Human-Robot Interaction with Pointing Gestures

Intuitive interaction between co-located humans and robots

Doctoral Dissertation submitted to the

Faculty of Informatics of the Università della Svizzera Italiana

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

presented by

Boris Gromov

under the supervision of

Prof. Luca Maria Gambardella and Dr. Alessandro Giusti

March 2020

Dissertation Committee

Prof. Silvia Santini	Università della Svizzera Italiana, Switzerland
Prof. Cesare Pautasso	Università della Svizzera Italiana, Switzerland
Dr. Rachid Alami	LAAS-CNRS, France
Prof. Valeria Villani	University of Modena and Reggio Emilia, Italy

Dissertation accepted on 18 March 2020

[illegible]

Research Advisor

Prof. Luca Maria Gambardella

Alessandro Giusti

Co-Advisor

Dr. Alessandro Giusti

PhD Program Director

Prof. Walter Binder / Prof. Olaf Schenk

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

[illegible]

Boris Gromov
Lugano, 18 March 2020

To my parents

Abstract

Human-robot interaction (HRI) is an active area of research and an essential component for the effective integration of mobile robots in everyday environments. In this PhD work, we studied, designed, implemented, and experimentally validated new efficient interaction modalities between humans and robots that share the same workspace. The core of the work revolves around deictic (pointing) gestures—a skill that humans develop at an early age and use throughout their lives to reference other people, animals, objects, and locations in the surrounding space. To use pointing to control robots, gestures have to be correctly perceived and interpreted by the system. This requires one to model human kinematics and perception, estimate pointed directions and locations using external or wearable sensors, localize robots and humans with respect to each other, and timely provide feedback to let users correct for any inaccuracies in the interaction process.

Our **main contributions** to state of the art lie at the intersection of related topics in psychology, human-robot and human-computer interaction research. In particular, we designed, implemented, and experimentally validated in real-world user studies:

1. a pointing-based relative localization method and its application to robot identification and engagement;
2. an approach for pointing-based control of robots on 2D plane and robots freely moving in 3D space;
3. efficient interaction feedback modalities based on robot motion, lights, and sounds.

Acknowledgements

First and foremost, I want to thank my supervisors Luca Maria Gambardella and Alessandro Giusti, for their trust, patience, and invaluable support through the years of my PhD at IDSIA and Gianni Di Caro for his supervision and help during the first year on this journey.

I am very grateful to my early academic advisors Ivan Ermolov and Yuri Poduraev, who convinced me to step on the path of robotics research, Jee-Hwan Ryu, to boost my research endeavors and carrier, and my friends Carol Tan and Ildar Farhatdinov for their continuous encouragement to stay on this path.

This work would not be possible without precious discussions and help from my colleagues Jérôme Guzzi, Omar Chavez-Garcia, Eduardo Feo Flushing, Gabriele Abbate, Dennis Broggini, Dario Mantegazza, Mirko Nava, and Fabrizio Patuzzo. Above all, I want to thank Jérôme Guzzi for his rigorous feedback and a great help with my experiments. I also thank our secretaries Cinzia Daldini and Daniela Mainini, for their selfless help with everyday issues and for making my stay in Lugano a pleasure.

I especially want to thank my friend Sophia Kvitko to show me the world beyond robotics and to kick-start a much-needed change of life perspective.

This work was partially supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics. I thank all my colleagues within NCCR Robotics with whom I had luck and pleasure to collaborate with all these years.

Finally, I want to thank my dissertation committee, Silvia Santini, Cesare Pautasso, Rachid Alami, and Valeria Villani, for their time and effort to review my work.

Contents

Figures

Tables

Chapter 1

Introduction

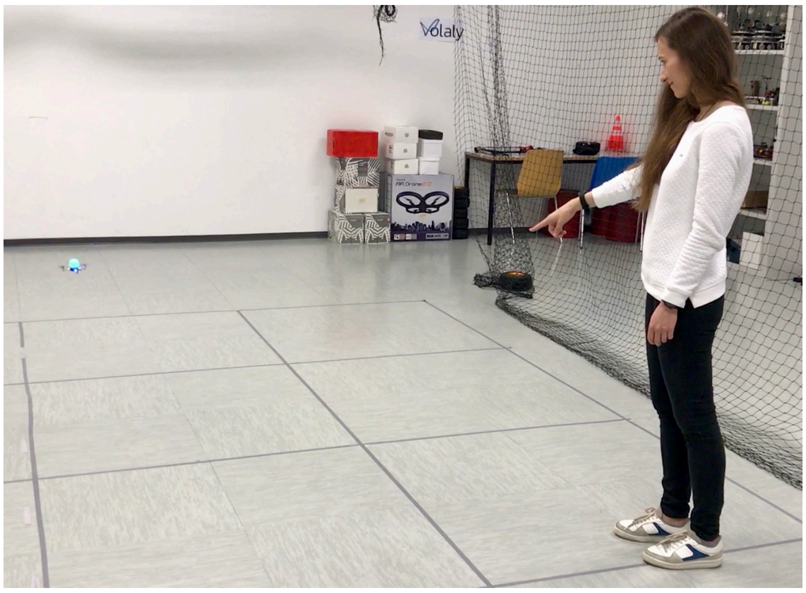


Figure 1.1. The user controls a miniature drone (Bitcraze Crazyflie 2.0) with pointing gestures acquired with a wearable IMU sensor (Mbientlab MetaMotionR+) placed on the user's wrist.

Human-robot interaction (HRI) is a large field that covers many areas of human knowledge. It lies at the intersection of robotics, cognitive sciences, and computer sciences. It involves analysis of human-human interactions, human and robotic perception, design of hardware and software systems.

The interaction between humans involves multiple modalities, such as speech, gestures, facial expressions, gaze, and physical contacts. In human-robot interaction, it is common to differentiate interaction as physical (pHRI) and cognitive (cHRI). An overview of these fields are given, respectively, by Bicchi et al. [2] and Breazeal et al. [6]. pHRI involves physical contacts between the agents, and thus, studies interaction forces and control stability, dominance distribution in multi-agent setups, haptic feedback, etc. A typical example of physical human-robot interaction would be the exoskeletons ('wearable robots') which augment the force of a wearer either to enhance their natural abilities or for rehabilitation purposes. In contrast, cHRI deals with a higher-level interaction, where the agents only exchange information, but not forces. The information in the cognitive interaction case can be conveyed in the form of concepts, e.g. directions, locations, and common objects ('a can of soda'). This PhD work focuses on cHRI and thus will further refer to it simply as HRI.

In the past years, a considerable amount of robotics research has been devoted to gesture-based human-robot interaction. The two types of gestures generally considered are *hand* and *arm* gestures. While hand gestures are usually associated with iconic commands or emblems, pantomimes and sign language, arm gestures are more often associated with pointing or *deictic* gestures [52; 72]. Because pointing requires to move both the arm and the hand (the index finger is extended and the arm is straight), many works that use the term *hand gestures* presume the use of *arm gestures* as well and thus both notions are often used interchangeably.

Depending on the type of interaction and the type of information to be communicated to a robot the gestures can be either static or dynamic, could accompany speech or can be used on their own. For example, periodic dynamic gestures, like arm or hand waving, could be used to attract a robot's attention [53]; pointing gestures could provide a context for the speech commands by linking deictic pronouns, e.g. 'that' and 'there', to a spatial entity, such that a cumbersome utterance like 'Go three meters forward, then five meters to the left', can be simply replaced by 'Go there' and the pointing gesture.

Pointing to entities or positions in an environment is a skill acquired within the first year of human life [9] and an effective device that humans use all the time. Thus, pointing is of particular interest to the robotics community: it allows the human user to intuitively communicate locations and other spatial notions to robots. Some of the tasks that can be efficiently solved with pointing gestures are pick-and-place [33], object and area labeling [79], teaching by demonstration [73], point-to-goal [84], and assessment of the joint attention [8].

In this work, we are looking for efficient ways to interact with robots using pointing gestures. We focus on point-to-goal applications for mobile flying and ground robots: the user points at a location in 3D space and the robot moves there using its on-board intelligence. This interaction pattern suits well many applications: it can be used for transporting goods with cargo robots in warehouses, indicating parking spots for autonomous vehicles, specifying areas of interest for inspection drones, indicating a precise landing spot for a delivery drone, or even for calling a cleaning robot to wipe a spill on the floor.

Research problem

A typical approach to human-robot interaction using gestures would be to equip the robot with a computer vision sensor, e.g. a stereo vision or a structured light camera. In this case, the robot has to actively look for a human in the environment, track them, try to recognize if they want to initiate the interaction, and then wait for commands to follow. Once the command is issued the robot may need to look away from the user to perform the given task, and therefore will not being able to receive new commands anymore.

We follow an alternative approach that does not require the robot's attention. It is based on the use of compact wearable inertial measurement units (IMUs) that provide an accurate estimation of their own orientation with respect to an arbitrary absolute reference frame. This type of sensors is ubiquitous and nowadays present in almost all modern smartphones, smartwatches, and fitness trackers. One or multiple IMUs are placed on the user's arm or held in the hand and provide the direct measurements of the arm's pose. The necessary computations are then performed on a device worn by the user, e.g. a smartphone or a smartwatch, such that the robot receives only high-level commands. The commands are communicated to the robot via a wireless link and do not require a robot's visual attention.

Both sensing approaches have their pros and cons. Although vision-based methods operate within a limited field of view and range, they allow the system to gather reacher information about the user's posture, e.g., they can capture positions of all visible human joints at once without adding new sensors. Moreover, the robot-centric vision systems estimate positions of the joints directly in the robot's coordinate frame, and thus, require no additional localization of the user relative to the robot. On the other hand, robot-centric vision poses an essential problem for gestures aimed directly at the robot or locations very close by: the arm axis gets nearly perpendicular to the image plane and makes it impossible to estimate the pointed direction accurately. This peculiarity makes pointing-based robot selection and continuous position control problematic.

On the contrary, inertia-based sensing does not rely on the direct line of sight and is suitable for a broader range of environmental conditions. It demands less computational power; however, because inertial sensors estimate accurately only rotation, it is necessary to use human body kinematics to calculate joint positions. In turn, it requires the re-localization of the human every time they move away from their initial position. In this work, we propose and demonstrate several efficient ways to deal with these limitations.

Regardless of the approach used to sense gestures, there are several research challenges one needs to address to successfully implement an efficient human-robot interaction system based on pointing.

Human perception. The way people point at objects, places, and other people depends on the context, social and physiological peculiarities of an individual: for example, if a person intends to pinpoint a small object, an object far away, or a single object that cannot be unambiguously identified among other similar objects, then the person is likely to point with a straight arm using the index finger; on the contrary, if the object is big and close by, the person may point with the entire hand and the arm bent at the elbow.

