simulators for large scale applications. The raw optical navigation performance is tested which is the building block for sensor fusion and on-board fast sensor consistency check. A use-case of a companion drone in cooperative navigation is also presented.**

***Index Terms*—UAV, optical navigation, vehicle-in-the-loop, hardware-in-the-loop**

I. Introduction

Vehicle-in-the-loop (VIL) simulation is an advanced development technique where an autonomous vehicle platform moves in a real dedicated test environment, while a virtual use case environment and actors are simulated around it through artificial sensor inputs. Hardware-in-the-loop simulation (where the movements are also simulated) is a widespread validation method for both autonomous ground and aerial vehicles. VIL has many applications for autonomous cars [1], [2], however, it is still challenging to apply this technique to UAVs. Drones have limited payload and computation capability, thus the integration of a simulator on-board is heavily constrained. Another possible way to perform VIL with a UAV is to create a simulator node on the ground and transmit the artificial sensor data to the drone, which creates wireless data transmission challenges (two-way delay, insufficient bandwidth at longer distances). VIL makes dangerous validation flights possible at a safe real site while the simulator mimics a crowded street or an industrial facility for the navigation and mission sensors (camera, LIDAR). Furthermore, it gives an opportunity to test the cooperation of multiple drones while only one real drone proves the airworthiness.

State-of-the-art simulation environments for UAVs with the possibility of VIL are indoor test facilities. The most advanced examples in Europe are the ETHZ Flying Machine Arena [3] and ENAC Drone Flight Arena Toulouse Occitanie [4]. VIL can be performed through a precise indoor positioning system which provides ground truth trajectories for validation of on-board control, and makes precise movement of the digital twin possible in the virtual environment. Current research attempts [5] aim for stable indoor solutions for Micro Aerial vehicles (MAV), where stable wireless communication is guaranteed, however, these small UAV platforms have significantly less payload capabilities for sensors and on-board computing. Most of the existing frameworks focus on the control validation and use complex 3D environments only for visualization, however, Unity and Unreal engines have the potential to generate realistic sensor data from the artificial environment. Advanced HIL solutions are presented for various UAV types [6], and swarms [7], where communication links between UAVs were also real. In [8] the swarm HIL is connected to AirSim [9], however, the usage of artificial sensors of AirSim was not presented.

Indoor VIL solutions are a compulsory step towards outdoor solutions. The main gap between the two is the positioning system which is available for indoor VILs. Without an independent robust source of positioning in real space, the UAV cannot use its virtual sensors (camera, LIDAR) for navigation or mission tasks in the virtual space, because the generation of artificial sensor signals at a given position in the simulator is required. For outdoor VIL, the real on-board sensors which are mainly independent from the local features of the environment can be used for sample point generation (GNSS, IMU, barometer, compass), furthermore, it is possible to add extra positioning anchors for a dedicated outdoor test environment (outdoor arena). This article focuses on the indoor setup where

Project no. TKP2021-NVA-01 has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.
The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002)

the test application is optical navigation.
Structure-based navigation methods provide high accuracy, however, do not meet real-time requirements [10]. End to end neural networks can regress the UAV pose from images, however, it is often overfitted to a specific area [11]. In state-of-the-art drone racing of UAVs, where fast navigation is crucial, even control is directly derived from images by a neural network [12]. The usage of multiple drones (companion) provides more room to add redundancy. The most promising approach is collaborative stereo [13] where stereo-vision, IMU, sonar and linear velocity are combined by an extended Kalman-filter.
This paper introduces an indoor VIL setup for UAVs which is based on the Unreal Engine4 CARLA simulator and OptiTrack positioning system for Crazyflie MAVs. The optical navigation performance on a virtual camera is presented with detailed description which supports the integration of OpenCV applications with the Unreal Engine. Beyond the mono camera navigation solution, a use-case is also presented, where a virtual companion drone provides additional navigation information to the main mission drone.

