Robot Identification and Localization with Pointing Gestures

Boris Gromov, Luca M. Gambardella, Alessandro Giusti

***Abstract*— We propose a novel approach to establish the relative pose of a mobile robot with respect to an operator that wants to interact with it; we focus on scenarios in which the robot is in the same environment as the operator, and is visible to them. The approach is based on comparing the trajectory of the robot, which is known in the robot's odometry frame, to the motion of the arm of the operator, who, for a short time, keeps pointing at the robot they want to interact with. In multi-robot scenarios, the same approach can be used to simultaneously identify which robot the operator wants to interact with. The main advantage over alternatives is that our system only relies on the robot's odometry, on a wearable inertial measurement unit (IMU), and, crucially, on the operator's own perception. We experimentally show the feasibility of our approach using real-world robots.**

Videos, Datasets and Code

Video, datasets and code to reproduce our results are available at: http://people.idsia.ch/~gromov/motion-relloc

I. Introduction

Pointing gestures, also known as deictic gestures, are a simple, intuitive and frequently used device when humans in close proximity want to communicate directions, positions or objects to each other; pointing gestures can also be an intuitive way to interact with mobile robots that share space with an operator, e.g. indicating a direction for a robot to explore, an object to inspect, a spot for a drone to land. One practical implementation strategy to realize this vision relies on a wearable device on the operator (such as a smartwatch) that is networked with the robot and includes an IMU to sense the orientation of the arm: then, one can robustly reconstruct a 3D ray in the operator's own local reference frame, on which the pointed location or object lies. In order to use it for robot interaction, one needs to express such ray in terms of the robot's own reference frame. This implies that the robot and the operator should be localized with respect to each other, i.e. their relative pose should be known.

The **main contribution** of this paper is a novel, simple and robust approach to estimate the relative 3D pose of the robot with respect to the operator, which mainly relies on a wearable IMU and on one powerful sensor that comes for free: the operator's own perception.

In the first phase, the operator points at the robot they want to interact with, in order to "select" it. The act of pointing at something (or an explicit input such as a button press or voice command) triggers the beginning of the second phase, which is marked by a clear feedback (e.g. the operator's bracelet vibrates or emits a sound, the robot lights up with a specific color): in this phase, the robot follows some trajectory, while the operator moves his arm in order to keep pointing at it. After a few seconds, the system is able to estimate the relative localization between the operator and the robot by comparing the operator's arm movements (known in the operator's reference frame) and the corresponding motion of the robot (estimated in the robot's odometry frame). Another feedback marks the end of the second phase.

This paper focuses on the process described above. Once the relative localization is established, the operator's position (and the ray corresponding to any further pointing gesture) is known in the robot's frame. Then, the interaction continues in a scenario-dependent way. For example, the robot may turn or move towards the operator in order to receive further commands; moreover, any further pointing actions by the operator can represent a spatial input to the robot (e.g. a path to be followed, a direction to be explored, or a goal to reach). We previously demonstrated [1] that pointing gestures can be used, even by untrained operators, to intuitively and efficiently guide a drone to land on a precise spot; in that work, however, relative localization was relying on the strong assumption that the operator initiated the interaction while standing exactly behind the drone, and also required that the drone was flying at a known height with respect to the operator.

The approach we propose here, instead, does not rely on any such assumption, but still allows one to optionally incorporate known constraints, for example: 1) in most cases it can be safely assumed that the z-axis of the operator's and robot's reference frames are parallel, because IMUs can accurately determine the direction of gravity; 2) if both the operator and the robot are equipped with a reliable magnetometer, the relative headings may be known at least with some precision; 3) if we assume a flat floor and a ground robot (or a flying robot with an accurate height sensor), the vertical displacement between the two frames can be known in advance. In our experiments, we only rely on the first of these three constraints.

Our approach also handles multi-robot systems: in this context, one additional problem is to determine which robot the user wants to interact with. In case multiple robots are in range, phase two is triggered for all robots simultaneously; crucially, each robot now follows a trajectory with a *different* shape; the system can then simultaneously determine which robot the operator was pointing to, and its relative pose with respect to the operator.

We first discuss related work (Section II), then formalize

(*) All authors are with the Dalle Molle Institute for Artificial Intelligence (IDSIA USI-SUPSI), Lugano, Switzerland. Email: {boris,luca,alessandrog}@idsia.ch.

