Tracking and Relative Localization of Drone Swarms with a Vision-based Headset

Maxim Pavliv[1,2], Fabrizio Schiano[2], Christopher Reardon[3], Dario Floreano[2], and Giuseppe Loianno[1]

***Abstract*—We address the detection, tracking, and relative localization of the agents of a drone swarm from a human perspective using a headset equipped with a single camera and an Inertial Measurement Unit (IMU). We train and deploy a deep neural network detector on image data to detect the drones. A joint probabilistic data association filter resolves the detection problems and couples this information with the headset IMU data to track the agents. In order to estimate the drones' relative poses in 3D space with respect to the human, we use an additional deep neural network that processes image regions of the drones provided by the tracker. Finally, to speed up the deep neural networks' training, we introduce an automated labeling process relying on a motion capture system. Several experimental results validate the effectiveness of the proposed approach. The approach is real-time, does not rely on any communication between the human and the drones, and can scale to a large number of agents, often called swarms. It can be used to spatially task a swarm of drones and also employed without a headset for formation control and coordination of terrestrial vehicles.**

***Index Terms*—Aerial Systems: Applications, Human-Centered Robotics, Localization**

I. Introduction

MICRO Aerial Vehicles (MAVs), often called drones, represent a promising technology in a wide range of applications such as aerial photography, mapping and topography, delivery, search and rescue, inspections in hard-to-reach environments, and border control [1]. In particular, aerial swarms can multiply the capabilities of single-drone systems [2] speeding up the overall task and increasing the overall system's resilience.

In parallel with the efforts to make drones more autonomous [3], [4], there is an increasing interest in intuitively controlling them. Human-Robot Interfaces (HRIs) define the way commands are sent to a robotic system and how a human user receives feedback. Most HRIs let humans control robots via remote controllers, joysticks, keyboards, and mice and receive feedback from the robots on screens, with LEDs, or sound. In the effort to make HRI more intuitive, researchers have experimented with Body-Machine Interfaces that rely on body motion to control robots and on haptic feedback, and virtual or augmented reality headsets [5], [6], [7], [8], [9]. However, most studies refer to a single human interacting with a single robot and/or do not explicitly address the relative localization of robots with respect to the human, which was only partially tackled in our recent work [10]. Human-swarm interaction is only at its dawn. A recent study [11] showed a human operator controlling a drone swarm through a decentralized connectivity-maintenance approach. Instead, [12] describes a human operator controlling multiple drones with hand gestures and employing cameras available on each robot to estimate the relative location between the human and the drones by face recognition. This approach is problematic since it requires the user to be in the field of view of each swarm agent. Finally, in [13] the authors propose a novel control technique that combines impedance control and vibrotactile feedback. From all these works, it is clear that the human has to self-localize with respect to the drones to establish an interaction paradigm. However, previous studies often assume that the drones access their relative positions with respect to the human through an external localization system, such as a motion capture system. This represents an important limiting factor when deploying the system in unknown and unstructured environments.

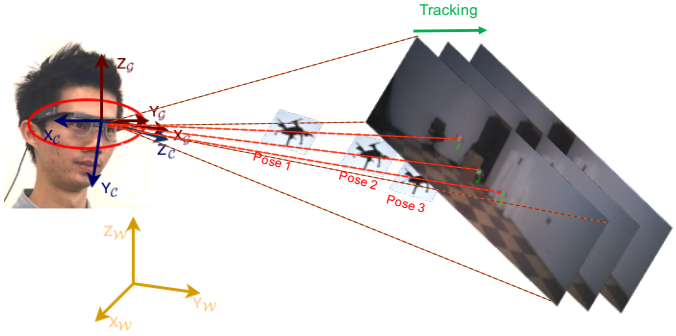


Fig. 1: Representation of our tracking and localization system.

The human-to-drones relative localization problem (i.e., human understanding their relative pose with respect to each drone of the swarm) is very similar to a single drone that must compute its relative localization with respect to other members of the swarm. Cameras and IMUs have been regularly employed for this task due to the vehicles' Size, Weight, and Power (SWaP) constraints. Several studies leverage algebraic

Manuscript received: October, 15, 2020; Revised December, 13, 2020; Accepted December, 13, 2020.

This paper was recommended for publication by Editor Pauline Pounds upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Qualcomm Research, the ARL grant DCIST CRA W911NF-17-2-0181, the Technology Innovation Institute, Nokia, and NYU Wireless. Also, it was supported by the Swiss National Science Foundation (SNSF) with grant number 200021-155907.

[1]The authors are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA. email: {mp5516,loiannog}@nyu.edu.

[2]The authors are with the Ecole Polytchnique Federale de Lausanne, Station 9 CH-1015 Lausanne, Switzerland. email: {fabrizio.schiano,dario.floreano}@epfl.ch.

[3]The author is with the U.S. Army Research Laboratory, 2800 Powder Mill Road, Adelphi, MD 20783, USA. email: christopher.m.reardon3.civ@mail.mil.

graph theory using either bearing [14], [15] or distance [16] measurements for the control and relative localization of the agents of the formations. However, those studies do not employ on board perception and estimation and rely on an external motion capture system to detect and track the agents. Some studies address the drone-to-drone relative localization problem by leveraging artificial markers (typically visual tags) [17], [18], [19] mounted on each robot to estimate their relative positions exploiting the markers' geometric properties. However, although the use of tags can be very useful in laboratory settings, these do not scale well in real–world settings with numerous agents due to the limited number of unique tag IDs or geometric shapes and the difficulty of equipping each robot with a tag. Finally, other works [20], [21] achieve indirect cooperative localization by sharing a set of characteristic features or landmarks across the agents. However, despite these approaches work without line of sight requirements, they need communication among the agents and require to store local maps.