To better understand how pointing gestures can be captured by sensors and how can we infer pointed locations and directions from this data, we study the works in experimental psychology. We then choose a *human pointing model* that would allow us to map a set of input parameters to a pointed location or object. The input parameters could be: a human posture, i.e. relative positions of the human limbs; visual perception peculiarities of an individual, e.g. handedness and eye dominance; cognition; context and environment, e.g. size of a pointed object and a distance to it.

We report a state of the art in psychology and human-robot interaction research in Chapter 2.

Estimation of pointed directions and locations. Because it is not always possible (or practical) to capture all input parameters of human pointing models using a specific technology, e.g. wearable inertial or computer vision sensors, it is necessary to derive appropriate approximated models. This strand of research, therefore, deals with practical implementations of human pointing models in human-robot interaction systems. Once the pointed direction is found we need to relate it to an object or location in the surrounding environment, for example by intersecting a pointing ray with a 3D model of the workspace.

In Chapter 3 we define the human pointing model used in our research—eye-finger model. We formally define its parameters and kinematic relations it is based on. Then, we discuss the ways to find a pointed location in the surrounding environment. Finally, we describe our **main contribution** to this field: *a practical approach to defining 3D target locations for robots freely moving in 3D space*. We validate that approach for a quadrotor control in a user study.

Relative localization. Pointing gestures inherently define locations in the user's local frame which might be unknown to a robot. Hence, it is necessary to define a common reference frame between the user and a robot. In other words, the user has to be localized with respect to the robot. This research direction considers various localization techniques appropriate for human-robot co-localization.

In Chapter 4 we consider several practical approaches to the problem of relative human-robot localization. We formally define the necessary coordinate transformation relations and show how each of the proposed methods allows acquiring the coordinate transformation between a user and a robot. Our **main contribution** is *a motion-based localization method that relies on human perception*: the user points at a moving robot and keeps following it with their arm for a few seconds; the system compares motions of the robot and the user and estimates a relative coordinate transformation between the two.

Robot selection and identification is required whenever an operator has to deal with more than one robot, or more generally when the number of operators is not equal to the number of robots. Even when there are a single operator and a single robot, there is a question: how would the operator start an interaction? We study various practical solutions to these problems that are based on push-buttons, voice control, and gesture recognition.

In Chapter 5 we show various ways to engage a robot and a user. In particular, we demonstrate how our relative localization method described in Chapter 4 can be efficiently used to solve this problem.

Efficient interaction feedback. In the process of human-robot interaction, the user gives commands to a robot and needs to be sure they are received and understood as intended. The interaction feedback plays an important role in this process and allows both humans and robots to immediately correct their actions.

In Chapter 6 we discuss various ways to improve the interaction performance using appropriate feedback.

During this work, we also attempted to come up with *realistic implementations* of interfaces being designed. For this, we searched the most suitable technologies

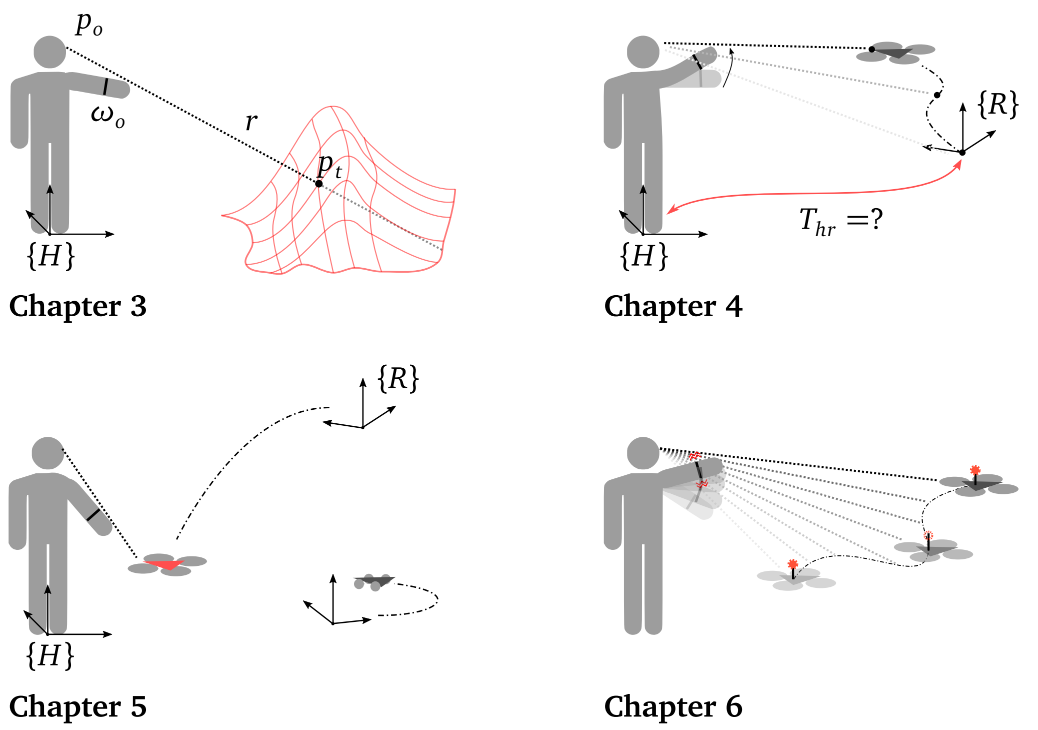


Figure 1.2. Thesis overview: **Chapter 3** describes the human pointing model adopted in this work and estimation of pointed locations in 3D space; **Chapter 4** presents various localization techniques and our method of relative localization from motion; **Chapter 5** discusses robot selection and identification methods, including the one based on our relative localization approach; **Chapter 6** demonstrates various types of feedback: a continuous real-time feedback provided by the robot motion itself, colored lights, and a haptic feedback on a wearable sensor.

that are available on the market today or anticipated to appear in the near future. To validate the usability of the resulting interfaces we conducted qualitative and quantitative user studies.

Finally, in Chapter 7 we draw conclusions, discuss limitations of the proposed methods and ways to improve them.

A graphical overview of our *core research contributions* is presented in Figure 1.2.

Contributions

In this section, the author summarizes his personal contributions to state of the art. These contributions span the Chapters 3–6 where they are described in detail.

Chapter 3: Estimation of Pointed Locations The author proposes a simplified human pointing model that uses a single inertial measurement unit (IMU) to estimate pointed locations both in 2D and 3D space. The main contribution of this chapter is a pragmatic method to specify targets in free 3D space that resolves an inherent ambiguity of pointing—mapping of a two-parameter pointing ray to a three-parameter target location. The author proposes to use a set of well-defined virtual workspace surfaces to constrain the robot's motion. In particular, for improved legibility of the system's state, the author proposes surfaces of two shapes: a vertical cylinder centered at a user's location and a horizontal plane. Each shape is defined to pass through a current robot position: this way, knowing the type of selected workspace shape and seeing the robot, the user can easily visualize the workspace surface and predict the trajectory of the robot. The method is validated in a user study.

Publications The idea to use wearable inertial sensors to estimate pointed locations was first exploited in a workshop paper [25] and further used in a conference paper [24]. In these works, we started with a sophisticated wearable system that included two IMUs: on the upper arm and on the forearm. The pointed locations in these works were estimated as an intersection of the pointing ray (shoulder-finger pointing model) with the ground plane. The main problem of such setups was different drift rates of the two IMUs that resulted in a distorted kinematics chain and the need for re-initialization of the system. The next iteration of the system was simplified to use a single IMU on the upper arm for pointing reconstruction, while the second IMU was used for the detection of pointing events. The author compared the performance of the developed pointing interface against a joystick interface in a drone landing task in [28; 30]. The method to define the robot's locations on a 2D horizontal plane was described in detail in [29], while the method to define positions in free 3D space is presented in [32]. In both cases, to estimate pointed direction we used the eye-finger pointing model.

Chapter 4: Relative Localization The author considers, implements, and validates several practical relative localization methods between a user and robots:

(1) using a global reference frame, (2) using fiducial markers installed on each robot and a camera on the user's forearm, (3) using a predefined heading (pointing at robot from its back), and (4) localization from motion—the main contribution of this chapter to the state of the art. The method (4) requires the user to point at and keep following a moving robot for a few seconds: the system is then able to estimate a relative coordinate transformation between the user and the robot.

Publications We used the method of a global reference frame in our first work (workshop paper) in [25]. We then presented the idea of using fiducial markers in a conference paper in [24], and further studied its accuracy in a workshop paper in [26]. Finally, in [27] we presented our main contribution of this chapter—relative localization from motion.

Chapter 5: Robot Selection and Identification In this chapter the author designed and implemented several robot selection methods. In particular, a combination of speech utterances and pointing gestures were used for the selection of individual and groups of robots, e.g. "You and another 2!". In this case, pointed locations are compared with known coordinates of available robots and used for their identification, while the speech is used as a selection trigger. Because cloud-based speech recognition services that we were using introduced a significant lag into the interaction process and were difficult to scale, we studied alternative selection methods. The author implemented a method based on fiducial markers with unique identifiers that allowed to instantaneously determine a robot the user was pointing at and therefore trigger selection. The main contribution of this chapter to state of the art is a robot identification and selection method based on relative localization from motion approach that is described in Chapter 4: assuming several robots are moving on unique trajectories, the user points at one of them and keeps following it for a few seconds, the system then compares relative localization residual errors and selects a robot whose trajectory yields the lowest error.

Publications We presented the speech-based selection method in the workshop paper [25] and further elaborated it in a conference paper [24]. The latter paper also describes the robot selection method based on fiducial markers. In collaborative work with Broggini et al. [7] we proposed and implemented a method that uses a 1D convolutional neural network to detect pointing events: once the pointing gesture is detected, robot identification can proceed by comparing a pointed-at location with robots' positions. The main contribution of this chapter—robot identification method using the relative localization from motion approach—was presented in a conference paper [27].

Chapter 6: Efficient Interaction Feedback Throughout this PhD work the author aimed at improving the usability of proposed methods by providing adequate feedback to the user. In this chapter, the author describes various types of feedback: voice, lights, vibrations, and robot motion itself. The main contribution of this chapter to the state of the art is a user study on the effects of continuous visual feedback during the pointing process. We show that a continuous control of the robot's position allows the user to timely correct for any inaccuracies induced by a simplified human pointing model and a drift of wearable inertial sensors.

Publications We designed and implemented a system with voice feedback in [24] where it is used for acknowledgment of user's commands. In the same paper, we also used colored lights installed on robots to show when a robot is selected. In [31; 30] we programmed a drone to "jump" in the air to show it has been engaged with the user. In [29] we describe our main contribution to the state of the art—a user study that shows the importance of real time visual feedback on pointing accuracy.