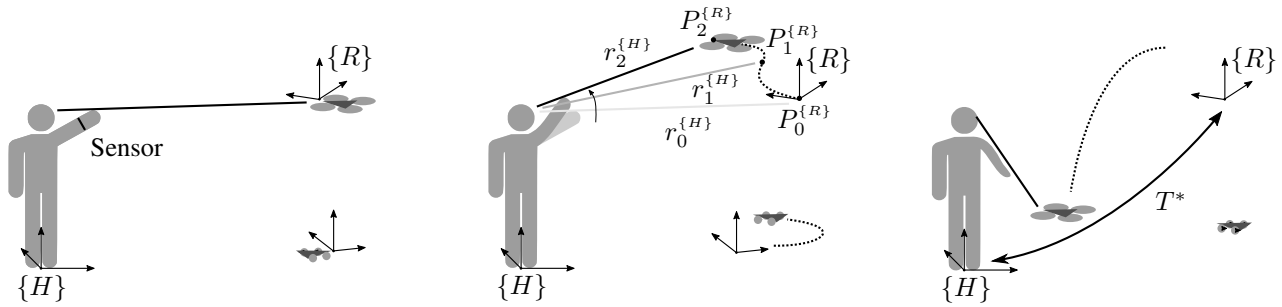

Fig. 1: *Left*, Phase 1: the operator points at the robot they want to interact with; an explicit (button press) or implicit (pointing gesture detection) event triggers the beginning of the interaction. *Center*, Phase 2: all nearby robots start moving along different paths, and the operator keeps pointing at the target robot; the system acquires a set of pairs each composed by: a pointing ray r in the operator's frame of reference $\{H\}$; a point P in the robot's fixed odometry frame $\{R\}$. *Right*: after a few seconds the system has identified the target robot and reconstructed the transformation T^* linking $\{H\}$ and $\{R\}$: pointing rays can be known in the robot's frame, and the interaction can continue in an application-dependent way. This paper focuses on phase 2.

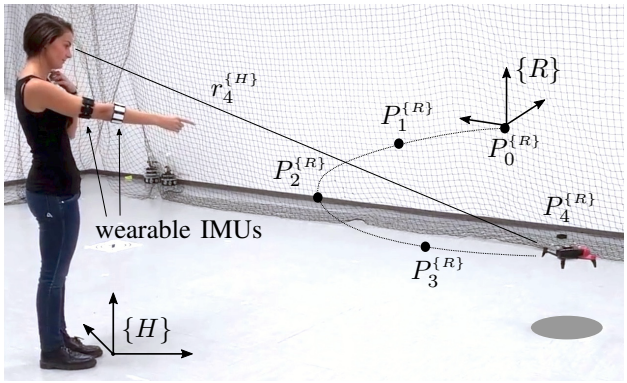

Fig. 2: Implementation of the proposed method in real-world experiment.

the problem and our solution (Sections III & IV); in Section V we describe our experimental setup in detail and report extensive quantitative and qualitative results (Section VI) that demonstrate the effectiveness of the approach on different types of robots and under different conditions. A discussion on future research topics concludes the paper.

II. Related Work

Our work consider two major topics in robotics: localization and human-robot interaction (HRI) using pointing gestures.

A. Localization

Localization is an essential part of any robotic system: it is used for relating a robot pose to a working environment, objects in it or other agents (including humans). The localization methods can be grouped into direct and indirect methods.

1) Direct methods: Direct methods assume that a position of one agent is estimated directly with respect to another agent, using, e.g., triangulation or multilateration of a radio [2], optical, or sound signal [3].

These also include vision-based methods, such as localizing using passive [4, 5] and active [6, 7] markers, known geometry or other visual features of the agent [8]. For example, when it is a human to be localized, features like natural skin color [9], face [10], or even legs [11] can be used.

2) Indirect methods: These methods assume the two agents are localized with respect to a common reference frame and thus a coordinate transformation can also be recovered between the agents.

A common reference can be established using geographical coordinates using global positioning system (GPS) [12], or using techniques that are suitable for indoor use, such as optical motion capture systems like Optitrack, or ultra-wideband (UWB) localization systems [2, 13].

An alternative approach is a co-localization of the agents [14] that individually perform simultaneous localization and mapping (SLAM) of the same environment. By finding correspondences on the maps it is possible to estimate a coordinate transformations between the agents.

The method we propose in this work falls into the category of direct methods, however none of the approaches in that category use inertial sensors for localization. In the indirect methods inertial sensors are used in conjunction with vision sensors to find better estimates of the egomotion of the system for the SLAM task [15].

B. Pointing gestures

Pointing gestures are an innate [16] and effective device that humans use all the time. Thus, they are of a particular interest to robotics community: they allow the human user to intuitively communicate locations and other spatial notions to the robots. The typical tasks to be solved in robotics with pointing gestures are: pick-and-place [17], object and area labeling [18], teaching by demonstration [19], point-to-goal [20] and assessment of the joint attention [21].

To the best of our knowledge the proposed method is the first to use pointing gestures for localization.

C. BB-8

The commercial toy robot BB-8 by Sphero [22] uses a wearable bracelet-like interface equipped with an IMU to provide velocity control: the user can perform a push-like gesture with their arm along a given direction, and the robot moves correspondingly. This requires to first calibrate the relative headings of the wearable bracelet and the robot, i.e. to fix the relative orientation of the human and robot frame. When the user initiates this calibration procedure, an LED lights up on a LED ring on the robot circumference: at this point, the user can control which LED is illuminated on the ring by twisting their wrist. Once the user aligns the illuminated LED with their direction, they press a button to complete the calibration procedure. Now, the heading of the user arm as measured by the IMU can be converted in the robot's frame. This approach is similar to ours in that it exploits the user's perception in order to constrain the robot's frame with respect to the operator's. On the other hand, our approach also determines the relative displacement of the two frames in addition to the relative heading, which enables us to provide a rich position-control interface that would otherwise be impossible (see Section VI-B); moreover, our approach relies on the robot's own motion ability, and can be adopted even when the robot's orientation can not be clearly recognized (e.g. a drone high in the sky).

III. Model

Let us define a reference frame of the operator $\{H\}$, a reference frame of the robot $\{R\}$, and the concept of pointing ray r.