In summary, none of the previous studies provides humans with the ability to automatically detect, track and localize several drones at the same time, without relying on an external infrastructure system, and in real–time.

We build on our previous work [10], where we spatially task a single vehicle in an indoor environment. However, we did not address the tracking and pose estimation problems from the human perspective , which are solved in the current work for complex multi-vehicles system. Estimating the full pose can be crucial in multi-agent systems with directional sensors or ones that need to interact with the environment physically. We propose a lightweight, vision-based system able to detect, track, and localize several drones in 3D space with respect to a human equipped with a headset featuring a camera and an IMU (eye–tracking glasses in our case). This work presents multiple novel contributions. First, we show the ability to deploy in real–time a Deep Neural Network (DNN) for drone detection. Second, we describe a Joint Probabilistic Data Association Filter (JPDAF) that tracks multiple drones in the image. The filter is able to solve the data association problem between the tracked objects and the measurements obtained from the DNN. Third, we complement the tracker with a 6–DoF (Degrees of Freedom) pose estimator based on semantic keypoints extracted by a separate DNN. To speed up the training process of both DNNs, we introduce an automated labeling process. Finally, we show several experiments in indoor scenarios to relatively localize the user with respect to each agent of the swarm. Our method runs in real–time on a small Nvidia Jetson TX2 module connected with the headset and solves the relative localization problem relying exclusively on local information using only the headset camera and IMU data (i.e., no external localization system is available to the drones and the drones do not communicate with each other or with the user). Our approach can be deployed on–demand since the detection, tracking, and estimation of the drones' state can be initialized at any time and without relying on any *a priori* information or triangulation strategies. Overall, this approach enables detection, tracking, and localization of multiple drones in real–time, making it possible to use it in environments without external positioning systems. Our experiments show this capability in the scope of human-swarm interaction, with numerous impactful applications, such as security, surveillance, inspection, and search and rescue. Nevertheless, the applicability of this approach extends to any domain that involves detecting, tracking, and localizing swarm agents (e.g., tracking potentially dangerous rogue, unknown MAVs in airplane flight paths). Ultimately, this work will help bring human-swarm research out of the laboratory setting and allow safe, fast, and efficient multi–modal interaction between humans and swarms in a multitude of real-world domains.

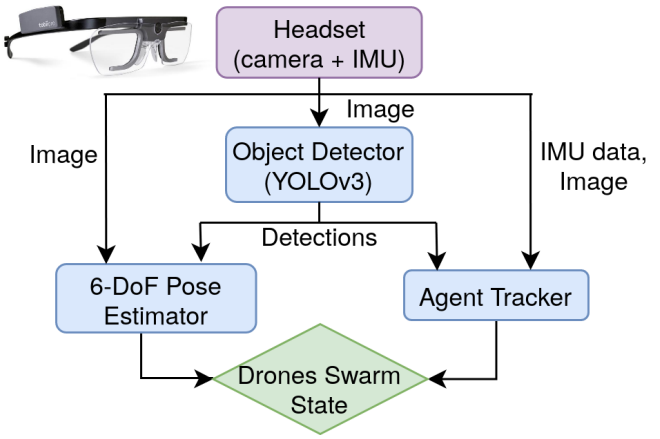


Fig. 2: Overview of the proposed pipeline.

The paper is organized as follows. Section II presents an overview of the proposed system while the overall methodology is described in Section III. Section IV presents the experimental results, whereas Section V discusses the conclusions.

II. System Overview

A. Preliminaries

As shown in Figs. 1 and 3, we define the camera frame denoted with $\mathcal{C}$, the robot frame with $\mathcal{D}$, the glasses' frame with $\mathcal{G}$, and the world frame with $\mathcal{W}$. Our algorithm provides the state of the swarm (i.e, tracking state and pose information estimates of each agent in the swarm) directly in the $\mathcal{C}$ frame. To compare the results with respect to motion capture system data, the pose of the robot and the camera one should be expressed with respect to the $\mathcal{W}$ frame. Denoting with $\bar{\mathbf{g}}_{\mathcal{AB}}$ the relative pose between a frame $\mathcal{A}$ and $\mathcal{B}$ expressed as a homogeneous transformation matrix, we build $\bar{\mathbf{g}}_{\mathcal{WG}}$ and $\bar{\mathbf{g}}_{\mathcal{WD}}$ with the tracking information provided by the motion capture system. An external calibration of the device is used to compute $\bar{\mathbf{g}}_{\mathcal{CG}}$. We compute $\bar{\mathbf{g}}_{\mathcal{CD}}$ the homogeneous representation of the change of the coordinates from the drone frame to the camera frame as

$$\bar{\mathbf{g}}_{\mathcal{CD}} = \bar{\mathbf{g}}_{\mathcal{CG}}\bar{\mathbf{g}}_{\mathcal{GW}}\bar{\mathbf{g}}_{\mathcal{WD}} = \bar{\mathbf{g}}_{\mathcal{CG}}\bar{\mathbf{g}}^{-1}_{\mathcal{WG}}\bar{\mathbf{g}}_{\mathcal{WD}} \tag{1}$$

B. Hardware and Software Setup

We use the camera and Inertial Measurement Unit (IMU) data available on the headset. The IMU rate is 160 Hz, whereas the image data is 25 Hz. A drone detector is responsible to output bounding boxes of the drones' locations in the image space along with their confidence levels. We use a custom