Chapter 2

Literature Review

Since pointing gestures are such a compelling solution to many human-computer and human-robot interaction problems, significant research efforts have been devoted to this topic.

Modern human-robot interaction developed mainly from the teleoperated robotics and the human-computer interaction fields. Arguably, one of the most famous and pioneering works in human-computer interaction is “Put-that-there” by Bolt [5]. In that work the MIT Architecture Machine Group’s “Media room” was staged to perform multi-modal interactions where the user should have used speech commands in conjunction with pointing gestures to manipulate virtual objects on a screen[1]. Bolt argues that this way the virtual interface is implicitly merged into the real world and creates a continuous real-space interaction environment. Moreover, the speech commands supported by pointing gestures can be significantly simplified by replacing the object’s and location’s names with the pronouns like ‘that’ and ‘there’. In this case, Bolt argues, the user even do not have to know what the object is or how it is called.

In the following sections we study the related works from experimental psychology, haptics, human-computer and human-robot interaction fields. We follow the same order of the research topics that we defined in the introduction, in Chapter 1.

2.1 Human perception

Deictic gestures, and in particular pointing to entities or locations in environment, is an innate and a specific to human species skill [9]. Pointing is effective

[1] `https://www.media.mit.edu/speech/videos/Putthatthere_1981_edited_full.mov`

means of communication and interaction that humans use all the time. It has a dialogic nature in that it requires a joint attention of the interacting people on the object of pointing. Butterworth studied the development of pointing comprehension and production in babies and indicates that *canonical pointing*[2] emerges by the end of the first year of life. However, at this age infants' abilities to understand what object is being pointed at are somewhat limited: babies generally fixate on targets only if the latter have salient features, such as when the objects are in motion. By the 18 months babies are able to localize the target without additional cues. Butterworth argues that it become possible thanks to eventual development of joint visual attention when babies are able to extrapolate the adult's head, gaze or pointing arm direction.

The ability to understand pointed locations is even limited in adults. Herbort and Kunde experimentally shown that locations meant by a pointer are different from that are interpreted by an observer [36]: one group of subjects (pointers) was asked to pointing at ticks on a vertical ruler, while the goal of the other group (observers) was to guess these pointed locations. The experiment has shown that most of the observers were indicating positions higher than they were meant by the pointers (Figure 2.1). Herbort and Kunde explain this phenomenon by the difference in geometrical rules that guide the production and interpretation of pointing gestures. While in the pointing production subjects tend to keep pointing finger close to the line of sight—but avoid occluding the target—observers interpret the pointing direction by extrapolating the long axis of the pointing arm.

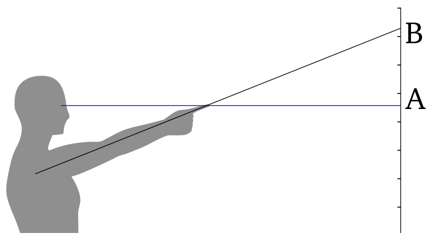


Figure 2.1. Pointing misinterpretation. Subject believes to point at A, while observers interpret she/he points at B. Figure has been adapted from the original work of [36].

Works in psychology research on pointing gestures show that the tip of the index finger of a pointing arm tend to lie close to the line of sight between the human and the object being pointed at when pointing with a straight arm [76; 36]. The human perception model based on this observation is often referred to as the *eye-finger* model. The 'eye' in this model is meant to be a dominant eye[3] of an individual. Khan and Crawford shown that the ocular dominance is not a static concept and depends on the relative position of the eyes with respect to the object they are looking at [43]. The pointing with a straight arm is typical

[2]A gesture with index finger and arm extended, palm down, other fingers curled.

[3]Ocular dominance is the tendency of a brain to prefer visual input from one eye to the other [43].

when the pointed target is far away or when the person is trying to be precise. However, when the targets are relatively close, people tend to bend the arm at the elbow [44]. In this case, the eye-finger model does not work anymore and the pointing is better described by the *elbow-finger* model [44].

In robotic applications of distal pointing we are primarily interested in determining target locations the user wants to indicate; therefore, this work will further focus on pointing production. To understand the extent of the application of pointing gestures we are also interested in intrinsic limitations of human motor control that sets the lower bound of pointing accuracy and the time necessary to perform a pointing gesture.

Motor control of the arm relies on two major senses: *proprioception*[4] and *vision* [22]. One of the early works on pointing production by Taylor and McCloskey shown that vision plays an important role in this process and dominates proprioception [76]. The authors compared the pointing of blindfolded subjects with those using vision and found that in the former case the pointing direction coincides with the shoulder-finger line; however, if the subjects were pointing first using vision and then were blindfolded the influence of visual perception remained for some time and influenced the pointing direction [76]. The difference between visually-monitored pointing and the blindfolded one was also later confirmed by Wnuczko and Kennedy, where the authors shown that the pointing guided by vision results in higher arm elevations as compared to the one guided only by proprioception [83].

Research in neurophysiology shows that vision is likely to be used as a source of information for generating spatial plans for movements towards visual targets within an extrinsic coordinate system, while proprioception is crucial for implementing these kinematic plans by transforming them into control forces of individual muscles [67]. According to the estimations of Sober and Sabes the first stage of motor control planning is dominated by visual information (80%), whereas the second stage relies mostly on proprioception (70%) [70]. These two stages loosely resemble an approach often used in robotics; there, a high-level trajectory planning in Cartesian-space is followed by the trajectory execution in joint-space using inverse dynamics of the robot.

According to Tan et al. the upper bound of human force control bandwidth is about 20 Hz [75], at the same time intrinsic involuntary oscillations (tremor) of the pointing hand have two dominant frequency peaks at 2–4 Hz and 8–12 Hz [54]. Morrison and Keogh analyzed tremor frequencies in a pointing task with

[4] *Proprioception* is the perception of positions and movements of the body segments in relation to each other without the aid of vision, touch, or the organs of equilibrium [4].

and without precise visual feedback and found that the tremor oscillations amplitude significantly increase when the subjects are asked to point at a small target (angular width of 0.42°) using a laser dot [54].

The inherent accuracy of the human arm was studied by Tan et al., they measured the arm joints angle resolution for shoulder, elbow, and wrist, and found that proximal joints are more accurate than distal ones. While the average angle resolution of the shoulder is 0.8°, the average resolution of an index finger (MetaCarpalPhalangeal joint) is much larger and equals to 2.5°. Interestingly, the resolution of the joints increase not just the faster they move, but also if the movements are active (voluntary) as compared to the case when they are moved by external apparatus [40].

There is also a distance limit at which humans with normal eyesight can perceive remote targets, e.g. robots. According to Nancel et al. the angular size of a target should be more than $1' = \left(\frac{1}{60^\circ}\right) \approx 0.017^\circ$ [57]. That means that the maximum distance at which the operator of a typical drone (circumference 30 cm) could see it is 1 km. Taking into account the challenging background of outdoor environments this number probably will be much smaller. As can be seen, the visual acuity is one magnitude better than the resolution of the pointing arm. However, these numbers give the upper bound distance at which a direct line of sight interaction can occur.

2.2 Estimation of pointed locations

The human perception models defined in psychology research do not put any constraints on the input parameters; however, in the engineering applications, not all the input parameters are possible (or practical) to acquire. This research area, therefore, attempts to approximate comprehensive human perception models with simplified models based on application constraints.

The estimation of pointed locations can be split into two sub-tasks: (1) estimation of the *direction* of pointing (usually in the form of a ray), and (2) estimation of the *intersection* of the pointing ray with the environment.

2.2.1 Pointed directions

In HCI and HRI research the two classes of pointing direction estimation methods can be distinguished: *head-rooted* and *arm-rooted*. The head-rooted techniques consider a pointing ray that originates somewhere within the head: from a dominant eye [64], cyclops eye [49], or head centroid [58; 80]. Arm-rooted methods assume the ray originates from a point laying on the pointing arm: at shoulder, elbow [58; 15; 48; 44], wrist [33] or index-finger [48]. The pointing direction is then defined by the second point, sometimes called *direction point*, that lies on the arm. For the head-rooted methods, this point is either a centroid of the hand or a tip of the index finger, while for the arm-rooted techniques the second point can be at the elbow, at the wrist or hand, or at the tip of the index finger.

In HCI these methods are often referred to as *ray casting* or *ray pointing* techniques. The main applications of pointing in HCI field are the interaction with graphical user interfaces (GUIs) on very large displays [5; 41; 12], 3D immersive environments [63], and recently interaction with smart environments [50; 18; 64; 42]. The most popular pointing techniques in HCI are: a *laser pointer* [41] and an *image plane manipulation* [63; 41]. The laser pointer technique refers to any handheld device that can report its orientation which is then used for determining the pointing ray. Among these types of devices are commercially available controllers such as Nintendo Wiimote, HTC Vive Controller, and Oculus Touch. The last two are specifically designed for interaction with virtual reality (VR) environments. The image plane manipulation is an approach to manipulate distant objects as if they are within arm's reach[5]. Essentially this type of technique is directly related to the *eye-finger* pointing model because it also uses one's line of sight to indicate an object.

Based on combinations of the root and direction points, the taxonomy of the pointing models includes head-hand, eye-finger, shoulder-elbow (upper arm), elbow-wrist (forearm), elbow-hand, and index-finger models.

2.2.2 Pointed locations

Once the pointed direction in the form of a ray is found a pointed *location* can be estimated as an intersection of the pointing ray with the environment. While a 3D model of a virtual environment is known in advance, a model of a real environment has to be acquired by some means, for example with computer vision sensors.

In HCI the problem of finding an intersection point is often circumvented by means of *relative pointing* techniques with additional visual feedback (i.e. a cursor on the screen). This type of interface is resembling a computer mouse and does not require the intersection point. Although relative pointing can be quite efficient it requires the system to use motion indexing (or 'clutching') in order to map the range of motion of the physical device to the workspace of

[5]A good illustration: when people pretend on a photo to hold the Sun between their fingers.

the environment. For example, when the computer mouse reaches the border of a tabletop the user has to disengage the mouse by lifting it above the surface and then reposition the device in the physical environment. On the contrary, absolute pointing devices map the entire physical range of motion to the entire workspace; this technique, however, requires the system to know the coordinate transformation between the user (or pointing device) and the screen.

In HRI, instead of a cursor or elements of a GUI, the user needs to control a robot in the real environment, for example, to reposition it or change its configuration. Therefore, for this to work, we have to establish a common reference frame between the user (and subsequently the pointing device) and the robot.