The reference frame $\{H\}$ is located at the operator feet with the x-axis pointing forward, y-axis to the left, and z-axis pointing up. The reference frame $\{R\}$ is an arbitrary fixed reference frame in which the robot reports its position; in practice, it is useful to assume it to be the robot odometry frame, however our model does not require it.

The pointing ray r is a 3D half-line on which the point that the human intends to indicate lies. Estimating r from sensing data is a challenging task that involves human perception and cognition processes, and depends on each individual. This topic has been extensively studied in psychology research [23, 24] which suggests two main models: 1) the *shoulder-wrist* model assumes that the point the human wants to indicate lies along the 3D line defined by the long axis of the pointing arm; 2) the *head-finger* model assumes that the point lies along the line connecting the dominant eye and the tip of the pointing finger. The first model relies solely on proprioceptive feedback and emerges when visual feedback is not available, e.g. user is blindfolded or operates in virtual reality environment without appropriate visual cues. The second model, in contrast, corresponds to normal conditions. The choice of a particular model in robotic applications mainly depends on the technology available for sensing the user's posture and on the task.

Our goal is to estimate the pose (position and orientation) of the robot's frame $\{R\}$ in the reference frame of the operator $\{H\}$: the operator points at the moving robot and keeps following it with a pointing gesture for a short period of time τ; we collect the pointing rays $r_i^{\{H\}}$ in the reference frame $\{H\}$ and corresponding robot's positions $P_i^{\{R\}}$ in the frame $\{R\}$, after which a coordinate transformation T^* between the two frames is estimated using an optimization procedure.

A. Formal definition

Given a finite set $\mathcal{R}$ of N pointing rays $r_i^{\{H\}}$, defined in the frame of reference of the operator $\{H\}$:

$$\mathcal{R} = \{r_1^{\{H\}}, \dots, r_N^{\{H\}}\},$$

for each $r_i^{\{H\}}$ we consider the corresponding robot position $P_i^{\{R\}}$ defined in the frame of reference of the robot $\{R\}$, and thus define a set of pairs $\mathcal{C}$:

$$\mathcal{C} = \{\left(r_1^{\{H\}}, P_1^{\{R\}}\right), \dots, \left(r_N^{\{H\}}, P_N^{\{R\}}\right)\},$$

We expect that the points $P_i^{\{R\}}$ lay close to their corresponding rays $r_i^{\{H\}}$.

For a given estimate T of the transformation, we can convert the robot positions $P_i^{\{R\}}$ defined in the robot frame into the operator frame, i.e. $P_i^{\{H\}} = TP_i^{\{R\}}$. Using these points we define a new ray $q_i^{\{H\}}$ that shares the origin with the ray $r_i^{\{H\}}$, but passes through the point $P_i^{\{H\}}$.

Now, we can define the error function θ for a set of pairs $\mathcal{C}$:

$$\theta\left(T, \mathcal{C}\right) = \frac{1}{N} \sum_{i=1}^{N} \angle(r_i^{\{H\}}, q_i^{\{H\}}) \tag{1}$$

where $\angle(\cdots) \in [0; \pi]$ represents the unsigned angle between the directions of two rays. The error function $\theta\left(T, \mathcal{C}\right)$ is therefore 0 iff all points lie on the respective ray, and > 0 otherwise.

We search for the coordinate frame transformation T^* between the operator frame $\{H\}$ and the robot frame $\{R\}$ that minimizes the error function, i.e. that minimizes the average unsigned angle between all the pairs of vectors $r_i^{\{H\}}$ and $q_i^{\{H\}}$.

$$T^* = \arg\min_T \theta(T, \mathcal{C}) \tag{2}$$

The residual error $\theta^* = \min \theta(T^*, \mathcal{C})$ indicates how well the transformed robot positions fit the corresponding rays.

B. Interaction length

The number of data points N in this model is defined by the interaction time τ (in seconds) and the data acquisition frequency f (in Hz), such that $N = \tau \cdot f$. We assume all the sensors of the system are synchronized to this common frequency. In Section VI, we test the influence of τ on the resulting error.

C. Pointing model

In our system, the pointing rays r in coordinate frame $\{H\}$ are found using orientation readings from wearable IMUs. They can be defined in various ways depending on the chosen human perception (pointing) model. The most popular models in robotics are [25, 26, 9, 27]: *head-finger*, *upper arm*, and *forearm*. These models define where does the ray r originate from and which other point does it go through. A pointing ray of the *head-finger* model originates at a centroid of the head and passes a point at the fingertip; respectively, the *upper arm* pointing model defines a ray with the origin at the shoulder and passing through a point at the elbow; the *forearm* model defines a ray with the origin at the elbow and passing via the wrist joint. In this work, we employ the head-finger model.

D. Constraints on the transformation

We express the transformation T^* as a composition of translation and rotation, where the translation is defined as a three-dimensional vector $t = [t_x, t_y, t_z]$ and the rotation as $\gamma = [\gamma_x, \gamma_y, \gamma_z]$. We further simplify the model by ignoring rotations around x- (roll) and y-axis (pitch). This is a fair assumption for our application since the z-axes of the operator and the robot coincide and correspond to the opposite direction of the gravity vector estimated by their IMUs.