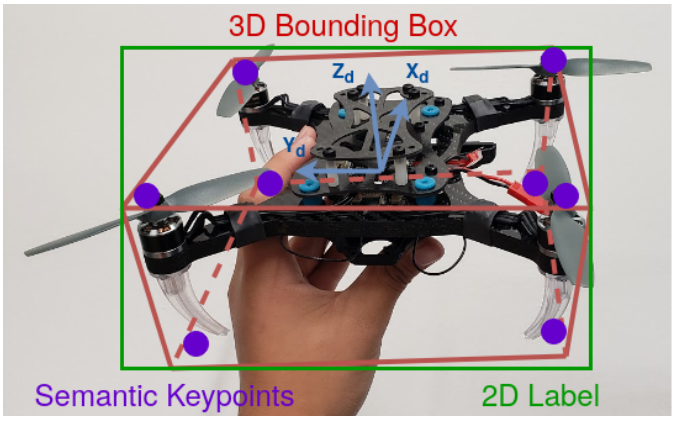


Fig. 3: 3D drone bounding box defined in the $\mathcal{D}$ frame.

small–scale drone from our previous work [3] for testing purposes. Once the drone is detected, the object tracker module is responsible for detecting and tracking each swarm agent in space and time and consequently associating a unique ID to each drone. The 6-DoF pose estimator computes the relative position and orientation of detected drones with respect to the user. The tracking and the pose data provide the state of the swarm agents and can potentially be employed for interaction. Our approach does not use any external localization system (e.g., GNSS, Motion Capture System) except during the detector's training phase. Importantly, our system is also lightweight and does not require a ground station since the algorithms can run on an Nvidia Jetson TX2 module. A Vicon[1] motion capture system is used only to capture ground truth data during our experiments and as an instrument to facilitate the creation of labeled datasets for the training of the drone detector and the semantic keypoints estimator (6-DoF pose estimator), as explained in Section III. Our algorithms are implemented in ROS using C++ and Python.

III. Methodology

In this section, we provide an overview of the hardware and software components of our system.

A. Automatic Dataset Generator

In order to detect the swarming drones in the headset camera stream, we employ the YOLOv3 algorithm [22]. This approach identifies each agent by defining a bounding box around it in the camera image plane. This is a supervised learning approach with a major drawback, namely the requirement of a large amount of training data to detect the agents accurately. The data is usually generated by a person who manually selects the areas of interest in the image. In this work, we design a procedure that automatically labels data to train both the object detector and the semantic 6-DoF estimator based on semantic keypoints. The latter uses a double-stacked hourglass network (see Section III-C). Our training procedure leverages a Vicon motion capture system to generate accurate ground-truth data as visualized in Fig. 3. It starts by first creating a 3D bounding box around the drone. Subsequently, we express its 8 edges as coordinates in the drone frame $\mathcal{D}$, and we project them on the camera image plane of the glasses by tracking the drone's and glasses' positions via motion capture system. The projections can then build a 2D bounding box directly used for the object detector training process or as semantic keypoints for the semantic keypoints stacked hourglass network training employed for pose estimation.

[1]www.vicon.com

Algorithm 1 JPDAF pipeline

1: **Prediction Step**: At every iteration, each track computes its predicted state (Kalman filter prediction step)
2: **Update Step**: Each track in the state is updated according to the following modified Kalman filter update steps
- The algorithm builds the association matrix to pair the measurements with the feasible tracks
- The association matrix is used to build the hypothesis matrices to identify the measurements/tracks one-to-one combinations. Each matrix corresponds to a feasible joint event of the measurements' process
- A probability of occurrence is then computed for each joint event
- The algorithm computes the weights corresponding to the influence of each measurement on each track

B. Multi-Agent Tracking

Once the swarm's agents have been detected, the main goal is to track each agent in space and time. All the objects (drones) being undifferentiable, the tracker's role is to attribute a different ID to each of the detected drones, and track the swarm along the stream of images. To achieve this goal, once a unique ID is assigned to each vehicle, it is necessary to associate the incoming measurements to the robots' tracks. We propose to achieve this goal by using a Joint Probabilistic Data Association Filter (JPDAF) [23], [24], [25] (see Algorithm 1).

1) Prediction Step: Each detected object is tracked employing a 4-dimensional vector, containing the track positions and velocities along the u and v axes of the image plane

$$\mathbf{x}_k = \begin{bmatrix} p_u & \dot{p}_u & p_v & \dot{p}_v \end{bmatrix}^\top. \tag{2}$$

At each algorithm iteration, each track motion is driven by a modified Kalman filter with a constant speed motion model

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{Q}_k, \tag{3}$$

where $\mathbf{A}_k$, $\mathbf{B}_k$ are the state transition and control input models respectively, and $\mathbf{Q}_k$ the process noise covariance matrix

$$\mathbf{A}_k = \begin{bmatrix} 1 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q}_k = q \begin{bmatrix} \frac{\delta t^2}{2} & 0 & 0 & 0 \\ 0 & \delta t & 0 & 0 \\ 0 & 0 & \frac{\delta t^2}{2} & 0 \\ 0 & 0 & 0 & \delta t \end{bmatrix},$$

$$\mathbf{B}_k = \delta t \begin{bmatrix} \frac{(u-c_u)(v-c_v)}{f} & -\frac{(u-c_u)^2}{f} - f & v - c_v \\ 0 & 0 & 0 \\ f + \frac{(v-c_v)^2}{f} & -\frac{(u-c_u)(v-c_v)}{f} & -u + c_u \\ 0 & 0 & 0 \end{bmatrix}, \tag{4}$$

where δt is the sampling time in s, q is the acceleration of the drones in px/s^2 assumed to be a Gaussian random variable, (c_u, c_v) are the principal point coordinates, f is the focal length, and $\mathbf{u}_k = \mathbf{B}_k \boldsymbol{\omega}_{cam}$ with $\boldsymbol{\omega}_{cam}$ the camera angular velocity obtained using gyro data. The model in eq. (3) is obtained by combining a constant speed motion model in the image plane with optical flow motion field equations. This strategy allows us to compensate for camera 3D rotations using IMU gyro data, while considering a constant velocity model in the image plane for each agent. Taking into account camera rotations in our model makes the tracking more robust (see Section IV). Specifically, let us consider a camera frame moving with respect to an inertial frame with linear and angular velocities respectively $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^\top$ and $\boldsymbol{\omega}_{cam}$. The velocity of a 3D point $p_c = (X, Y, Z)$ expressed in the camera frame is

$$\dot{p}_c(t) = -\boldsymbol{\omega}_{cam} \times p_c(t) - \mathbf{v}. \tag{5}$$