2.2.3 Pointing correction

While dealing with continuous space of possible pointed targets, the accuracy of pointing gestures may have a significant impact on the interaction performance and cannot be neglected. The simplified models (as compared to the actual human perception models) reduce the accuracy of the estimated locations and directions and has to be carefully assessed. There are three main sources of errors that contribute to the final accuracy: (1) intrinsic characteristics of human perception and motor skills, (2) inaccuracies of gesture interpretation model, and (3) external measurement errors. The first error is characterized as the repeatability of pointing gestures, i.e. how well can a human point at the same target over and over again. The second error is defined as a difference between the estimated target location and the location human intended to point at. Finally, the external measurement errors are those induced by misplacement of the sensors, their noise, etc.

These errors can be reduced by means of additional visual feedback, similar to the cursor on a screen [5], a better gesture interpretation model, and a thorough sensor calibration. However, perhaps a more practical approach is to learn a pointing model directly from the observations. Droeschel et al. utilized Gaussian Process Regression (GPR) to train function approximator that maps human body features extracted from a time-of-flight depth image data to a pointing direction [15]. Authors compared the accuracy of their GPR-based model with the trivial line-based models, i.e. head-hand, elbow-hand and shoulder-hand, and shown significant improvement in the pointing accuracy: 0.17 m vs. 0.39 m (head-hand model) average distance error. One of the important conclusions made by Droeschel et al. is that the trivial line-based methods can be inadequate for certain ranges of target directions: while the GPR-based model produced evenly distributed errors, the line-based models shown significant bias at certain targets.

A work by Mayer et al. follows a similar approach, but instead, they use user-, pose-, and distance-independent quadratic polynomial to correct systematic displacements of pointing gestures [48].

2.3 Relative Localization

An important aspect of absolute pointing gestures is the necessity of a common point of reference between the human and the robot. The gestures are captured in the sensor's coordinate frame, but eventually, have to be interpreted by the robot in its own frame. In the case of external sensing, the sensors are often installed directly on the robot and thus the frame transformation is not necessary, i.e. acquired gestures are already in the robot's frame. However, wearable sensors, e.g. IMUs, report the data in an arbitrary inertial frame and the measurements have to be converted first to the human's frame and then to the robot's frame.

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). Localization of a rigid body, such as a robot, in 3D space, assumes the knowledge of its full 6D pose: three translations and three rotations with respect to x-, y-, and z-axes. Some methods allow one to acquire the entire pose, while other methods could only provide partial information, e.g. a 3D position or a 3D orientation. In general, all localization methods can be grouped into direct and indirect methods.

2.3.1 Direct methods

Direct methods assume that the position of one agent is estimated directly with respect to another agent, using, e.g., triangulation or multilateration of a radio [81], optical, or sound signal [68].

These also include vision-based methods, such as localization using passive [19; 21] and active [17] markers, known geometry, or other visual features of the agent [20]. For example, when it is a human to be localized, features like natural skin color [58], face [55], or even legs [62] can be used.

An interesting example of the partial direct localization is a method implemented in a commercial toy robot BB-8 by Sphero [71]. The robot is able to move on a horizontal flat surface and is controlled by a wearable bracelet-like interface fitted with an IMU to provide velocity commands: the user can perform a push-like gesture with their arm along a given direction to move the robot. Because the robot is bound to a horizontal plane and control in velocity space, it requires localization of only one degree of freedom—a relative heading between the human and the robot, i.e. a relative rotation around z-axis. When the user initiates the localization procedure, a single light on an LED ring on the robot circumference turns on: 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 the pointed direction, they press a button to complete the calibration procedure. Now, the heading of the user's arm as measured by the IMU can be converted into the robot's frame.

2.3.2 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 a global positioning system (GPS) [16], or using techniques that are suitable for indoor use, such as optical motion capture systems like Optitrack, or ultra-wideband (UWB) localization systems [81; 35].

An alternative approach is a co-localization of the agents [69] that individually perform simultaneous localization and mapping (SLAM) of the same environment. By finding correspondences between the maps produced by each robot it is possible to estimate coordinate transformations between the agents.

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 [45].

2.4 Robot selection and identification

When there are multiple robots, and possibly multiple operators, that each need to control their own robot, the system must provide a way to select individual robots among a group. The ability to select a robot is also useful for a single robot / single operator case: it allows one to engage with the robot and trigger the start of an interaction.

Couture-Beil et al. developed a computer vision pipeline that uses an orientation of the user's face to estimate which robot the user wants to engage with [14]. Another work from the same research group by Pourmehr et al. extends that approach with a multi-modal interface based on natural speech where the user can engage with individual robots and groups of robots by using indirect speech references, e.g. "You two! Take off!" [65].

Monajjemi et al. used periodic waving arm gestures to engage with a drone that flies far away (20–30m), once the robot's attention is attracted it approaches the user and waits for another, short distance, commands such as taking a selfie [53].

Robot selection can also be performed indirectly. Cacace et al. proposed a system that implicitly selects a robot from a pool of available robots that are suitable for a particular mission. The availability, in this case, can be determined based on various parameters, for example, the battery level of a quadrotor [10] and its capabilities. Wolf et al. shows a simple yet effective way to select robots within a fixed group by iterating through them using an arm gesture [84].

Pointing gestures are particularly intuitive for the robot selection task. Nagi et al. used a distributed consensus protocol for a swarm of robots equipped with vision sensors to estimate which of the robots is being addressed by human pointing [56]. Interestingly, a robot that is being pointed at has the worst view of a pointing hand: in this case, the finger and the hand coalesce and are perceived as a single blob. That means the closer the robots are in the swarm to each other the more difficult for them to differentiate which one of them is being pointed at—a familiar experience we have from our daily lives.

Inertial sensors allow one to have a trivial robot identification system that is based on a mere comparison of pointed locations with the coordinates of the robots themselves. However, engaging a robot immediately when such comparison succeeds may be undesirable as an indicated point in the environment may cross through many other robots before reaching the one intended by the operator. For this reason, it is necessary to trigger the interaction explicitly, only once the operator is convinced to select the pointed-at robot.

Among the ways to detect pointing events are fixed time delays [33; 13], using a hand gesture of the other (non-pointing) arm [38; 78], or detecting the phase of a pointing arm gesture [58; 15; 61; 7].

Using two wearable sensors setup Broggini et al. trained a 1D convolutional neural network (CNN) to predict the probability of an arm gesture to be a pointing gesture [7]. Once the probability passes an arbitrary threshold the gesture is considered to be pointing and the system can engage the robot and the operator.

2.5 Efficient interaction feedback

A feedback signal is an important element of human-robot interaction as it allows both humans and robots to immediately correct their actions.

There are various ways robots could communicate their state to a user: sounds and speech, lights and graphical displays. The state could also be encoded in the robot's behaviour by adjusting its trajectories.

Feedback sounds and lights are used by many commercial mobile robots. For example, the Parrot Bebop 2 drone would play a distinctive sound pattern and blink its light after a collision, which signifies the system has to be restarted. A toy robot StarWars BB-8 uses the light on its LED-ring during the adjustment of the relative heading between the robot and the wrist controller ForceBand worn by the user; the ForceBand would also play various sounds when it recognizes appropriate control gestures.

For their simplicity lights are also a popular choice in the HRI research. Szafir et al. used a circular strip of light-emitting diodes (LEDs) on the circumference of the drone to study the ways of communicating motion directions to the user. The authors tested four feedback patterns, namely: blinker—a metaphor of car turning signal; beacon—searchlight of a lighthouse; thruster—a metaphor of the flames from the engine of an aircraft; gaze—represent two stripes of lights and mimic eyes. The survey among participants shown that the gaze and blinker metaphors have the best ability to communicate the robot's motion intentions [74].

Monajjemi et al. installed an RGB LED strip on the front side of the drone to communicate its intent to the user. The drone is able to detect a waving user, approach them, and follow hand gesture commands, e.g. take a photo (selfie) of the user. The drone is able to communicate such states as search, approach, being lost, camera timer countdown, etc. [53].

Cauchard et al. studied the ways a drone can communicate its state through emotions that are encoded by changing the drone's speed, altitude, and orientation. For example, the drone would fly low and slow to show it is exhausted, i.e. the battery level is low [11].

Chapter 3

Estimation of Pointed Locations

Human pointing is not a trivial concept and as has been shown in related psychology research depends on many complex parameters, e.g. eye dominance. Not all of these parameters are technically easy or even possible to capture to estimate pointed locations. Therefore, it is necessary to find a reasonable approximation of the true human pointing model that is both sufficiently precise and practical to implement for robotics tasks.

This chapter describes the main concepts related to a human pointing model adopted in our research. We start by formally defining the human kinematics model and coordinate transformations that constitute it. We then define the *pointing ray* associated with the chosen pointing model and show how we obtain it from readings of a wearable inertial sensor placed on the user's arm.

Finally, we describe the estimation of pointed locations based on these models and one of the main contributions of the present research to the state of the art—*estimation of pointed locations in free 3D space* and its validation in a user study.

Estimation of pointed locations requires finding a pointed direction in the form of a ray and intersecting that ray with objects or surfaces in a given environment. The mathematical notion of the ray is important as it constrains intersections in a forward direction only. Estimated locations are relative to the user and depend on the user's posture.

3.1 Pointed directions

In our work we use the *eye-finger* pointing model that is defined by two 3D points: a midpoint between the eyes (often called a "cyclops eye") and a tip of the index finger. We estimate the positions of these points through a minimalistic kinematics model of the human body with only two dynamic parameters—arm's pitch

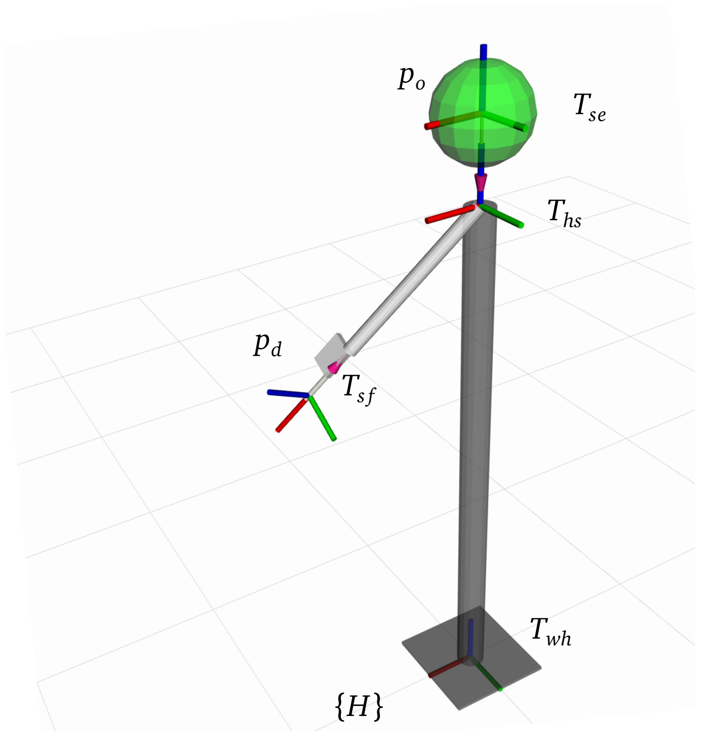


Figure 3.1. Minimalistic kinematics model of the human body adopted in this work. The red-green-blue triads show local coordinate frames that constitute the kinematics chain rooted at the human frame $\{H\}$.

and yaw. In this model we make the following assumptions:

- The user stands upright and points with a straight arm;
- The eyes and the shoulder are vertically aligned, the distances from the ground to the shoulder (L_{hs}), from the shoulder to the eyes (L_{se}) and to the fingertip (L_{sf}) are known and fixed.