The optimization problem is now reduced to that of finding a four-dimensional vector:

$$\rho = [t_x, t_y, t_z, \gamma_z] \tag{3}$$

E. Application to multiple robots

In case multiple robots are in the scene, one needs to simultaneously reconstruct the pose transformation and identify which robot the operator is pointing at. If all robots follow a different trajectory and we assume the user is pointing at one of them, this is easily implemented by solving the minimization problem separately for each robot, and then identifying the target robot as the one which yields the lowest residual error. A schematic representation of this process is presented in Figure 1.

F. Relation with extrinsic camera calibration and the perspective-n-points (PnP) problem

The problem of reconstructing the relative pose of the robot with respect to the user resembles the two well-known problems in computer vision that are closely related to each other: reconstructing the pose of an intrinsically-calibrated [28] camera which observes a known calibration pattern (extrinsic camera calibration), and; reconstructing the pose of a known object [29] when observing the image coordinates of some of its points (perspective-n-points). In both cases, image points can be backprojected as viewing rays in the camera's reference frame (analogous to $\{H\}$); the 3D points are known in the object frame (analogous to $\{R\}$), and the goal is to reconstruct the transformation between the two frames.

In this analogy, the intrinsic calibration of the camera relates the measurements (i.e. the image points) to the viewing rays in the camera's reference frame; similarly, our pointing model relates the measurements of our sensors to pointing rays in frame $\{H\}$. In computer vision, the 3D points are known in the frame of the object; in our model, instead, the 3D points are built from the robot's motion, in frame $\{R\}$. When solving these problems in computer vision, one often minimizes the average reprojection error over all points, which is analogous to the way our error function is defined.

IV. Implementation

Similar to our previous work [1], we equipped the operator with a pair of wearable Myo Armband sensors that include the 9-axis inertial measurement units (IMUs). The sensors are placed on the upper arm and the forearm; the data is streamed to a host PC via Bluetooth LE link with the 50 Hz rate, where it is resampled at 30 Hz rate and synchronized with other data. Internally, each sensor fuses the data from its accelerometer, gyroscope and magnetometer into a robust orientation estimate that is provided to the user as a quaternion.

Using predefined parameters of the operator body, i.e. the shoulder and eyes height above the ground and the length of the arm (shoulder to index fingertip), we calculate the static positions of the head and the shoulder with respect to the operator coordinate frame $\{H\}$ placed at their feet. We also predefine relative transformations between the joints of human arm to later determine dynamic position of the index finger.

We perform a model-constrained arm pose reconstruction using just a single sensor on the forearm and thus require the user to point with a straight arm. The data from both sensors serve as an input to the pointing detector [30] that is used for triggering the start of the robot motion; the exact method used for triggering depends on the application and is out of the scope of the present paper.

The static and dynamic points described above allow us to define pointing rays r for various human perception models, described in Section III-C. For this work we chose the *head-finger* model.

We assume that the robot and the operator are networked and the robot is able to track its own position with a reasonable accuracy in some fixed reference frame $\{R\}$, e.g. its odometry frame.

The proposed system requires no fixed infrastructure and can be used both indoors and outdoors.

First, we acquire 3D-orientation data from a pair of inertial sensors, and calculate arm poses and their respective pointing rays $r_i^{\{H\}}$ in the frame of reference of the operator. At the same time, we acquire robot positions $P_i^{\{R\}}$ in its reference frame and synchronize them with corresponding pointing rays in order to build the set $\mathcal{C} = \{(r_i^{\{H\}}, P_i^{\{R\}})\}$. In practice, the set is implemented as a buffer of size N. It is starting to fill in when the interaction sequence is triggered

by the user, otherwise the data is dropped. Once the buffer is full we pass the set to the optimization procedure.

The nonlinear optimization problem (Eq. 2) is solved with the quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shanno [31] as implemented in the `optimize.minimize` function from the SciPy library [32], with default parameters. As the initial guess, we use $\rho_0 = [0, 0, 0, 0]$, i.e., we consider the robot frame coinciding with the user frame.

We base our implementation on the Robot Operating System (ROS) [33] framework, where we synchronize and process all streams of data.

The transformation retrieved with the optimization procedure is then published to the ROS *tf*-tree.

V. Experimental Setup

We implemented our system in an indoor flying arena of approximately 7×7 m size. The arena is equipped with the Optitrack motion capture (MOCAP) system, that is capable of tracking objects in 3D space with up to 200 Hz rate.

As the target robotic platform we used a commercial drone Parrot Bebop 2. We equipped it and the operator with markers to acquire the ground truth for our experiments.

A. Data collection

We acquired data in several sessions. In each session, the drone repeatedly followed a predetermined closed trajectory without interruption; in each session, the user was standing at a predefined position and was instructed to continuously point at the drone with a straight arm. To start the session, the operator had to point at the drone and press and hold a button on a joypad. We recorded five such sessions, three of which were performed on a triangular trajectory and two on a circular one. On average each full session lasted 50 s. A single loop of the circular and triangular trajectories took approximately 12 s and 7 s, respectively.

Each session resulted in: 1) a set $\mathcal{C}$ of pointing rays $r_i^{\{H\}}$ obtained from IMU data and corresponding robot positions $P_i^{\{R\}}$ recorded by the tracking system; 2) the set of ground truth operator positions $\mathcal{U} = \{U_1^{\text{gt}}, \dots, U_N^{\text{gt}}\}$ recorded by the tracking system. Samples of a single loop of the trajectories, and the operator positions in each session are depicted in Figure 3.

We then analyze this data to evaluate the performance of the algorithm under different conditions and settings.