Considering the camera projective equations

$$p_u = f\frac{X}{Z} + c_u, \quad p_v = f\frac{Y}{Z} + c_v, \tag{6}$$

we compute their time derivative and we plug eq. (5) in it and take the rotational component of the resulting expression. This provides the $\mathbf{B}_k \mathbf{u}_k$ term in eq. (3).

2) Update Step: The detector provides bounding boxes around recognized objects on each frame, without any specific association between current and past detections. Therefore, it is necessary to solve the association problem between measurements and the corresponding track in the state. The observation model is the bounding box centroid position

$$\mathbf{z}_k = \begin{bmatrix} p_u \\ p_v \end{bmatrix} = \mathbf{C}_k \mathbf{x}_k + \eta, \tag{7}$$

and η is assumed to be Gaussian white noise $\eta \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. The update of the tracks' state is done across all tracks and all detections in a joint manner. We define the innovation vector $\mathbf{y}_k^i$ as the innovation of one track by the measurement $\mathbf{z}_k^i$

$$\mathbf{y}_k^i = \mathbf{z}_k^i - \mathbf{C}_k \boldsymbol{\mu}_k \tag{8}$$

where $\boldsymbol{\mu}_k$ is the predicted state mean. We define β_k^i as the weight of the influence of the measurement i on the track t, corresponding to the probability that the measurement i was generated by the track t. Furthermore, we define β_k^0 as the probability that no measurement has been generated by the track t. For each track, these coefficients respect the probability equation

$$\sum_{i=1}^{m_k} \beta_k^i + \beta_k^0 = 1. \tag{9}$$

where m_k is the number of measurements (detections). The combined innovation vector $\mathbf{y}_\mathrm{k}$ for a track is computed as

$$\mathbf{y}_k = \sum_{i=1}^{m_k} \beta_k^i \mathbf{y}_k^i, \tag{10}$$

and the state $\mathbf{x}_k$ is updated according to

$$\mathbf{x}_k \leftarrow \mathbf{x}_k + \mathbf{K}_k \mathbf{y}_k. \tag{11}$$

The predicted state covariance $\mathbf{P}_k$ is updated accoding to

$$\mathbf{P}_k \leftarrow \beta_k^0 \mathbf{P}_k + (1 - \beta_k^i)\mathbf{P}_k^c + \tilde{\mathbf{P}}_k, \tag{12}$$

where

$$\mathbf{P}_k^c = \mathbf{P}_{k,k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\top, \tag{13}$$

with $\mathbf{S}_\mathrm{k} = \mathbf{C}\mathbf{P}_{\mathrm{k,k\text{-}1}}\mathbf{C}^T + \mathbf{R}$ the innovation covariance, $\mathbf{P}_{k,k-1}$ the predicted covariance, $\mathbf{P}_k^c$ is the covariance of the state updated with the correct measurement, and $\tilde{\mathbf{P}}_k$ is the updated covariance due to the uncertainty of the association

$$\tilde{\mathbf{P}}_k = \mathbf{K}_k \left(\sum_{i=1}^{m_k} \beta_k^i \mathbf{y}_k^i \mathbf{y}_k^{i\top} - \mathbf{y}_k \mathbf{y}_k^\top \right) \mathbf{K}_k^\top. \tag{14}$$

To compute the update weights β_k^i, we first build the association matrix $\mathbf{I}$, representing the possible associations between the measurements and the tracks. Each row corresponds to a measurement, and each column to a track. The first column corresponds to clutter noise generated detections. If an element $a_{i,j}$ of the association matrix $\mathbf{I}$ is equal to 0, the measurement i is not associated to source j (track or clutter noise). Instead, if $a_{i,j} = 1$, then they are associated. The elements of the association matrix are 1 if the following gating condition is verified

$$V(k, \gamma) := \{\mathbf{z}_k^i : (\mathbf{z}_k^i - \mathbf{C}_k \mathbf{x}_k)^\top \mathbf{S}_k^{-1} (\mathbf{z}_k^i - \mathbf{C}_k \mathbf{x}_k) \leq \sigma^2\}. \tag{15}$$

Finally, the algorithm performs track management (i.e., it creates new tracks if needed, gives IDs to confirmed tracks, and deletes deprecated tracks). Once the association matrix is built, the hypothesis matrices are generated $\hat{\mathbf{I}}(\chi)$. Each joint event χ_i is represented by a hypothesis matrix $\hat{\mathbf{I}}_i = [\hat{v}_{jt}(\chi)]$. A joint event is defined as

$$\chi = \bigcap_{j \in M} \chi_{j,t_j}, \tag{16}$$

where χ_{j,t_j} is the event that the measurement j originated from target t_j, $0 \leq t_j \leq T$, and t_j is the index of the target to which the measurement j is associated. In the JPDAF, the probabilities of joint events indicate if each measurement is either associated to one target, or to the clutter noise, and only feasible events are used. A feasible joint event in this particular case is an event where no target has more than one measurement associated to it

$$j \neq l \text{ and } t_\mathrm{j} > 0 \text{ implies } t_\mathrm{j} \neq t_\mathrm{l}, \tag{17}$$

where each measurement is associated to either one target or to clutter noise. In a hypothesis matrix, an element $a_{i,j}$ equal to 0 means that the measurement i was not generated by the target j, and an element $a_{i,j}$ equal to 1 means that the measurement i was generated by the target j, within this particular joint event. The feasibility condition constraints impose the corresponding hypothesis matrix to have at most one element equal to 1 per column (with the first column as exception to this rule), and that each row has exactly one element equal to 1 (each measurement was generated by one source, and a target can generate at most one measurement). The hypothesis matrices generation is a combinatorial problem under the feasibility constraint. The gating step previously mentioned limits the