The visualization of this model is presented in Figure 3.1.

We define the pointing ray r as a half-line originating at the operator's eyes at point p_o and passing through the tip of the pointing finger at point p_d (direction point). Equivalently, this pointing ray can be defined as a vector with orientation ω_o originating at point p_o:

$$r = (\omega_o; p_o) \tag{3.1}$$

In the arm-rooted pointing models, the orientation ω_o can be directly measured with an inertial sensor worn on the arm; however, in our case, the orientation of the arm does not coincide with the pointed direction and has to be calculated using human body kinematics model mentioned above.

We define the coordinate transformation tree as depicted in Figure 3.1, which is rooted at the user's feet. T_{wh} represents a transformation that defines *human* position and orientation in an arbitrary world frame $\{W\}$, i.e. it is the coordinate transformation that brings one from the world frame to the human frame $\{H\}$[1]. Similarly, T_{hs} is a transformation from the human feet to the *shoulder*, T_{se} is a transformation from the shoulder to the *eyes*, and T_{sf} is a transformation from the shoulder to the *fingertip*. These transformations represent homogeneous coordinate transformation matrices but can be also viewed as a combination of pure rotation and pure translation operations. For clarity we use the latter representation and define a transformation as $T = (\omega; t)$, where ω is a rotation matrix, and $t = [t_x, t_y, t_z]$ is a translation vector.

For clarity, in this section, we assume that human is located at the world's origin and therefore $T_{wh} = I$ is the identity transformation.

We now parametrize individual transformations via predefined human body length parameters:

$$T_{hs} = (I; [0, 0, L_{hs}]) \tag{3.2}$$
$$T_{se} = (I; [0, 0, L_{se}]) \tag{3.3}$$
$$T_{sf} = \left(\omega_{IMU}; [L_{sf}, 0, 0]\right) \tag{3.4}$$

where I is the identity rotation and ω_{IMU} is the rotation measured by the IMU on the user's arm.

Using the above transformations, we define the positions of the fingertip p_d (ray *direction* point) and the eyes p_o (ray *origin* point) in the human frame $\{H\}$:

$$p_o = T_{hs} T_{se} \mathbf{0} \tag{3.5}$$
$$p_d = T_{hs} T_{sf} \mathbf{0} \tag{3.6}$$

where $\mathbf{0} = [0, 0, 0]$ is the zero vector, i.e. the origin point.

Next, we define the direction vector v that connects the eyes and the fingertip and consequently find the orientation ω_o of the pointing ray r:

$$v = p_d - p_o \tag{3.7}$$
$$\tag{3.8}$$

[1]In this notation the first subscript defines a parent reference frame and the second defines a child frame. Therefore the inversed transformation of T_{wh} is T_{hw}.

We now derive the yaw ω_{yaw} and pitch ω_{pitch} rotations of the direction vector v:

$$\omega_{yaw} = RPY\left(0; 0; \arctan\frac{v_y}{v_x}\right) \tag{3.9}$$

$$v_{new_x} = \omega_{yaw} e_x \tag{3.10}$$

$$\omega_{pitch} = RPY\left(0; \arctan\frac{-v_z}{v \cdot v_{new_x}}; 0\right) \tag{3.11}$$

where $RPY(roll; pitch; yaw)$ is a function that maps roll, pitch, and yaw angles to a rotation matrix; $e_x = [1, 0, 0]$ is the x-axis basis vector, v_{new_x} is an intermediary unit vector along the x-axis of the coordinate frame defined by ω_{yaw}.

Finally, we can define the rotation of the pointing ray as a matrix product of the two rotations:

$$\omega_o = \omega_{yaw} \omega_{pitch} \tag{3.12}$$

Note that we ignore the roll rotation, i.e. the rotation around the user's wrist, as it does not influence the pointed-at location in our model.

An important drawback of the proposed simplified implementation is an increased tangential pointing error. For example, assuming the user is pointing straight in front of them, such that the yaw of the pointing ray $\omega_{yaw} = 0°$; assuming the user's arm length $L_{sf} = 0.69$m (shoulder to finger) and the shoulder length $L_{ns} = 0.18$m (neck to shoulder), then the pointing arm heading (yaw) angle in this case is:

$$\omega_{yaw} = \arccos\frac{0.18}{0.69} \approx 15.1° \tag{3.13}$$

That means the tangential pointing error at the distance of 2.0m equals 0.53m.

Later, in Chapter 6, we show that this seemingly large error can be neglected if a user is provided with real time visual feedback, such as a motion of the robot itself.

3.2 Pointed locations

3.2.1 Environment surface and objects

Now that we found the pointing ray r, we need to intersect it with the environment to find the location the user is referring to. In its simplest, the environment

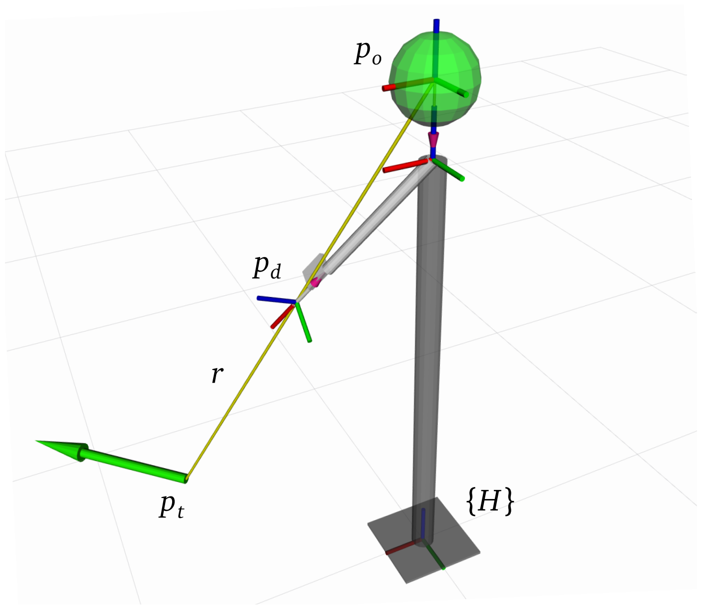


Figure 3.2. Finding pointed location p_t as an intersection of the pointing ray r with the ground plane. The green arrow represents the target pose of a robot at location p_t.

represents a flat world free of obstacles and is defined by a single horizontal plane at the ground level (Figure 3.2). This is a fair assumption for many practical applications, such as navigation of ground robots or landing flying robots at precise locations on flat terrain.

We define the pointed (target) location p_t as:

$$p_t = r \cap S \tag{3.14}$$

where S is a surface the pointing ray r is intersected with.

Note that the *full pose* of a rigid body, such as a robot, in 3D space is defined by six degrees of freedom: three translations and three rotations with respect to x-, y-, and z-axes. In our model we explicitly define only the target *position* of a robot, while the target *orientation* (yaw angle) is set implicitly along a pointed direction, i.e. equals to ω_{yaw}.

For more complex environments and tasks, e.g. navigation on rough terrain or manipulating objects, a 3D model of the surrounding world must exist. Such a model can be created with various sensors including Light Detection and Ranging (LIDAR) devices and depth cameras that many field ground and flying robots are often equipped with. An intersection of the pointing ray r with such a surface may result in multiple points, in this case, the one closest to the operator must be considered as the intended target point p_t:

$$p_t = \underset{p \in r \cap S}{\arg\min} \|p - p_o\| \tag{3.15}$$

3.2.2 Free 3D space

Although the flat world model can be efficiently used to guide flying robots at fixed heights, it essentially allows one to control independently only two degrees of freedom, i.e. along x- and y-axes (with z-axis pointing up); however, one would ultimately expect a control interface for a flying robot to provide a capability to change the altitude as well, or more generally—define a 3D target position in free space.

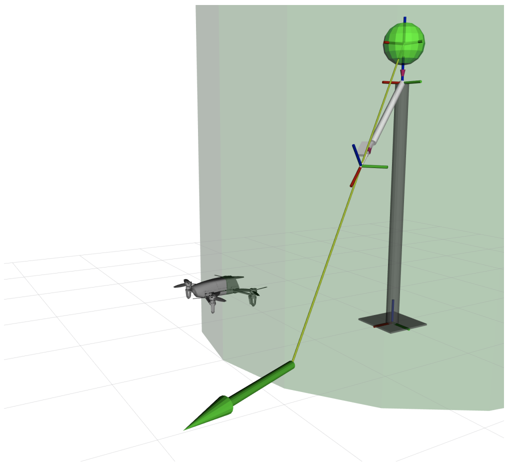
The problem that arises from this requirement is that it is impossible to unambiguously define p_t from the pointing ray r as the target location may lie anywhere along the ray and there is no surface to intersect with. For instance, consider the case in which the operator is pointing at the robot that flies a few centimeters above the ground and one meter in front of them. The operator then adjusts their pointing stance by slightly increasing the elevation of their arm. Does this mean that the robot should move farther from the user while staying at the same height, or that it should move up?

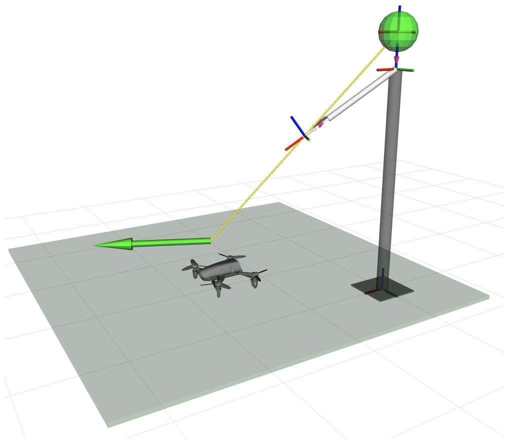
We approach this problem by introducing a set of *virtual workspace surfaces* that constrain the robot motion. We let the operator switch between these workspace surfaces to enable robot control in the entire 3D space.