II. Drone arena hardware and software setup

The hardware of the drone arena is the extended version of the system which is described in [5]. The OptiTrack system which provides accurate position and orientation of the MAV consists of 10 Prime13 infrared motion capture cameras which are capable of tracking special markers in 3D. Each camera emits IR light, and the markers have high IR reflection which ensures robust detection of these point-like features in each camera image. Each marker can be seen by at least 4 cameras which makes it possible to pair and triangulate their 3D position. Crazyflie 2.1 drones [14] are used as MAV platforms with 4 optical markers. For the rigid body calculations of the UAV 3 markers are sufficient, however, the fourth marker gives more robustness. The system has a Windows software component called Motive which can calculate rigid body position with sub-millimeter precision at 240Hz and transmits the OptiTrack measurements through UDP.

The drone is controlled by an ubuntu desktop computer through the Robot Operating System (ROS). The main components of the drone arena are presented in Figure (1). The crazyswarm open source project [15] provides the basis of our manual and unmanned control. The drone can be flown inside a 7m x 8m x 3.5m space which is covered by the OptiTrack setup, and the communication between the ground station and the drone is performed through radio.

III. Integration of the CARLA simulator

CARLA [16] is an open urban simulator based on Unreal Engine4 which focuses on the simulation of ground vehicles, and artificial sensors. The integration of a UAV can be done easily as an unreal engine actor with an attached static mesh

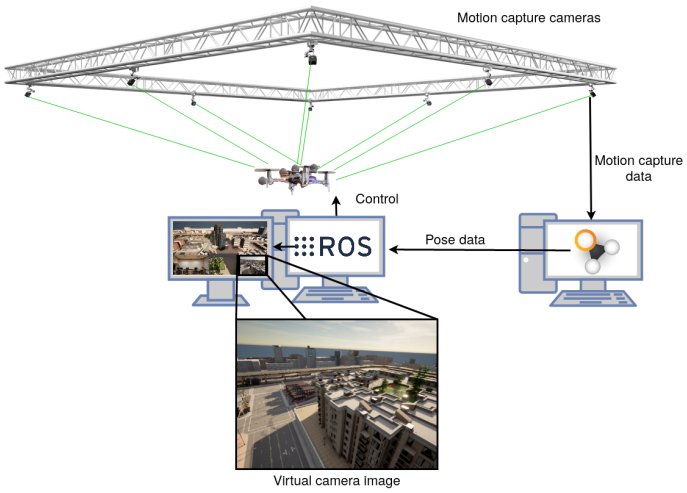

Fig. 1. Components of the SZTAKI MIcro aerial vehicle and MOtion capture (MIMO) Arena.

of the drone, which let us attach CARLA sensors to the drone actor in the Python API. Dynamics of the drone has to be defined by the user, because CARLA has no native support of UAVs. It can be done by using recorded trajectories from real flights, or simulating drone dynamics in Matlab online [17], or using real-time drone state measurements in a VIL setup.
CARLA 0.9.10 is used with Unreal Engine 4.24 which is connected to ROS by the carla_ros_bridge package. Actual position of the drone can be accessed from the ROS topic of the vrpn_client_node. The position and orientation of the UAV is updated in CARLA at 30Hz which results in a smooth accurate movement in the simulator. The images of the virtual on-board camera can be accessed as a ROS topic published by the carla_ros_bridge. This ROS environment makes data visulaization (RViz) and recording (ROS bag) simple and professional.

A. Coordinate systems in optical navigation

Movement of an object in 3D has many possible representations, and it can be challenging in a VIL system which consists of a real UAV which is controlled in North-East-Down (NED) right handed coordinate system, while it is simulated in a North-East-UP left handed system of Unreal CARLA simulator, and its camera sensor is using OpenCV functions which operates in X-Y-Depth camera coordinate system, and 3D scatter plots using a 4th coordinate system during visualization. CARLA uses meters, however, any direct measurements in Unreal Engine (position of landmarks) uses unreal units which is 1cm. Furthermore, CARLA defines rotations in degrees, while all other mathematical tools uses radians. This section summarizes the necessary transformations in a CARLA-OpenCV VIL for UAV optical navigation.
In optical navigation for UAVs, there are three main coordinate systems: global NED which is a flat Earth approximation Cartesian system with an origin on the surface; body NED

which origin is the center of the UAV; and camera NED which has the focal point as the center of the coordinate system. In a VIL setup, the camera NED position and orientation are measured (can be derived from UAV state) and these are transformed to the CARLA system for image generation from the simulated world Eq. (1).