Example 3.1: Let us take the following association matrix

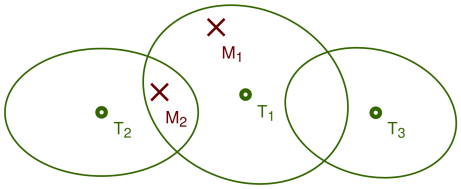
$$\mathbf{I} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

This matrix would generate the following hypothesis matrices

$$\hat{\mathbf{I}}(\chi_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \ \hat{\mathbf{I}}(\chi_2) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix},$$

$$\hat{\mathbf{I}}(\chi_3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \ \hat{\mathbf{I}}(\chi_4) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\hat{\mathbf{I}}(\chi_5) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$



number of joint events to consider, which otherwise grows exponentially with the number of tracks and measurements. A sample of this process is provided in Ex. 3.1 for two measurements and three tracks.

After generating the hypothesis matrices corresponding to the joint events, we compute the probability of each joint event to be the one that occurred in reality. We define two indicators

- The measurement association indicator denotes whether measurement j is associated with any track in the joint event χ

$$\tau_t(\chi) := \begin{cases} 1, & \text{if } t_j > 0 \\ 0, & \text{if } t_j = 0 \end{cases} \tag{18}$$

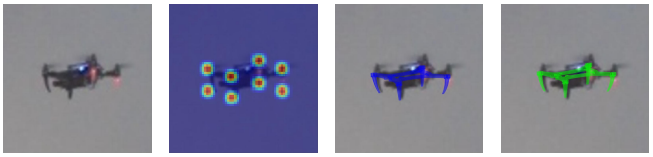
- The target association indicates whether any measurement is associated with the target t in the joint event χ

$$\delta_t(\chi) := \begin{cases} 1, & \text{if } t_j = t \text{ for some } j \\ 0, & \text{if } t_j \neq t \text{ for all } j \end{cases} \tag{19}$$

The number of false positive measurements (generated by clutter noise) are modeled according to a Poisson distribution with rate $\lambda = CV$, where V is the volume of the observed space (the image area in our case) and C the density of false measurements. A measurement (detection) generated by a target is modeled by a k-dimensional normal distribution (in our case $k = 2$). At each time step, each target may or may not be detected. The target detection probability is denoted with P_{D}. Using these assumptions,the probability corresponding to each hypothesis matrix can now be computed as

$$P\{\chi|Y_k\} = \frac{C^{\phi}}{c} \prod_{t:\tau_j=1} \frac{e^{-\frac{1}{2}\mathbf{y}_j^{t_j}\mathbf{S}_{t_j}^{-1}\mathbf{y}_j^{t_j}}}{(2\pi)^{\frac{M}{2}}|\mathbf{S}_{t_j}|^{\frac{1}{2}}} \cdot \prod_{t:\delta_t=1} P_D^t \prod_{t:\delta_t=0} (1-P_D^t). \tag{20}$$

with ϕ is the number of false measurements in a joint event and c a normalization constant. A similar derivation leading to the presented results can be found in [23], [24]. The constant



(a) (b) (c) (d)

Fig. 4: Pose estimation steps using semantic keypoints. (a) Input image: cropped detected object (b) Predicted heatmaps for each keypoint shown in Fig. 3, (c) Weak perspective estimated pose, and (d) Full perspective one.

c is unknown, so first we compute the probability of each joint event to be the correct one (using the same c for all probabilities, for example $c = 1$). Then the probabilities need to be normalized such that

$$\sum_{all\chi} P\{\chi|Y_k\} = 1. \tag{21}$$

The weights β_t^j and β_t^0, corresponding to the probability that the track t generated the measurement j or that the track t generated no measurement respectively, can now be computed. The influence of the measurement j on the track t is the sum of probabilities of all joint events where the measurement j was generated by the track t, i.e.

$$\beta_t^j = \sum_{all\chi} P\{\chi|Y_k\} \cdot \hat{v}_{jt}(\chi), \quad \beta_t^0 = 1 - \sum_{j=1}^{m} \beta_t^j. \tag{22}$$

We can define a matrix $\mathcal{B}$, having the same shape as the association matrix, containing all the weights for each measurement and each track, the columns being the tracks and the rows being the measurements. $\mathcal{B}$ can be computed as

$$\mathcal{B} = \begin{bmatrix} \alpha_1 & \beta_1^1 & \dots & \beta_T^1 \\ \alpha_2 & \beta_1^2 & \dots & \beta_T^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_m & \beta_1^2 & \dots & \beta_T^m \end{bmatrix} = \sum_{\text{all events}_{\chi_j}} P\{\chi_j|Y^k\} \cdot \hat{\mathbf{I}}(\chi_j). \tag{23}$$

We break this matrix into weights β_t^j by selecting one column for each track, and by computing the resulting β_t^0 weight.