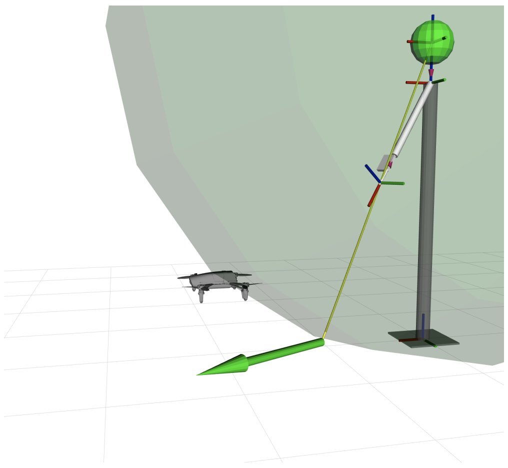
Workspace shapes

We now describe four possible shapes for workspace surfaces: cylinder, horizontal plane, sphere, and vertical plane (Figure 3.3). The first three shapes are defined as a one-parameter family of surfaces. When the user switches to the shape, the free parameter is set in such a way that the surface passes through the current position p_{robot} of the robot. The vertical plane shape requires to set an additional parameter: plane orientation around z-axis.

Cylinder $S_{\mathbf{cylinder}}$ with a vertical axis passing through the user's head (p_o) (Figure 3.3a); the cylinder radius is the free parameter. This option allows the operator to control the robot's vertical position without limitations, never affecting the horizontal distance to the operator.


(a) Cylinder


(b) Horizontal plane


(c) Sphere

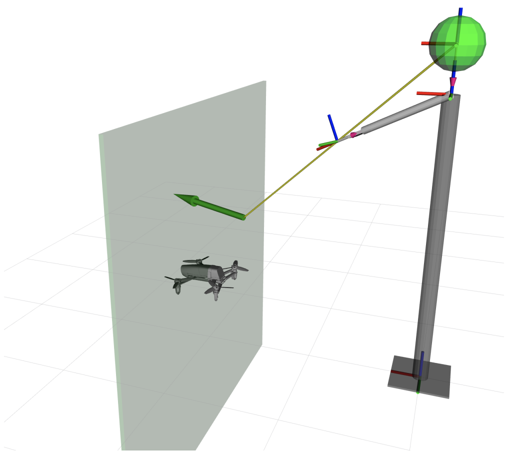

(d) Vertical plane

Figure 3.3. The variety of workspace shapes implemented in this work.

Horizontal plane $S_{\textbf{h-plane}}$ with the distance from the ground plane as the free parameter (Figure 3.3b). This workspace serves a similar role to the ground plane that humans have life-long experience with: it is a natural choice for indicating locations and an intuitive tool to control the robot's position on that plane but does not allow height control.

Sphere $S_{\textbf{sphere}}$ centered at the user's head (p_o); the sphere radius is the free parameter (Figure 3.3c). Operating in this workspace roughly corresponds to the user holding a rod in their hand, with the robot affixed at its extremity.

Vertical plane $S_{\textbf{v-plane}}$ parallel to z-axis and perpendicular to horizontal projection of the line connecting the user's head (p_o) and the robot p_{robot} (Figure 3.3d); this shape therefore has two free parameters: rotation around z-axis and the distance. This option allows one to change the robot's height without limitations and move it on a straight line with respect to the ground. This modality loosely resembles image plane object manipulation.

To achieve intuitive interaction, an operator should always have a clear idea of the workspace the robot is operating in; if the workspace shape is known, *the robot position itself* uniquely defines the workspace surface S. For example, if the workspace shape is $S_{\text{h-plane}}$, the user can expect that the robot will keep its current vertical position; if the workspace shape is S_{cylinder}, one can easily visualize the user-centered cylinder passing through the robot; with the workspace set to S_{sphere} the user can predict the robot to keep constant distance to their head. In turn, if S is known, the user can always predict p_t given r.

On the contrary, the vertical plane shape $S_{\text{v-plane}}$ is a more complex concept and demands additional cognitive effort from the user: initially, the surface is set to pass through the robot and is oriented perpendicular to a horizontal projection of user–robot line, once the robot moves this line rotates but the plane keeps its configuration—now the plane orientation is an implicit state the user should remember. In the following discussion we keep $S_{\text{v-plane}}$ for the sake of completeness, but overall consider it a suboptimal choice.

Workspace switching

Further, we considered the case when the user can switch between multiple shapes described above. To minimize the complexity of the system, we limit the user to toggle between *two* possible shapes; properly choosing these allows one to reach any position in 3D space.

Among the possible choices, $\langle S_{\text{cylinder}}, S_{\text{sphere}} \rangle$ and $\langle S_{\text{v-plane}}, S_{\text{sphere}} \rangle$ pairs are discarded as they prevent controlling the robot's horizontal position independently from its height. For example, if the initial position is close to the user's feet, reaching a point a few meters away can only be executed through a convoluted operation: choosing the cylindrical or vertical plane shape, raising the robot to the arbitrary height, then switching to a spherical shape and lowering the robot again while it gains distance.

Similarly, $\langle S_{\text{h-plane}}, S_{\text{sphere}} \rangle$ does not allow independent control of the robot's vertical position, which is something one would expect. Reaching a point at a considerable height in this model requires one to fly the robot farther than necessary using the plane shape, then switch to the sphere to gain height while reducing distance.

We, therefore, determine to allow the user to toggle only between $S_{\text{h-plane}}$ and S_{cylinder} workspace shapes. The possible switching sequence is shown in Figure 3.4: the user starts moving the robot on a cylindrical surface and then switches to a horizontal plane (Figure 3.4a–c).

Although $\langle S_{\text{v-plane}}, S_{\text{h-plane}} \rangle$ combination is a reasonable alternative, the vertical plane $S_{\text{v-plane}}$ has important limitation as compared to S_{cylinder}: the orientation of the plane is set at the moment the workspace is switched and the user has to remember its configuration or re-setup it every time they want to change horizontal movement direction. Unlike $S_{\text{h-plane}}$ shape that is always parallel to the ground and easy to visualize, $S_{\text{v-plane}}$ has no such environment reference. For this reasons we reject $S_{\text{v-plane}}$ and $\langle S_{\text{v-plane}}, S_{\text{h-plane}} \rangle$ combination as overly complex and non intuitive. Note that using vertical planes parallel to one of the walls of a rectangular room or any indoor environment with a few clearly defined vertical planes would be a good alternative, but requires some knowledge of the environment.

3.3 Implementation

3.3.1 Gesture sensing

We implemented the system using an inexpensive wearable IMU (Mbientlab MetaWearR+ [51]) that has a form-factor of a wrist smartwatch (Figure 3.5, right). The device is equipped with a three degrees of freedom (3-DoF) accelerometer, 3-DoF gyroscope, and 3-DoF magnetometer. The onboard firmware runs the necessary sensor fusion algorithms in real time and outputs an accurate estimation of the device's absolute 3D-orientation in an arbitrary fixed reference frame whose

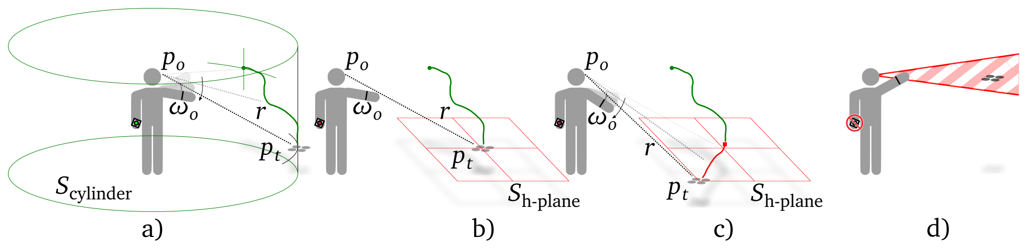


Figure 3.4. Interaction using workspace shapes: a) User guides the drone in the primary workspace S_{cylinder}; b) User switches to the secondary shape by pressing a button; c) User guides the drone in secondary workspace $S_{\text{h-plane}}$; d) User is forbidden to switch the workspace to $S_{\text{h-plane}}$ when the drone flies close to the eyes height, in this case, the pointing ray is almost parallel to the surface and makes it very difficult to accurately control the drone.

z-axis points up. The device also features a microswitch button, a light-emitting diode (LED) light, and a vibration motor. The data is streamed to the host PC with approx. 50 Hz rate via a Bluetooth Low-Energy (Bluetooth 4.0) link.

The acquired orientation is used within the head-finger pointing model (described in Section 3.1) to recover r, which is then intersected with the active workspace surface S to define the pointed-to location.

3.3.2 Workspace switching

We define S_{cylinder} as the primary workspace shape (Figure 3.4a), i.e., the one that is active by default; $S_{\text{h-plane}}$ is considered a secondary shape (Figure 3.4b), i.e., one that user can switch to upon request (Figure 3.4c).

The operator switches between the two shapes using a single spring-loaded trigger button on a joystick (Logitech F710); the other joystick buttons and controls are ignored in our experiments. When the trigger is in its default position (not pressed), the primary workspace is used; keeping the trigger pressed uses the secondary workspace. This specific choice is crucial for usability, for two reasons.

First, mapping the explicit state of the button to the chosen workspace shape—an important state of the system—ensures that the operator is kept aware of the system state (the user must consciously keep the button pressed). On the contrary, toggling the active workspace once the button is released, would make the state implicit (the one the user cannot directly observe).