$$\begin{pmatrix} \text{x} \\ \text{y} \\ \text{z} \end{pmatrix} \xrightarrow{\text{to CARLA}} \begin{pmatrix} \text{x} \\ \text{y} \\ \text{-z} \end{pmatrix} \quad \begin{pmatrix} yaw \\ pitch \\ roll \end{pmatrix} \xrightarrow{\text{to CARLA}} \begin{pmatrix} yaw \\ -pitch \\ -roll \end{pmatrix} \tag{1}$$

OpenCV uses a camera system where x and y are image coordinates (can be connected to real distances by $\frac{pixel}{mm}$), and z is the depth thus the image plane is at $z = f$, where f is the focal length in pixel. The relationship between the CARLA and OpenCV coodinates is given in Eq. (2).

$$\begin{pmatrix} \text{x} \\ \text{y} \\ \text{z} \end{pmatrix} \xrightarrow{\text{to OpenCV}} \begin{pmatrix} \text{y} \\ \text{-z} \\ \text{x} \end{pmatrix} \quad \begin{pmatrix} yaw \\ pitch \\ roll \end{pmatrix} \xrightarrow{\text{to OpenCV}} \begin{pmatrix} yaw \\ -pitch \\ -roll \end{pmatrix} \tag{2}$$

For a camera, the intrinsic matrix K is defined as Eq. (3), where (p_x, p_y) is the principal point.

$$K = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The camera matrix K describes the pinhole projection from camera coordinate system to image coordinates. Division of the homogeneous coordinates by w in Eq. (4) gives the pixel coordinates in the image.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{4}$$

The position vector of the camera is denoted by $\underline{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$ and its Euler rotations by $yaw = \psi$, $pitch = \theta$ and $roll = \phi$. Since the rotations in Carla are left sided, pitch and roll should be multiplied by -1. The 3D rotation matrix R is given in Eq. 5.

$$R = \begin{bmatrix} c(\theta)c(\psi) & c(\psi)s(\theta)s(\phi) - s(\psi)c(\phi) & -c(\psi)s(\theta)s(\phi) - s(\psi)s(\phi) \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & -s(\psi)s(\theta)c(\phi) + c(\psi)s(\phi) \\ s(\theta) & -c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \tag{5}$$

Then, the extrinsic matrix Eq. (6) defines the projection from camera 3D homogeneous coordinates to world homogeneous coordinates.

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The inverse of the extrinsic matrix gives the projection from world coordinates to camera 3D which can be projected to the image plane with K (Eq. (7)) after the necessary axis swap (camera NED to x-y-depth).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot [R|t]^{-1} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\top} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{7}$$

The connection between 3D world features and their corresponding image pixel coordinates which is given in Eq. (7) is crucial in all image-based navigation methods.

IV. Optical navigation use-cases

Our VIL system is designed to test and evaluate the possible use-cases of a companion drone, which supports the safety of the main mission drone, by enhancing its navigation capabilities by redundancy. First, the possible approaches of optical navigation are briefly discussed, then a Perspective-n-points (PnP)-based solution is shown in details, and finally a use-case, where the additional optical navigation sensor is placed on a companion drone.

A. Optical navigation methods

In optical navigation the 3D position and orientation of the camera (fixed relative pose to drone) is determined based on image feature points. If the 3D world coordinates of the features are known, ($n \geq 4$) detections in one image are necessary to compute the 6D pose of the camera in world coordinates. This task is called perspective-n-Points (PnP) problem which has several solvers [18].