C. 6-DoF Pose Estimator

To estimate each drone's 6-DoF pose in 3D space, we leverage the approach presented in [26]. Unlike other approaches [27], [28] where the network is trained on the full image, we train our network only on semantic keypoints corresponding to the four propellers and four landing gears of the drone. Training on semantic keypoints instead of the full image provides increased robustness to changes in the drone's appearance (e.g., due to a change or reorganization of the hardware or partial occlusions). The algorithm directly employs semantic keypoints to perform object pose estimation. To further speed up this process, we apply the algorithm on the image region (bounding box area) containing the object obtained using the detector and tracker previously described. We train the hourglass network [29] on the semantic keypoints corresponding to the drone's four propellers and four landing

Fig. 5: Snapshots of the JPDAF tracking three quadrotors at two time instants. The pink crosses represent the measurement (detection), the bold green circles show the tracks positions (with their corresponding ID), and the thin green ellipsoids around each track show the position covariance in the image space.

gears. The pose estimation process (see Fig. 4) is performed in several steps

1) The neural network takes as input the image cropped around the detected drone (Fig. 4a).
2) A neural network (double-stacked hourglass network) predicts a heatmap of positioning for each semantic keypoint (Fig. 4b).
3) Using the predicted positions of each semantic keypoint, the algorithm computes the object's pose using an optimization approach (gradient-based method) (Fig. 4c-d).

Specifically, the pose optimization uses the model of the drone's semantic keypoints in 3D. The uncertainty of the keypoints during the prediction (in 2D) is incorporated to give a relaxation for the errors and noise. The pose optimization is conducted starting by a weak perspective pose estimation (Fig. 4c) and followed by a full perspective pose estimation (Fig. 4d). The weak perspective optimization estimates only the 2 first rows of the rotation matrix and of the translation vector. Therefore, we do not assess any depth information. The full perspective algorithm employs the intrinsic camera parameters instead. It is initialized using the weak perspective pose estimation solution to better condition the nonlinear optimization, and it estimates the full rotation and translation matrix to the drone frame in the camera frame.

IV. Experimental Results

In this section, we report results from experiments conducted in an indoor arena with a flying space of $7 \times 5 \times 4$ m^3 at the Agile Robotics and Perception Lab (ARPL) lab at New York University. We conduct the experiments with 3 quadrotors. The rate limit is set by the object detector (YOLOv3), which runs at 30 Hz on a common laptop and at 5 Hz on the Nvidia Jetson TX2 platform. From a computational point of view the JPDAF approach grows exponentially with the number of agents, but the gating process described in Section IV-A helps to reduce this issue.

A. Multi-agent tracker

In Fig. 5, we present qualitative results of the multi-agent tracker at different time instants. The multimedia material of this paper shows results from extensive experiments in multiple conditions that further show the successful performance of the proposed approach. We quantitatively assess our approach in 3 experiments, as described below.

1) Multi-agent tracker: filtering experiment: It is well known that computer vision algorithms suffer under fast camera rotations, which increase image blur [30]. Typically, fast rotation effects cause degradation in the performance of the detector. Therefore, we present an experiment to show that (a) the JPDAF filters out false negatives (undetected objects) produced by the detector, and (b) we quantify the dependence of this aspect on the angular speed of the glasses. In this experiment, we use a dataset of 28 minutes and 58 seconds. The headset's camera is pointing towards a scene that contains three drones in flight. The drones are autonomously controlled [3]. The user wearing the glasses with the camera is either remaining still, or moving his head (rotational movements) with varying angular speed $\boldsymbol{\omega}_{cam}$. We consider that k detections have disappeared from a frame F_{i-1} to a frame F_i if the number of detections drops across the two frames. After running the tracker on the whole dataset, we analyze the influence of the angular speed norm $\|\boldsymbol{\omega}_{cam}\|_2$ on the detections and track disappearances. To avoid bias or oversampling in our sample distribution of angular velocities (some datasets might have predominantly low values or high values of angular velocities) as shown in Fig. 6, we normalized the disappearances with respect to the density of $\|\boldsymbol{\omega}_{cam}\|_2$.

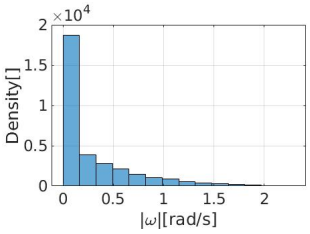
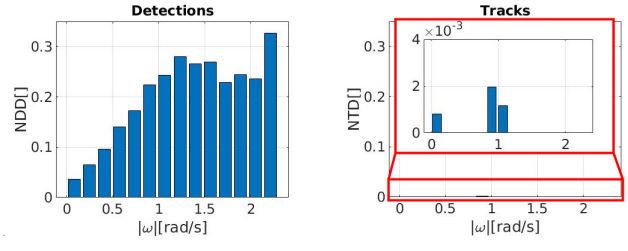


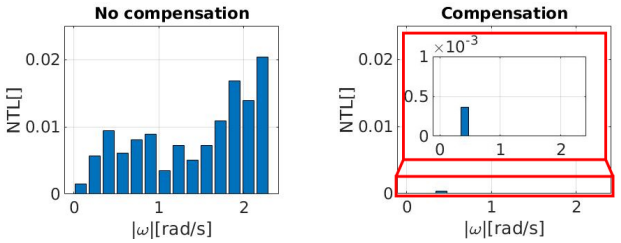
Fig. 6: Density of $\|\boldsymbol{\omega}_{cam}\|_2$ in the dataset.