B. Odometry error model

Our approach relies on the robot's odometry to estimate the trajectory in the robot's frame. For our experiments, we acquired the ground truth trajectory $P_1^{\text{gt}}, \cdots, P_N^{\text{gt}}$ using the motion tracking system, then we perturbed it in order to simulate the trajectory that the robot's odometry would yield in different operating conditions.

In particular, we consider a simple synthetic model for the errors of a visual odometry (VO) pipeline which we pessimistically assume to never perform relocalization: the estimated trajectory therefore accumulates errors and drifts

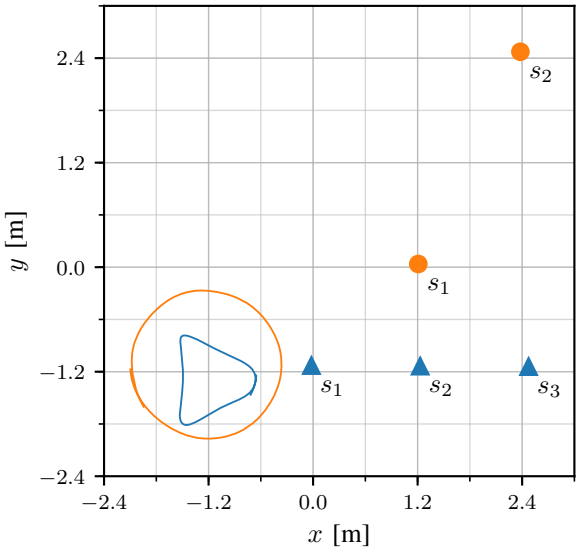

Fig. 3: Experimental setup for data collection (ground truth): two trajectories (circular and triangular) are represented by thin orange and blue lines, the operator positions for each of the five sessions are represented by the circular and triangular markers. Labels identify the respective sessions in Figure 5.

away from the real one. This is implemented as follows. We consider the sequence of ground truth positions sampled at 30 Hz; we define $P_1^{\text{vo}} = \mathbf{0}$ (i.e. the origin of the robot's odometry frame), then iteratively compute $P_i^{\text{vo}} = P_{i-1}^{\text{vo}} + (P_i^{\text{gt}} - P_{i-1}^{\text{gt}}) + \epsilon$, with $\epsilon \sim \mathcal{N}_3(\mathbf{0}, \text{diag}(\sigma, \sigma, \sigma))$. We test four scenarios: $\sigma = \{0, 0.001, 0.005, 0.015\}$, corresponding respectively to ideal, good, bad, and terrible odometry performance.

The effects of such transformation on a sample trajectory are visualized in Figure 4; quantitative data averaged on a large amount of simulations are reported in Table I. In this table, for example, we observe that, on average over all experiments, a 10-second trajectory is 5.19 m long, and has an extent (farthest distance between any two points) of 1.38 m. The good ($\sigma = 0.001$) and bad ($\sigma = 0.005$) VO models yield a maximum deviation from the ground truth trajectory of 0.03 and 0.17 m respectively, which correspond to 2% and 12% of the trajectory extent.

VI. Results

A. Quantitative evaluation

1) Experiment description: Using the data acquired as described above, we run the following experiments.

One experiment is defined by three parameters: session, trajectory duration τ, VO noise σ. To run one experiment, we extract a random segment of the trajectory from the defined session, with the required duration τ; then, we corrupt the measured robot trajectory using the VO error model described in Section V-B with the specified VO noise value σ. This yields a set of ray-point pairs, whose cardinality depends on the duration of the trajectory. The nonlinear

Trajectory duration [s]	Trajectory length [m]	Trajectory extent [m]	Mean VO error [m] for $\sigma =$				Max VO error [m] for $\sigma =$			
			0.0	0.001	0.005	0.015	0.0	0.001	0.005	0.015
0.5	0.25	0.24	0.0	0.00	0.02	0.07	0.0	0.01	0.04	0.11
1.0	0.51	0.48	0.0	0.01	0.03	0.08	0.0	0.01	0.05	0.14
2.0	1.03	0.86	0.0	0.01	0.04	0.11	0.0	0.01	0.07	0.20
3.0	1.54	1.11	0.0	0.01	0.05	0.15	0.0	0.02	0.09	0.26
5.0	2.58	1.33	0.0	0.01	0.06	0.19	0.0	0.02	0.11	0.33
10.0	5.19	1.38	0.0	0.02	0.10	0.29	0.0	0.03	0.17	0.51
20.0	10.36	1.41	0.0	0.02	0.12	0.37	0.0	0.05	0.23	0.70

TABLE I: Trajectory estimation errors due to the visual odometry model. For each trajectory duration (row), we report the average over all experiments of the following measures (all in meters): the ground truth trajectory length and extent (i.e. the distance between its two farthest points); the mean and maximum error due to visual odometry for $\sigma = \{0, 0.001, 0.005, 0.015\}$.