If the world coordinates of the features are not known, relative navigation can be performed from image to image through paired feature points. In this case, the 3D rotation and the direction of movement can be determined w.r.t. the first image coordinate system. In the most general case, 8 point pairs are sufficient to give the relative transformation (Fundamental matrix) [19]. There are numerical stability problems with the 8-points algorithm, thus RANSAC and other optimization methods are applied to calculate the Fundamental matrix from two views [20]. The camera matrix (eq. 3) can be calculated through camera calibration which decreases the open parameters of the transformation, and only the Essential matrix should be defined. For the Essential matrix, there are stable solvers [21], and camera calibration can be performed for cameras with automated tools which are available for any UAV application.

B. Optical navigation sensor on mission drone

The mission drone has an on-board camera for its main task which is interrupted for 2-4 seconds in each minute to check navigation sensor consistency. An image buffer is continuously saved, and triangulation of paired image features can be calculated from recorded navigation data. After the triangulation, these features are tracked at high frame rate and the PnP solution provides the optical navigation output.

OpenCV is used for the PnP computation in which solvePnPRansac function performs PnP with RANSAC for optimization with outlier detection. The solvePnPRansac function returns the 3D rotations as a vector $rvec'$ and the translation vector $tvec'$. The results should be interpreted in CARLA coordinate system. Rodrigues transformation is applied on $rvec'$, which results in a rotational matrix R'. Then, by R' and $tvec'$ the extrinsic matrix $E[R|t] = [R'|tvec']^{-1}$ can be defined. The translation vector $tvec$ can be derived from $E[R|t]$. The rotation matrix according to the coordinate system of the CARLA simulator R^{SIM} is given in Eq. (8)

$$R^{SIM} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}^{\top} \cdot R \cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \quad (8)$$

Since R^{SIM} is a rotation matrix, the following relations exist for the Euler angles:

$$\begin{aligned} R_{11}^{SIM} &= \cos(\theta)\cos(\psi) \\ R_{21}^{SIM} &= \cos(\theta)\sin(\psi) \\ R_{31}^{SIM} &= \sin(\theta) \\ R_{32}^{SIM} &= -\cos(\theta)\sin(\phi) \\ R_{33}^{SIM} &= \cos(\theta)\cos(\phi) \end{aligned}$$

First, the pitch can be derived from R_{31}^{SIM}. Then, using $\frac{R_{32}^{SIM}}{R_{33}^{SIM}} = -\tan(\phi)$ the roll angle is defined. And finally, yaw is coming from $\frac{R_{21}^{SIM}}{R_{11}^{SIM}} = \tan(\psi)$.

C. Optical navigation sensor on companion drone

A companion drone follows the main mission drone and gives redundancy to its main task, and for 2-4 seconds in each minute checks the relative position of main drone and sends this information to give consistency check capability to the main drone without additional computation burden. In our VIL setup, this drone is virtual, with an artificial camera. This setup can decrease the cost of flight tests during development in the case of outdoor VIL. The physical dimensions of the main drone is known, and the companion also has self-checks for 2-4 seconds as described in previous subsection.

Without additional markers on the main drone, only the oriented bounding box can be detected with real-time object detectors. For quad- and hexa-copters the 3D shape of the UAV is a disk. Furthermore, this disk approximation is more or less also true for fixed-wing aircraft [22]. It means that the 3D features which has the extremal points of the projection of the object in the image, represents the same physical distance (physical dimensions of main drone is known). Figure (2) shows the plane in 3D which is defined by the two vectors pointing from the camera origin to one extremal point and the center of the detection of the main drone on the image sensor. These two vectors ($\overline{AC}$, $\overline{AB}$) can be defined based on the pixel coordinates of the points. The z coordinate is f in mm and the x and y coordinates come from $\frac{\text{pixel}}{\text{mm}}$ of the calibrated