The Normalized Detection Disappearances (NDD) and Normalized Tracks Disappearances (NTD) are shown in Fig. 7. In Fig. 7a, we notice the negative impact of the angular speed of the glasses on the detection quality, especially in the $\|\boldsymbol{\omega}_{cam}\|_2 \in [0; 1.33]$ rad/s interval. For higher values of $\|\boldsymbol{\omega}_{cam}\|_2$ the detection disappearances remain stagnant. If we compare Fig. 7a and 7b, we see the filtering effect of the tracker. The remaining track disappearances are due to scenarios where a drone is not detected for more than a given number of consecutive updates, after which they are deleted. This event typically happens when the background behind the



(a) (b)

Fig. 7: (a) NDD and (b) NTD as a function of $\|\boldsymbol{\omega}_{cam}\|_2$.



(a) (b)

Fig. 8: NTL without (a) and with (b) user motions' compensation as a function of $\|\boldsymbol{\omega}_{cam}\|_2$.

drone is ambiguous or makes the drone invisible in the image (for the detector and for humans as well). Overall, it is clear that this filtering approach is useful to estimate the drones' positions when the movement of the camera observing the scene causes the detector to fail.

2) Multi-agent tracker: control-input model experiment: The goal of this experiment is to show the importance of the head motion compensation procedure designed as control input in our filter prediction model in eq. (3). We show that the proposed approach explicitly takes into account head motion to improve the tracking performance. When a measurement falls too far from its track, it is not associated to the track anymore (the corresponding β_t^j weight is too small). When no track is associated to a measurement, a new track will be created for this measurement. The new track will get the whole update weight of the measurement (since it is created based on the measurement), and the old track will then be deleted; we call this a *tracking loss*. We can assess that k tracking losses occurred when, in addition to 3 tracks with IDs, k tracks with no IDs are created. We use a pool of IDs such that additional agents can be added at any time. A track is deleted once a given number of iterations with no more measurements associated with it elapsed and the covariance becomes large. Considering a pre-defined number of agents, the IDs do not get switched between the tracks if only one is lost at a time (see supplementary video material) since the re-detected track obtains the past ID. If several tracks are lost simultaneously, the IDs are redistributed in a random manner.

Using the definition of tracking loss, we run the tracker on the same dataset used in our previous experiment (section IV-A1). We analyze the influence of $\|\boldsymbol{\omega}_{cam}\|_2$ on the tracking losses, and compare results without (Fig. 8a) or with (Fig. 8b) compensation for head movements. Similar to the previous experiment, we normalize the tracking losses with respect to the density of $\|\boldsymbol{\omega}_{cam}\|_2$. The Normalized Tracking Losses (NTL), indicating lost and re-assigned tracks (NTD denotes disappeared tracks instead), of the tracker running without compensation (Fig. 8a) show that the performance of the tracker decreases with angular camera motions. Although the histogram's evolution is not smooth, the growing tendency is still visible. The lack of consistency is probably due to the sparseness of the dataset. There is also another backlash phenomenon explaining the lack of smoothness of the curve: at higher angular speeds, we noticed that several tracks were already lost in previous frames, when the angular speed was lower. Conversely, Fig. 8b shows how the incorporation of the angular speed in our motion model can improve the performance of the tracker. After taking it into account, the tracking was lost only once, at an angular speed norm $\|\boldsymbol{\omega}_{cam}\|_2 = 0.3966\ rad/s$. However, a manual check showed that this tracking loss is due to a synchronization issue of the IMU with the image data caused by the inability of the device to allow hardware synchronization among the sensors.

3) Multi-agent tracker: robustness to occlusions: As previously mentioned, each drone is associated with a different track. Each track is made of four variables (two corresponding to the drone position in the image plane and two corresponding to its velocity in the image plane). Based on this 4-dimensional state, it is possible to predict each track's position in the next time step using our motion model in eq. (3). This aspect is useful when the drone passes in front of a cluttered background as well as when it is not detected for several frames or the 3D paths of different drones cause an overlap of the tracks in the image. In this experiment, we show robustness to occlusions. Specifically, we verify that our approach is able to resolve an ambiguous tracking situation when a drone, and the corresponding track, crosses another drone's track or path in the image frame. To test the robustness of the developed tracker in this scenario, we put together the following experiment. The glasses lie on a fixed support and a drone moves relative to the other two drones. The motion of the third drone crosses the hovering drones horizontally or vertically on the image plane. During the experiment, we record the following metrics: the distance between the tracks at crossing time in px; the speed of the moving drone in px/s; and the success or failure of the crossing. We expect bigger distances and velocities during the crossing to lead to higher chances of correct IDs re-attribution as shown in Table I, where we report the mean of the distance between the tracks and the Standard Deviation (STD).

	Distance at crossing $[px]$	**Speed at crossing** $[px/s]$
Mean	4.88	268.01
STD	4.77	71.07

TABLE I: Summary of the robustness to occlusions experiment for the multi-agent tracker (section IV-A3). The dataset has the following properties: total number of crossings = 101; correct ID reattributions = 98; wrong ID reattributions = 3.

The experiments present many cases of full occlusions of the bounding boxes associated to the drones (distance between the drones' bounding boxes centroids equal to 0 px). These cases show that the overlapping area between the bounding boxes is not a limiting factor to correctly re-attribute the IDs.

B. 6-DoF pose estimator: performance experiment

Finally, we provide qualitative and quantitative results of the 6-DoF pose estimation algorithm. To quantitatively evaluate the pose estimator performance, we create a semantic keypoints dataset with the method described in Section III-A (the dataset contains 19265 frames of which 13487 are used for training and 5778 for testing). We train the stacked hourglass network, and then estimate the poses of the drones in the testing dataset. Table II summarizes this experiment setup and results. We compare the estimated poses with the ground truth pose, available from the Vicon motion capture system. On the

	Rotation error $[^\circ]$	**Translation error** $[m]$
Mean	1.1263	0.0190
STD	1.4504	0.0166

TABLE II: Summary of the 6-DoF pose estimator performance experiment. The dataset contains 5778 testing samples.

testing dataset, we measure the translation error as well as the rotation error among $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ using the geodesic distance

$$e_{\boldsymbol{t}} = ||\boldsymbol{t}_1 - \boldsymbol{t}_2||_2,\ e_{\boldsymbol{R}} = \frac{||\log_{SO(3)}\left(\boldsymbol{R}_1^\top \boldsymbol{R}_2\right)||_F}{\sqrt{2}}. \quad (24)$$

The results show a centimeter-level precision for position estimation and 1–2 degrees for the rotational part.