Second, the trigger functions as a dead man's handle and ensures a fail-safe behaviour by switching the active workspace to the primary (cylinder) config-

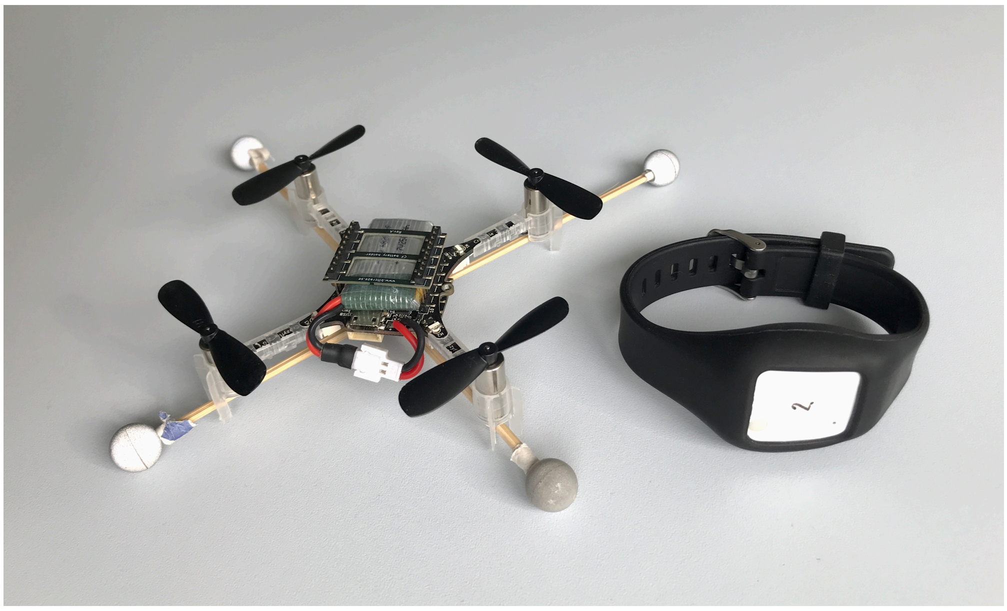


Figure 3.5. The hardware used in the experiments: (*left*) Bitcraze Crazyflie 2.0 quadrotor with retro-reflective markers for motion capture system; (*right*) Mbientlab MetaWearR+ IMU bracelet.

uration when the trigger is released. Since the cylinder workspace is centered around the user, it is impossible for the robot to collide with them because of a control mistake. For this reason, the system may also refuse to switch to the secondary workspace if it is considered unsafe: for example, when the drone is flying at the height of user's eyes and the pointing ray is almost parallel to the horizontal plane; in this case, small changes in the arm elevation would result in very large displacements of the pointed location. We prevent switching to the secondary workspace if the elevation angle of the drone with respect to the user's head is within $\pm 5^\circ$ from the horizontal direction (see Figure 3.4d). Whenever a requested switch to the secondary workspace is refused, the joystick vibrates to make sure the operator is notified; an alternative approach would be to mechanically block the trigger button when it is unsafe to use it.

3.4 Experiments

We conducted an experimental study that evaluates the efficiency of the proposed 3D-control method against conventional joystick control.

Experiments take place in a room with a safety net of roughly 6x6 meters

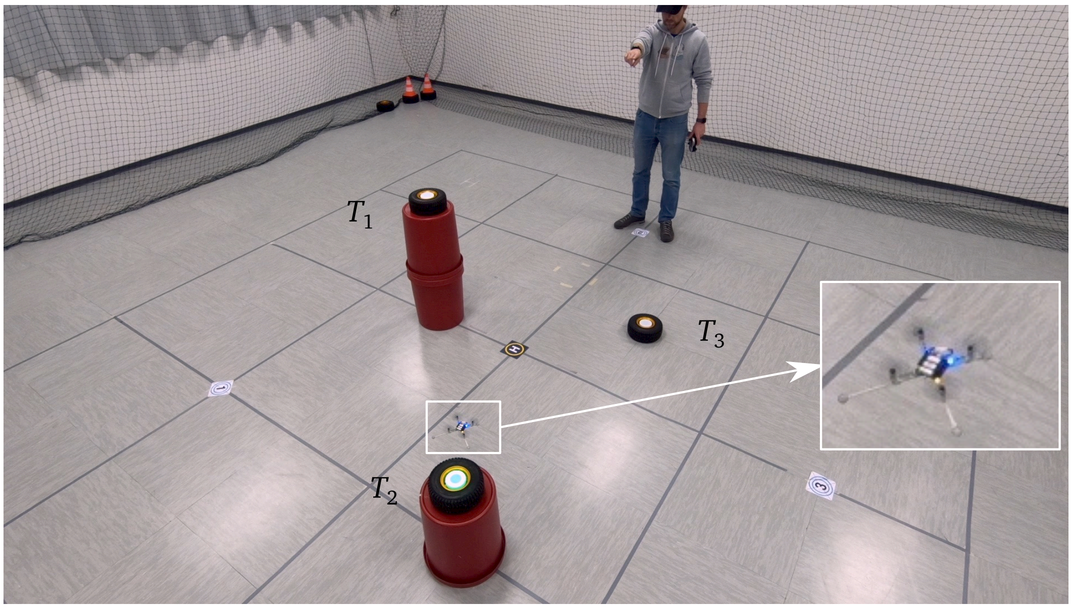


Figure 3.6. An overview of the experimental environment during one of the sessions. The targets $\{T_1, T_2, T_3\}$ (LED beacons) are placed at the wheel hubs that are located at different heights.

size, outfitted with a commercial optical motion capture system (12 Optitrack PRIME17-W cameras). The Optitrack data is streamed to the Robot Operating System (ROS) with a 30 Hz rate.

We use a miniature quadrotor Bitcraze Crazyflie 2.0 [3] tracked through a rigid-body marker (Figure 3.5, left) to perform closed-loop velocity and position control. We also track the location of the user's head through a rigid-body marker attached to a hat. The coordinate transformation between the human and the robot is therefore known at any moment in time.

We configure the kinematic parameters of the human body before the interaction starts: we set the height of the user's shoulder, the position of the head with respect to the shoulder, and the length of the user's arm. That allows us to minimize the pointing model errors and thus reduce the displacement between the pointed and the real position of the robot.

We recruited 10 participants (male, mean age 31.1, sd = 5.0) with computer science background, whose goal was to fly the miniature quadrotor over a set of three flat stationary wireless LED beacons (in-house hardware based on Adafruit nRF52 Feather Bluetooth LE board with a ring of 24 RGB NeoPixel LEDs) placed at different predefined heights at known locations (Figure 3.6).

For each subject we recorded two sessions: one, using the pointing interface

(IMU-equipped bracelet on the wrist of their dominant arm); and another, using joystick control (Logitech F710). The order of the sessions was assigned to each subject at random. Each session consisted of three runs. During each run, a subject was asked to stand at predefined locations 1–3 and to fly the quadrotor between given targets. The following high-level description of the task was given to participants: “once a target gets illuminated, fly the drone above it”.

All subjects reported to be proficient or expert joystick users and had little or no previous experience with the proposed pointing interface. Short “dry run” sessions were performed for all the users to familiarize them with the interfaces and the task.

The experimental session with the pointing interface proceeds as follows:

1. The operator enters the flying arena and stands at the location 1, the quadrotor lies on the floor at the center of the room.

2. The operator presses once the switch on the bracelet to take off the drone, and the second time to initiate the interaction.

3. The operator then points at the drone and holds his arm still for 3 seconds.

4. The bracelet vibrates, the drone makes a small “jump” to show that it is now attached to the operator; the workspace is initialized with the primary surface (cylinder).

5. One of the targets lights up in blue.

6. The operator directs the robot to the target, while when necessary switches the active workspace to the secondary surface (horizontal plane); once within the confirmation zone (10 cm from target’s center in a horizontal plane and 20 cm in along z-axis), the target turns yellow and a timer is triggered.

7. To clear the target the subject must keep the robot within the confirmation zone for 2 seconds. Then, the target shortly turns green and switches off; if the robot leaves the confirmation zone before the two seconds expire, the target turns blue and the timer is reset.

8. Once a target is cleared, the next target lights up in blue: steps 5–7 are repeated.

9. Once all targets are cleared, the operator directs the quadrotor to the landing spot and lands it there by holding their arm still for 3 seconds.

10. The operator moves to the next numbered location and the steps 2–9 are repeated until all three runs are completed.

The procedure for experiments with the joystick is similar but omits steps 3 and 4 of the above sequence (selection of the quadrotor and switching of the workspace surfaces), which are specific to the pointing interface. The joystick control works in drone egocentric frame and thus before acquiring the first target, the operator adjusts the drone's heading to their liking; we further exclude this trajectory segment from our analysis for both interfaces.

3.4.1 Data collection

To assess the performance of the system, we collected the ground truth positions of the participants and the quadrotor, and the times of state transition events of the targets. In our analysis, we ignore parts of the trajectories from the moment the operator takes control till the moment the *first* target is cleared and from the moment the *last* target is cleared till the landing as they do not represent the actual task. We split the resulting trajectories into three segments. Each segment represents a part of the trajectory between two targets. This yields a total of 90 segments for each modality (pointing, joystick), i.e. three segments per run per operator per location.

We collect the data with a standard ROS tool `rosbag` and analyze it offline using Python.

To compare the performance of the two interfaces we use the *time-to-target* and the *trajectory length* metrics, which are applied per segment of the flight path.

3.5 Results

In Figure 3.7 we report an evolution of the distance to target in time; the comparison of means on the right plot shows that the performance of both interfaces is very similar. These results also confirm our previous finding for a similar 2D task [29].

To find exact differences in the performance of the two interfaces, we further analyze *relative lengths* of trajectories flown with each interface. This measure is calculated as a ratio between the length of a segment flown and the length of an ideal trajectory, i.e. a straight distance between corresponding targets. The closer the relative length to 1.0, the better the performance of a given interface. Figure 3.8 shows these data on a 2D plot, where median relative lengths for

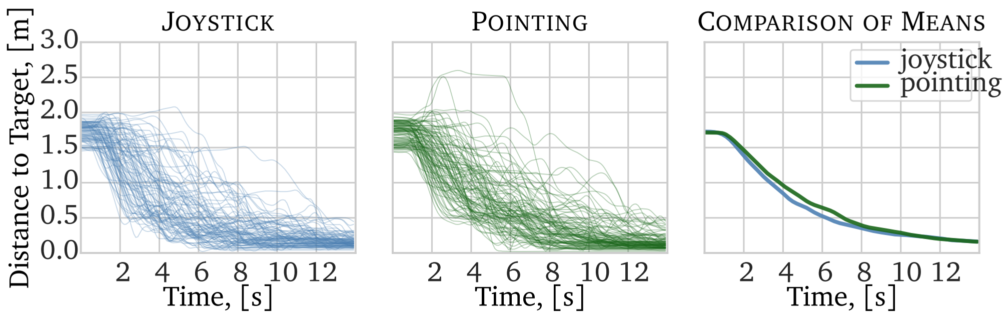


Figure 3.7. Analysis of the evolution in time of the distance to the target, for each trajectory flown; each trajectory is represented as a line; $t = 0$ on the plot corresponds to the t_0 time of each trajectory. Left: joystick interface ($N = 90$). Center: pointing interface ($N = 90$). Right: average over all trajectories for each interface.

pointing and joystick interfaces define the x and y coordinates of black dots and their mean values define blue crosses. The relative length of trajectories flown with pointing is shorter than the one for the joystick interface.