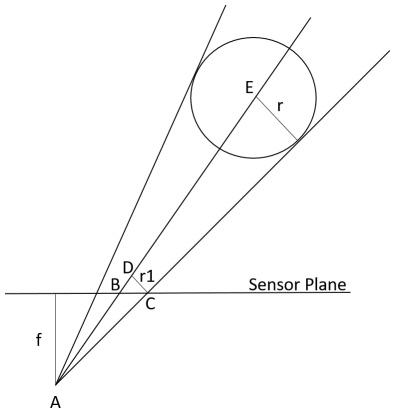

Fig. 2. Distance and relative orientation calculation of an object with known physical size based on image of a calibrated camera. Point A denotes the focal point of the camera with f focal distance.

camera. After the change from pixel-based image detection to 3D vectors, a kind of image disk with radius ($r1$) is required which then can be connected to the known radius (r) of the object through the intercept theorem. For the following calculations, A is used as the origin because the required $\overline{AE}$ is a relative 3D vector which can be transformed with the camera 3D pose to have the 3D pose of the main drone. The center of the image disk $\overline{D}$ is the intersection of the 3D line with $\overline{AB}$ direction vector and the 3D surface with $\overline{AC}$ normal vector which contains $\overline{C}$. From eq. (9) the scalar k is given and $\overline{AD} = k \cdot \overline{AB}$.

$$\begin{aligned} &\overline{AC}_x \cdot (k \cdot \overline{AB}_x - \overline{AC}_x) + \overline{AC}_y \cdot (k \cdot \overline{AB}_y - \overline{AC}_y) + \\ &+ \overline{AC}_z \cdot (k \cdot \overline{AB}_z - \overline{AC}_z) = 0 \end{aligned} \quad (9)$$

$$\frac{\|\overline{AD}\|}{\|\overline{AE}\|} = \frac{r1}{r} \quad (10)$$

With $\overline{AD}$, $r1 = \|\overline{AD} - \overline{AC}\|$. Equation (10) uses the intercept theorem to get $\|\overline{AE}\|$. The unit vector $\frac{\overline{AD}}{\|\overline{AD}\|}$ multiplied by $\|\overline{AE}\|$ results the relative position vector $\overline{AE}$.

V. Results

The camera sensor measurements are done in the CARLA simulator, thus this system is used for validation of the optical navigation results. The following measurements can be conducted without a VIL setup in a simulator only scenario with recorded UAV trajectories, because the control of the main drone is not using the virtual camera, however, these measurements are the starting point of an outdoor VIL with closed-loop control. Beyond the demonstration of a functional VIL indoor, these results give insight into the characteristics of optical navigation algorithms.

A. Optical navigation sensor on mission drone

In the two use-cases of this subsection, the main drone has a forward looking camera with 20 degree pitch which has a FIFO buffer for images with on-board navigation data. At the beginning 2-4 second checks, the ORB features are paired in

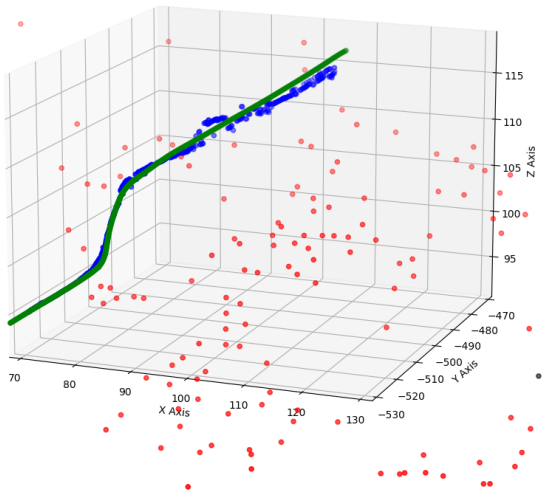

Fig. 3. Real-flight data replay. Green is the ground truth trajectory, blue denotes the optical navigation output, and red dots represents the triangulated 3D feature points (between -200 and 200 meters in Y axis).