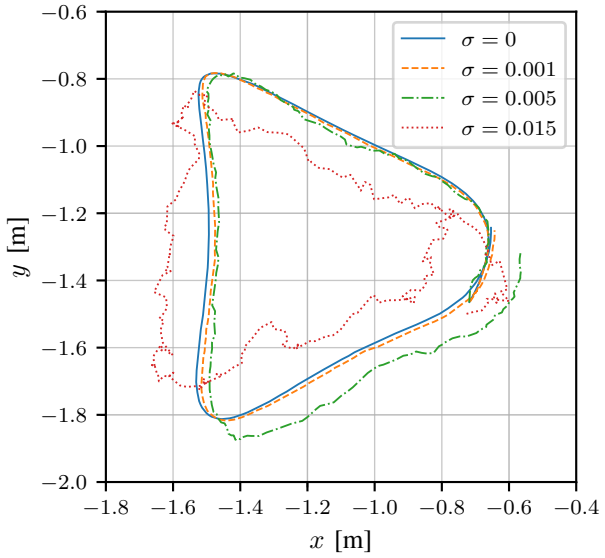

Fig. 4: Simulated odometry estimates of the drone trajectories: (blue/solid) *perfect*, i.e. ground truth; (orange/dashed) *good*; (green/dot-dashed) *bad*, (red/dotted) *terrible*.

minimization procedure (Eq. 2) is then executed using such input data, which returns an estimated transform T^*.

2) Error metric: For a given experiment, we are interested in measuring the error of the estimated transform T^* with respect to the true transform between the human and robot frames.

In order to quantify this error, we consider the location of the top of the operator's head. In the human frame, this point has coordinates $(0, 0, 1.83)$ (tailored to the user), since the human frame $\{H\}$ is defined to lie at the feet of the operator. In the robot frame $\{R\}$, the ground truth coordinates of the same point are measured for each session by means of the motion tracking system, since the operator wears a hat with a marker. If the reconstructed transformation T^* was exact, such ground truth point would be transformed to $(0, 0, 1.83)$ when expressed in the human frame. In the following, we report as an error metric the horizontal component of the distance between the transformed position of the top of the head, and the point $(0, 0, 1.83)$ in the human frame.

This error metric has an intuitive interpretation: in fact, it corresponds to the distance from the true operator's position that a robot would reach if, after relative localization, it was tasked to move to the estimated operator position (assuming perfect odometry during such path).

Note that this metric accounts for both the translational and rotational component of the error in the reconstructed transformation.

3) Accuracy results: In Figure 5 we report, for each value of σ (rows) and each scenario (columns), the value of the error metric as a function of the duration of the trajectory τ (x-axis of the plot). For each setting of the three parameters, we repeat the experiment 20 times (replicas), each with a different random sampling of the trajectory from the chosen session, and a random realization of the VO noise. We report the distribution of the error metric in the 20 replicas as a boxplot. The figure therefore summarizes the results of $3 \times 5 \times 7 \times 20 = 2100$ experiments. We observe the following.

Estimation error decreases with time; this is expected as the number of correspondences increases, which limits the impact of measurement noise and temporary inconsistencies in pointing by the operator; most importantly, longer trajectories have a larger extent, meaning that the localization can become more accurate as the cone of pointing rays spans a larger angle.

Estimation error heavily depends on the session: in the triangle-s1 session, where the operator is very close (about 1.2 m, see Figure 3) to the robot, we obtain very accurate estimates (error < 0.25 m) for trajectory durations as short as 1 second, and 0.5 seconds already yield acceptable results. Longer robot–operator distances still yield median errors close to 0.25 m but only after 3 or 5 seconds, depending on the scenario.

The approach is very robust to odometry errors: the error metric is only marginally impacted by a VO noise $\sigma = 0.005$, which yields significant deviations from the true trajectory (Table I). A large VO noise value ($\sigma = 0.015$) still yields a median error below 0.5 meters on all sessions for a 5-second long trajectory. Interestingly, in triangle sessions, which have a smaller extent than circle sessions, increasing the trajectory duration over 5 seconds is detrimental to accuracy, as accumulating odometry errors negatively affect