V. Conclusion

In this work, we presented an approach to detect, localize, and track a set of drones from a moving human's perspective using a headset with an embedded camera and IMU. The proposed approach works in real time, is capable of tracking the robots in space and time, and can also be leveraged in other scenarios that do not have external positioning systems, such as swarms of terrestrial robots or adapted to solve humans' pose estimation problems. It can be potentially employed on any camera/IMU system, including the one available on small drones for formation control. The presented framework incorporates a 6–DoF drone pose estimator based on semantic keypoints prediction and pose optimization. We also presented an automated approach to create labeled datasets for the training of the object detector and of the semantic keypoints neural network leveraging the motion capture system data.

Our results show the effectiveness of the proposed approach and can be employed in multiple domains including virtual and augmented reality, human–robot interaction, and multi-robots formation control.

Future work will investigate how to extend the approach to reach an association consensus among the tracked objects in the different camera/IMU systems. We would also like to study how to leverage the tracking data across frames to further increase the accuracy of the pose estimation network and speed up its convergence time or reduce the network size.

References

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[2] S. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A Survey on Aerial Swarm Robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855.

[3] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.

[4] K. Elia, L. Antonio, R. René, M. Matthias, K. Vladlen, and S. Davide, "Deep drone acrobatics," *RSS: Robotics, Science, and Systems*, 2020.

[5] M. Macchini, F. Schiano, and D. Floreano, "Personalized telerobotics by fast machine learning of body-machine interfaces," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 179–186, 2019.

[6] J. Bruce, J. Perron, and R. Vaughan, "Ready—Aim—Fly! Hands-Free Face-Based HRI for 3D Trajectory Control of UAVs," in *14th Conference on Computer and Robot Vision*, 2017.

[7] J. Nagi, A. Giusti, G. A. Di Caro, and L. M. Gambardella, "Human control of uavs using face pose estimates and hand gestures," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2014, pp. 252–253.

[8] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 2018, pp. 316–324.

[9] C. Reardon, K. Lee, and J. Fink, "Come see this! augmented reality to enable human-robot cooperative search," in *IEEE Symposium on Safety, Security, and Rescue Robotics*, 2018.

[10] L. Yuan, C. Reardon, G. Warnell, and G. Loianno, "Human gaze-driven spatial tasking of an autonomous mav," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1343–1350, 2019.

[11] M. Aggravi, C. Pacchierotti, and P. R. Giordano, "Connectivity-maintenance teleoperation of a uav fleet with wearable haptic feedback," *IEEE Transactions on Automation Science and Engineering*, 2020.

[12] J. Nagi, A. Giusti, L. M. Gambardella, and G. A. Di Caro, "Human-swarm interaction using spatial gestures," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 3834–3841.

[13] E. Tsykunov, R. Agishev, R. Ibrahimov, L. Labazanova, A. Tleugazy, and D. Tsetserukou, "Swarmtouch: Guiding a swarm of micro-quadrotors with impedance control using a wearable tactile interface," *IEEE Transactions on Haptics*, vol. 12, no. 3, pp. 363–374, 2019.

[14] F. Schiano and P. Robuffo Giordano, "Bearing rigidity maintenance for formations of quadrotor UAVs," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1467–1474.

[15] F. Schiano and R. Tron, "The dynamic bearing observability matrix: Nonlinear observability and estimation for multi-agent systems," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3669–3676.

[16] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. Robuffo Giordano, "Decentralized rigidity maintenance control with range measurements for multi-robot systems," *The International Journal of Robotics Research*, vol. 34, no. 1, pp. 105–128, 2015.

[17] R. Tron, J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "A distributed optimization framework for localization and formation control: Applications to vision-based measurements," *IEEE Control Systems Magazine*, vol. 36, no. 4, pp. 22–44, Aug 2016.

[18] D. Dias, R. Ventura, P. Lima, and A. Martinoli, "On-board vision-based 3d relative localization system for multiple quadrotors," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1181–1187.

[19] V. Walter, N. Staub, A. Franchi, and M. Saska, "Uvdar system for visual relative localization with application to leader–follower formations of multirotor uavs," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2637–2644, 2019.

[20] E. Montijano, E. Cristofalo, D. Zhou, M. Schwager, and C. Sagüés, "Vision-based distributed formation control without an external positioning system," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 339–351, 2016.

[21] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual slam," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2466–2473.

[22] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," in *arXiv preprint arXiv:1804.02767*, 2018.

[23] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE journal of Oceanic Engineering*, vol. 8, no. 3, pp. 173–184, 1983.

[24] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, 1980, pp. 807–812.

[25] K. Mohta, "State estimation, control, and planning for a quadrotor team," *Publicly Accessible Penn Dissertations. 3360*, 2018.

[26] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-dof object pose from semantic keypoints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2011–2018.

[27] M. Vrba and M. Saska, "Marker-less micro aerial vehicle detection and localization using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2459–2466, 2020.

[28] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4523–4530, 2019.

[29] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 483–499.

[30] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," *ACM transactions on graphics (TOG)*, vol. 27, no. 3, pp. 1–10, 2008.