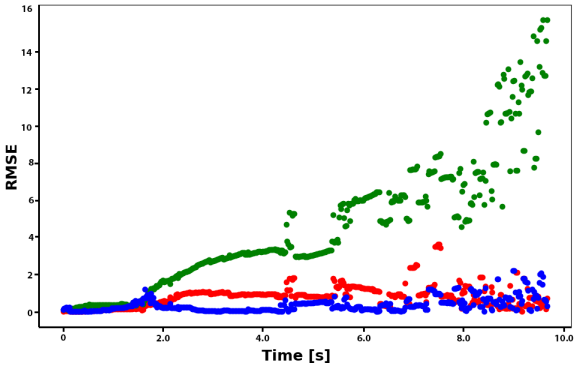

Fig. 4. Root mean squared error [m] of optical navigation output on real-flight data replay. Error component X Y Z are red, green (direction of movement), and blue respectively. Feature distances are around 500m.

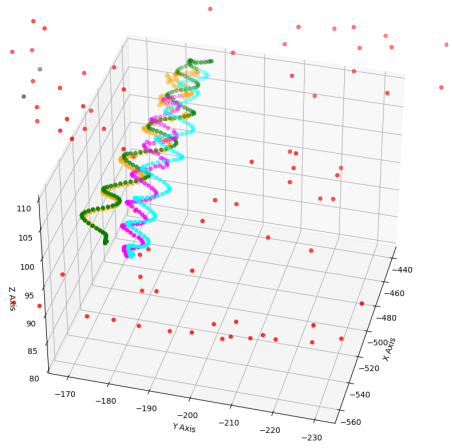

Fig. 5. VIL test on-line upscaled data at 20Hz. Red dots represent the triangulated feature points at around -200m and 200m in X axis. Green is the forward ground truth with orange optical navigation result, cyan is the backward ground truth with purple optical navigation output.

the first and the last frame of the buffer, and 3D position of these features are triangulated. During triangulation and afterward, these triangulated features are tracked by Lucas-Kanade method at 30Hz. ORB feature pairing and triangulation takes more time on the payload computer, but tracking can be started immediately after ORB feature detection. Based on the Kanade–Lucas–Tomasi tracks, the features with known 3D positions are the inputs of the PnP solver. In the indoor VIL setup of this paper, all calculations are done on the simulator desktop PC, because the CrazyFlie drone is not capable of carrying an on-board payload computer.

In figure (3), recorded data on a fixed-wing UAV is used which trajectory is placed above the CARLA city at 107m altitude.The path takes approximately 200m in Y direction at 20m/s ground speed. This setup is a kind of worst-case, because the absolute error of optical navigation is proportional to the distance of feature points from the camera (500m in this case).

Figure (4) shows the position error of the PnP-based optical navigation sensor. It is clear that the X and Z errors are sufficient, however, the precision in the direction of movement decreases rapidly and can be utilized only in the first 2-4 seconds.

Figure (5) presents a VIL measurement result in which the MAV flies a forward backward line with sinusoidal perturbation. The flight was performed by a human pilot in the drone arena and the real 3D pose from the motion capture subsystem was upscaled to have a reasonable movement above the CARLA city.

The results of figure (6) suggest that the degradation of the precision in the direction of movement only depends on the travelled distance from the two camera positions of initial triangulation, however, many features are lost during the maneuver which leads to extremely high uncertainty at the end of the backward route.

B. Optical navigation sensor on companion drone

In this case a companion drone flies 15m behind the main drone and continuously tracks it in the image of its on-board camera. This object detection and tracking can be done by real-time object detectors like YOLO. In this measurement, instead of an oriented bounding box, the image projection of the ground truth positions of the 6 rotors of the main drone is used to separate the precision of the optical navigation approach from the precision of the oriented bounding box of a neural net. The disk approximation causes inaccuracies