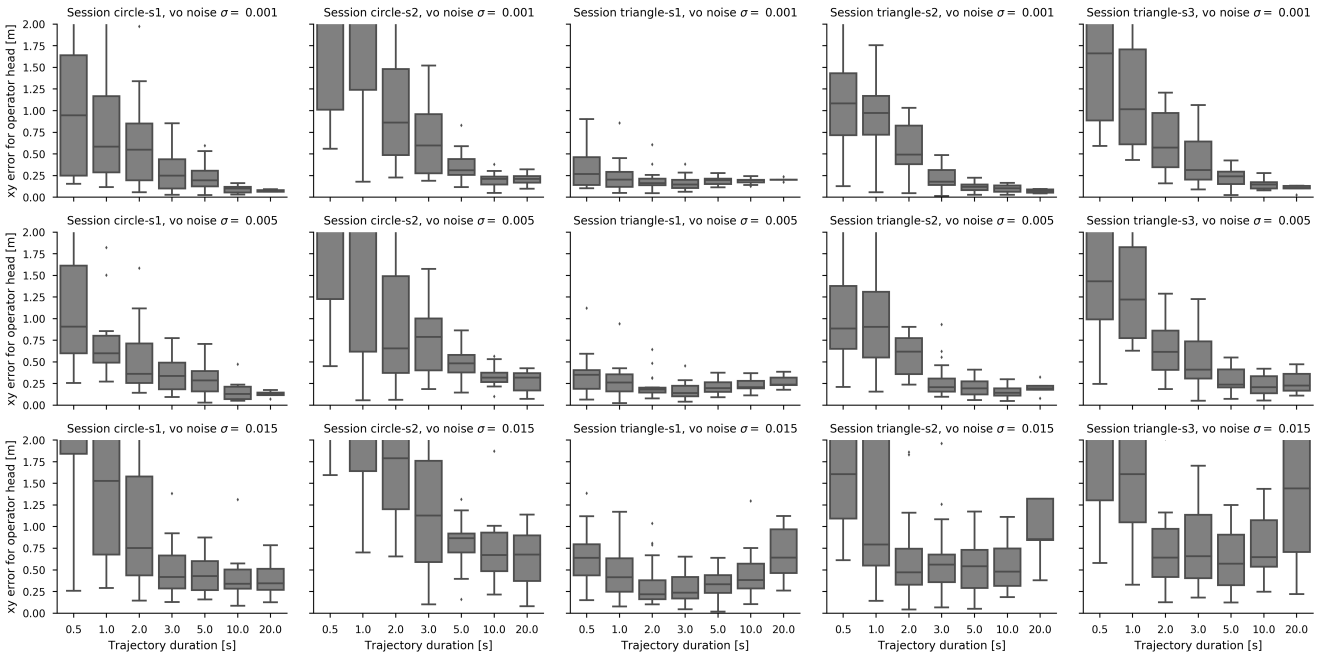

Fig. 5: We report one row for each value of the VO noise $\sigma \in \{0.001, 0.005, 0.015\}$, and one column for each session. Each plot reports the distribution (box) of the error metric (y axis) as a function of the duration of the trajectory (x axis) over 20 replicas. Results for ideal odometry ($\sigma = 0$) are indistinguishable from for plots with $\sigma = 0.001$ (first row) and are therefore not reported.

estimation results.

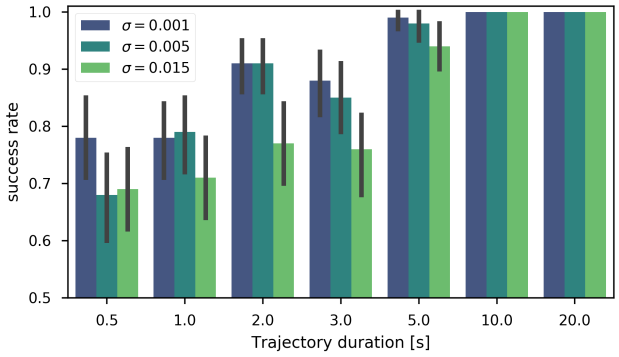

Fig. 6: Success rate in identifying the correct robot among two (see text). 100 replicas per bar. Error bars represent 90% confidence interval.

4) Robot identification results: In Figure 6 we report the success rate in identifying the pointed-to robot among two. For each experiment, we sample a segment of trajectory from one of the five sessions at random (s_{target}), and the corresponding pointing rays; we also sample an equal-length trajectory segment from a *different* session (s_{other}), chosen at random among those with a different shape than s_{target} (circle if s_{target} is triangle, or the other way around), but we ignore the corresponding rays. We now have one set of rays and two sets of points, which are both corrupted with VO error with a given $\sigma \in \{0.001, 0.005, 0.015\}$. The two set of points represent the measured trajectory by two robots: the one pointed to by the operator, and a different one that the operator was ignoring, and that was following a differently-shaped trajectory. We solve the minimization problem associating the one set of rays with each of the two sets of points, and measure the fraction of experiments in which the residual error θ^* is lower for the target trajectory than for the other trajectory. This corresponds to the fraction of experiments for which our approach would have identified the correct robot (the baseline being 50%).

We observe that 5 s are sufficient to discriminate the correct robot in the largest majority of cases, even in case of heavy VO error. In any tested condition, 10 s are sufficient to yield perfect accuracy. It is surprising that, under reasonable odometry performance, the system exceeds 80% accuracy already from 2 s, as such short trajectories are by necessity very simple, almost rectilinear segments.

B. Qualitative evaluation

To show the viability and performance of our method, we set up another real-world experiment.

First, the operator and the drone are localized using the proposed method, then the drone turns to the reconstructed position of the operator, flies towards him, and lands at the the position where the operator was standing, i.e. at the origin of the operator frame $\{H\}$. Figure 2 illustrates the implemented system, and the supplementary video demonstrates many consecutive iterations of the procedure.

Figure 2 and the supplementary video also demonstrate our relative localization approach integrated with our previously-

published system [1], where a drone is guided to land to the precise spot indicated by the user. In this case, after the relative localization procedure is completed, pointing rays are interpreted by the robot in its own odometry frame; the robot is controlled in real time to hover over the intersection of such rays with the ground, and eventually land when the user points to the same spot for a few seconds.

VII. Conclusions

We proposed a novel approach for recovering the relative localization of an operator and a robot, which relies on the innate ability of humans to point to an object they can see (in this case, the robot), and on the robot's odometry; despite the imprecision intrinsic to pointing gestures, the approach yields accurate relative localization from just a few seconds of data, even when the robot's odometry is affected by accumulating errors. In case many robots are in the scene, the system is easily extended to also identify the pointed robot.

Acknowledgments

This work was partially supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics.

References

[1] B. Gromov, L. Gambardella, and A. Giusti, "Video: Landing a drone with pointing gestures," in *HRI '18 Companion: 2018 ACM/IEEE International Conference on Human-Robot Interaction Companion, March 5–8, 2018, Chicago, IL, USA*, 2018.