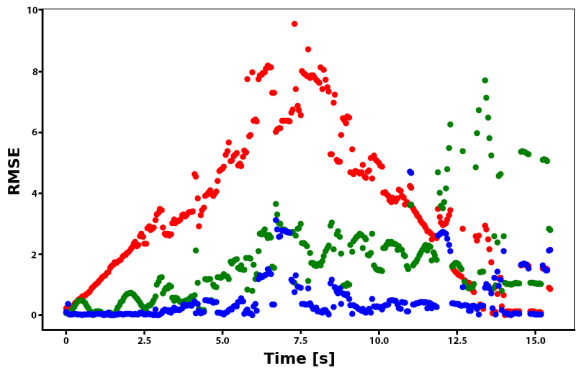

Fig. 6. Root mean squared error [m] of optical navigation output on VIL test on-line upscaled data. Error component X Y Z are red (direction of movement), green, and blue respectively. Feature distances are around 500m and the frame rate is 20Hz.

even for a hexacopter. Figure (7) shows the trajectory of the main drone and the companion, with the companion-based navigation result of the main drone.

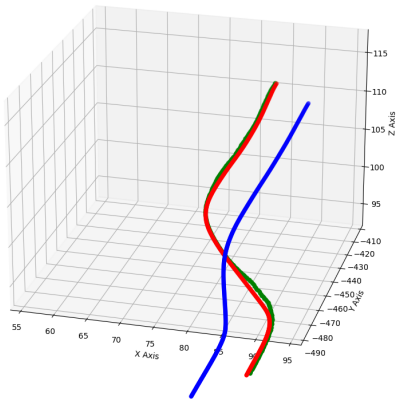

Fig. 7. Real-flight data (red) replay with artificial companion (blue). Red denotes the result of the optical navigation of main drone from the sensor on the companion.

The main drone has an aggressive roll and yaw during the turn in X (around t=2 sec). This changes the width of its image projection regardless of the same relative distance when the rotors arrange differently (disk model conflict). Error measurement of figure 8 shows that the disk model conflicts mainly contributes to the error of the direction of movement (same as direction of main drone from companion), and has minimal error on the remaining two axis.

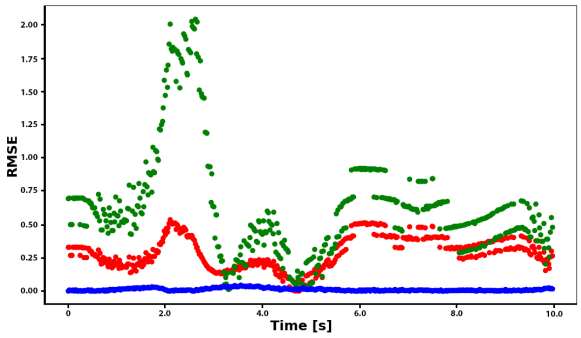

Fig. 8. Root mean squared error [m] of optical navigation output for main drone position which is computed on the companion drone. Error component X Y Z are red, green (direction of movement), and blue respectively.

VI. Conclusion

This paper presents an indoor vehicle-in-the-loop simulation of optical navigation sensors with CrazyFlie MAVs. The 3D position and orientation is measured by an OptiTrack system and the drone has a digital twin in an Unreal4 CARLA simulator which follows the movement of the MAV in real-time. The drone performs optical navigation in the simulated environment with its simulated camera sensor. This is done through an image buffer with saved navigation data. The setup of this paper is idealistic in many factors. First, the VIL should be done for real mission drones in outdoor flights where a precise positioning system is not available (on-board RTK GNSS and sensor fusion can provide position for simulated camera and LIDAR), and wireless communication has bandwidth and latency burdens. This paper focuses on the raw performance of an on-board real-time optical navigation sensor before sensor fusion and indoor VIL with closed-loop control. The results show that the PnP-based approach can provide pose data with acceptable precision in the first 2-4 seconds, and provides reasonable precision except the direction of movement for consistency check in each minute. The companion drone use-case shows the capabilities of the disk model for calculating 3D relive position of hexa-copters. This framework is the prototype of future outdoor VIL solutions where dangerous mission tasks can be tested at safe real test airfields.

References

[1] T. Bokc, M. Maurer, and G. Farber, "Validation of the vehicle in the loop (vil); a milestone for the simulation of driver assistance systems," in *2007 IEEE Intelligent vehicles symposium*, pp. 612–617, IEEE, 2007.

[2] T. Tettamanti, M. Szalai, S. Vass, and V. Tihanyi, "Vehicle-in-the-loop test environment for autonomous driving with microscopic traffic simulation," in *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 1–6, IEEE, 2018.

[3] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, 2022.

[4] Z. Bilgin, M. Bronz, and I. Yavrucuk, "Experimental evaluation of robustness of panel-method-based path planning for urban air mobility," in *AIAA AVIATION 2022 Forum*, p. 3509, 2022.

[5] S. Gazdag, A. Kiskaroly, T. Sziranyi, and A. L. Majdik, "Autonomous racing of micro air vehicles and their visual tracking within the micro aerial vehicle and motion capture (MIMO) arena," in *ISR Europe 2022; 54th International Symposium on Robotics*, pp. 1–8, 2022.

[6] I. Lugo-Cardenas, S. Salazar, and R. Lozano, "The mav3dsim hardware in the loop simulation platform for research and validation of uav controllers," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1335–1341, IEEE, 2016.

[7] S. Khaliq, S. Ahsan, and M. D. Nisar, "Multi-platform hardware in the loop (hil) simulation for decentralized swarm communication using ros and gazebo," in *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 310–315, IEEE, 2021.

[8] M. Akcakoca, B. M. Atici, B. Gever, S. Oguz, U. Demirezen, M. Demir, E. Saldiran, B. Yuksek, E. Koyuncu, R. Yeniceri, *et al.*, "A simulation-based development and verification architecture for micro uav teams and swarms," in *AIAA Scitech 2019 Forum*, p. 1979, 2019.

[9] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*, pp. 621–635, Springer, 2018.

[10] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, *et al.*, "Benchmarking 6dof outdoor visual localization in changing conditions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8601–8610, 2018.

[11] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle, "Lens: Localization enhanced by nerf synthesis," in *Conference on Robot Learning*, pp. 1347–1356, PMLR, 2022.

[12] C. Pfeiffer, S. Wengeler, A. Loquercio, and D. Scaramuzza, "Visual attention prediction improves performance of autonomous drone racing agents," *Plos one*, vol. 17, no. 3, p. e0264471, 2022.

[13] N. Piasco, J. Marzat, and M. Sanfourche, "Collaborative localization and formation flying using distributed stereo-vision," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1202–1207, IEEE, 2016.

[14] "Bitcraze crazyflie 2.1 versatile open source flying development platform data sheet rev 3." https://www.bitcraze.io/documentation/hardware/crazyflie_2_1/crazyflie_2_1-datasheet.pdf. Accessed: 2023-01-23.

[15] 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)*, pp. 3299–3304, IEEE, 2017. Software available at https://github.com/USC-ACTLab/crazyswarm.

[16] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[17] A. Hiba, P. Bauer, M. Nagy, E. Simonyi, A. Kisari, G. I. Kuna, and I. Drotar, "Software-in-the-loop simulation of the forerunner uav system," *IFAC-PapersOnLine*, vol. 55, no. 14, pp. 139–144, 2022. 11th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2022.

[18] 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, no. 2, pp. 155–166, 2009.

[19] R. I. Hartley, "In defence of the 8-point algorithm," *Proceedings of IEEE International Conference on Computer Vision*, pp. 1064–1070, 1995.

[20] D. Barath, "Five-point fundamental matrix estimation for uncalibrated cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 235–243, 2018.

[21] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[22] P. Bauer, "The 'sense and avoid' aircraft system based-on a monocular camera as the last chance to prevent accidents," in *Intelligent and Safe Computer Systems in Control and Diagnostics* (Z. Kowalczuk, ed.), (Cham), pp. 367–385, Springer International Publishing, 2023.