[2] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-Level Localization with a Single WiFi Access Point," in *NSDI 2016*, 2016, pp. 165–178.

[3] Y. Sasaki, S. Kagami, and H. Mizoguchi, "Multiple sound source mapping for a mobile robot by self-motion triangulation," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 380–385.

[4] M. Fiala, "Designing highly reliable fiducial markers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1317–1324, 2010.

[5] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.

[6] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A monocular pose estimation system based on infrared leds," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 907–913.

[7] B. Gromov, L. M. Gambardella, and G. A. Di Caro, "Wearable multi-modal interface for human multi-robot interaction," *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 240–245, 2016.

[8] A. Fossati, M. Dimitrijevic, V. Lepetit, and P. Fua, "From canonical poses to 3d motion capture using a single camera," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 7, pp. 1165–1181, 2010.

[9] K. Nickel and R. Stiefelhagen, "Pointing Gesture Recognition based on 3D-Tracking of Face , Hands and Head Orientation Categories and Subject Descriptors," *Proceedings of the 5th international conference on Multimodal interfaces*, pp. 140–146, 2003.

[10] J. Nagi, A. Giusti, G. A. Di Caro, and L. M. Gambardella, "Human Control of UAVs using Face Pose Estimates and Hand Gestures," *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction - HRI '14*, no. c, pp. 252–253, 2014.

[11] A. Pesenti Gritti, O. Tarabini, J. Guzzi, G. A. D. Caro, V. Caglioti, L. M. Gambardella, and A. Giusti, "Kinect-based people detection and tracking from small-footprint ground robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4096–4103.

[12] G. Dudek and M. Jenkin, *Inertial Sensors, GPS, and Odometry*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 477–490.

[13] B. Hepp and N. Tobias, "Omni-directional person tracking on a flying robot using occlusion-robust ultra-wideband signals," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 189–194, oct 2016.

[14] P. Schmuck and M. Chli, "Multi-UAV Collaborative Monocular SLAM," *Icra 2017*, pp. 3863–3870, 2017.

[15] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[16] G. Butterworth, *Pointing: Where language, culture, and cognition meet*. Manwah, NJ: Lawrence Erlbaum Associates, 2003, ch. Pointing is the royal road to language for babies, pp. 9–33.

[17] B. Großmann, M. R. Pedersen, J. Klonovs, D. Herzog, L. Nalpantidis, and V. Krüger, "Communicating Unknown Objects to Robots through Pointing Gestures," in *Advances in Autonomous Robotic Systems 15th Annual Conference, TAROS 2014*. Birmingham: Springer, 2014, pp. 209–220.

[18] A. J. B. Trevor, J. G. Rogers, A. Cosgun, and H. I. Christensen, "Interactive object modeling & labeling for service robots," *ACM/IEEE International Conference on Human-Robot Interaction*, p. 421, 2013.

[19] J. Sugiyama and J. Miura, "A wearable visuo-inertial interface for humanoid robot control," in *ACM/IEEE International Conference on Human-Robot Interaction*. IEEE, mar 2013, pp. 235–236.

[20] M. T. Wolf, C. Assad, M. T. Vernacchia, J. Fromm, and H. L. Jethani, "Gesture-based robot control with variable autonomy from the JPL BioSleeve," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1160–1165, 2013.

[21] A. G. Brooks and C. Breazeal, "Working with Robots and Objects: Revisiting Deictic Reference for Achieving Spatial Common Ground," *Gesture*, pp. 297–304, 2006.

[22] "Special Edition BB-8: Battle-Worn Droid with Force Band by Sphero," https://www.sphero.com/starwars/bb8-se, [Online; accessed: 2018-03-01].

[23] J. L. Taylor and D. McCloskey, "Pointing," *Behavioural Brain Research*, vol. 29, no. 1-2, pp. 1–5, jul 1988.

[24] O. Herbort and W. Kunde, "Spatial (mis-) interpretation of pointing gestures to distal spatial referents," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 42, no. 1, pp. 78–89, 2016.

[25] D. Droeschel, J. Stückler, and S. Behnke, "Learning to interpret pointing gestures with a time-of-flight camera," *Proceedings of the 6th international conference on Human-robot interaction - HRI '11*, pp. 481–488, 2011.

[26] A. Cosgun, A. J. B. Trevor, and H. I. Christensen, "Did you Mean this Object?: Detecting Ambiguity in Pointing Gesture Targets," in *HRI'15 Towards a Framework for Joint Action Workshop*, 2015.

[27] S. Mayer, K. Wolf, S. Schneegass, and N. Henze, "Modeling Distant Pointing for Compensating Systematic Displacements," in *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems*, vol. 1, 2015, pp. 4165–4168.

[28] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[29] S. Li, C. Xu, and M. Xie, "A robust o (n) solution to the perspective-n-point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1444–1450, 2012.

[30] D. Broggini, B. Gromov, L. M. Gambardella, and A. Giusti, "Learning to detect pointing gestures from wearable IMUs," in *Proceedings of Thirty-Second AAAI Conference on Artificial Intelligence, February 2-7, 2018, New Orleans, Louisiana, USA*. AAAI Press, 2018.

[31] J. Nocedal and S. Wright, *Quasi-Newton Methods*. New York, NY: Springer New York, 2006, pp. 135–163.

[32] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," http://www.scipy.org/, 2001–, [Online; accessed: 2018-03-01].

[33] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.