In Table 3.1, we present a detailed comparison of median and mean trajectory lengths (relative to a straight line) and median and mean times to target for the two interfaces. For each subject (rows) we report the median and mean performance over all segments, for each of the two control modalities (columns). Since the samples are matched, i.e. we can compare the performance of the same subject on both control modalities, the hypothesis that pointing yields better performance than joystick—lower length or duration—can be assessed for statistical significance using a *paired* difference test. Because the sample size is small and the population cannot be assumed to be normally distributed [47], we use the Wilcoxon signed-rank test [82] as an alternative to the paired Student's t-test. We found that pointing yields shorter trajectories ($p < 0.05$) than joystick; the impact of the control mode on trajectory duration is not statistically significant, even though the median duration is slightly better for pointing (14.4 s) than for Joystick (15.4 s).

3.5.1 Discussion

The shorter lengths obtained with the pointing interface can be explained by smoother trajectories. In fact, visual comparison of trajectory segments performed with the joystick (Figure 3.9) and the pointing (Figure 3.10) shows that joystick trajectories are comprised of multiple orthogonal pieces in both vertical

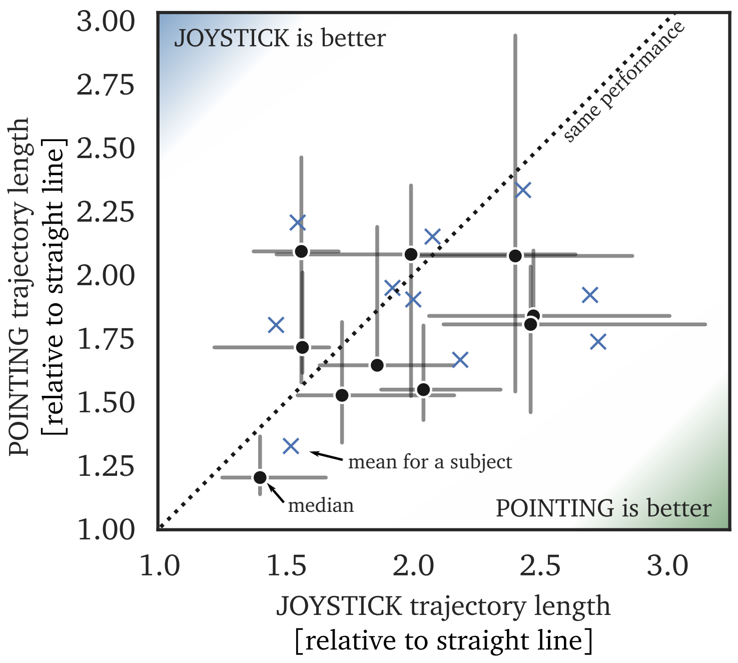


Figure 3.8. Relative lengths of the trajectories flown with the pointing and the joystick interface, normalized by straight distances between targets. The black dots and the blue crosses represent median and mean values respectively for each user ($N_u = 10$). The error bars represent 25-th to 75-th percentile.

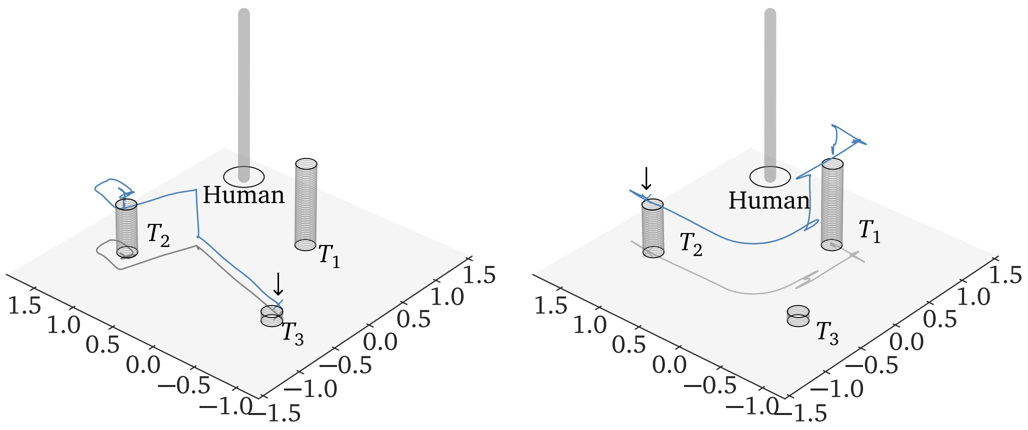


Figure 3.9. Visualization of two segments of a trajectory performed with the joystick. The short cylinders represent the targets and the tall thin cylinder represents the user.

Table 3.1. Performance per subject (median and mean over all segments)

	Trajectory length, [relative to straight line]				**Duration, [s]**			
Subject	Joystick		Pointing		Joystick		Pointing	
	med	mean	med	mean	med	mean	med	mean
1	1.72	**1.92**	**1.53**	1.95	11.6	**12.7**	**10.4**	14.9
2	**1.57**	**1.46**	1.72	1.81	16.8	18.2	**16.5**	**18.1**
3	**2.00**	**2.08**	2.08	2.15	13.3	13.9	**11.5**	**12.7**
4	2.48	2.70	**1.84**	**1.92**	25.8	26.2	**15.7**	**15.9**
5	2.47	2.73	**1.81**	**1.74**	19.8	18.6	**14.8**	**16.4**
6	1.40	1.52	**1.21**	**1.33**	14.4	14.8	**11.9**	**11.8**
7	2.41	2.44	**2.08**	**2.34**	**15.6**	**16.0**	22.2	21.8
8	**1.56**	**1.55**	2.09	2.21	**10.2**	**10.7**	11.8	14.1
9	1.86	2.00	**1.65**	**1.91**	**11.1**	**12.3**	14.9	15.5
10	2.04	2.19	**1.55**	**1.67**	15.6	15.4	**14.0**	**15.2**

The better result of the two interfaces is shown in bold.

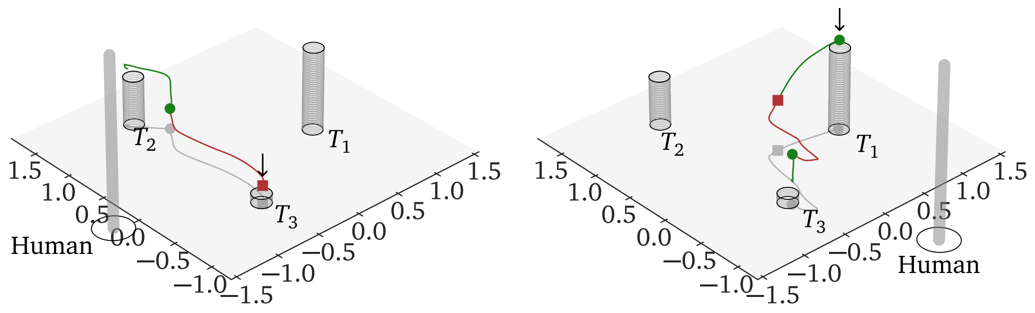


Figure 3.10. Visualization of two segments of a trajectory performed by pointing. The green segment is performed with using the cylinder workspace, while the red one using the horizontal plane workspace. The short cylinders represent the targets and the tall thin cylinder represents the user.

and horizontal planes, while for the pointing that is true only for vertical segments. On the other hand, the time to target metric did not show statistically significant improvement. This could be due to several reasons.

First, with the pointing interface the operator directly indicates the target position, meaning that low-level trajectory execution (e.g. deceleration when approaching the target) is performed automatically and more efficiently. On the contrary, with the joystick interface operators give commands in velocity space and have to plan the trajectory themselves. This leads to a sub-optimal control and results in longer trajectories.

Second, quadrotor control with pointing requires more attention from the operator because they cannot "pause" the interaction: once they are attached to the drone they should be careful with the movements of their pointing arm as it immediately translates into the movements of the robot. On the contrary, joystick interface allows to leave the controls at any moment—the drone will just stop and hover. Our intuition is that these differences lead to different strategies adopted by the participants: with pointing they tend to be precise and slow, while with joystick more aggressive, resulting in trial-and-error approach with fast and less precise commands.

In their informal feedback, subjects praised the pointing interface for its ease of use and engagement; however, some of them also mentioned it was more stressful because they could not just "leave the controls" (which is instead possible with the joystick) which resulted in arm fatigue during long sessions. One subject mentioned he prefers the joystick interface for such a precise control task, but would prefer pointing for more coarse control tasks. Our intuition is that this preference could be related to an increased arm tremor that subjects typically experience in precise pointing tasks [49] and that in turn may affect their confidence and the stress level.

Using this feedback, we identify two problems: 1) inability to disengage from the robot; 2) arm fatigue. The first problem can be solved by introducing additional "clutch" that would engage control only while a button is held down, i.e. a drag&drop concept. The second problem can be partially solved by the same strategy, which would allow users to rest. More generally, the system can be improved by allowing users to move around while controlling the robot. We implemented this by equipping the operator with a visual odometry sensor. We discuss this option in detail in Chapter 4 (Section 4.5). This also allows users to walk closer to the target position, which yields increased control accuracy and lower arm fatigue as the required arm elevation is reduced.

3.6 Conclusions

In this chapter, we defined a human pointing model: a mapping between the user's stance and the pointed-at location in the environment. We first estimate a pointed direction in the form of a ray using a single IMU sensor, and then intersect this ray with an environment to estimate the pointed-at location. Our main contribution is a method that allows one to estimate such locations even in free 3D space, for example, when controlling a quadrotor. In a user study, we show that the proposed method compares favorably with a baseline joystick control interface.

Chapter 4

Relative Localization

In the previous chapter, we have described a human pointing model that allows one to estimate pointed directions and locations using a single inertial sensor worn on the user's arm. This model inherently defines estimated pointed directions and locations in the human frame of reference. The relation between that frame and the frame of a robot may not be known and has to be established before the interaction can proceed; in other words, we need to localize the human with respect to the robot.

In this chapter, we describe several localization approaches that we used in our work and validate them experimentally. We start with a trivial absolute localization method using a global reference frame, we then describe a relative localization method based on fiducial markers, and finally—our main contribution—a *relative localization from motion*.

4.1 Global reference frame

A global reference frame or a world frame $\{W\}$ is an arbitrary frame that is common to all the interacting agents—robots and humans. When the positions (more generally 6D poses) of a robot and a human are known in the world frame, finding a relative coordinate transformation between the two frames is trivial:

$$T_{rh} = T_{wr}^{-1} T_{wh} \tag{4.1}$$

where T_{wh} is a coordinate transformation from the world frame $\{W\}$ to the human frame $\{H\}$, T_{wr} is a transformation from the world frame to the robot frame $\{R\}$, and T_{rh} is the resulting transformation from the robot to the human frame.

Consequently, to find a pointed location p_t expressed in the human reference frame in the frame of reference of the robot:

$$p_t^{\{R\}} = T_{rh} p_t^{\{H\}} \tag{4.2}$$

Localizing agents within the global reference frame outdoors can be partially done employing a global navigation satellite system (GNSS), for example, GPS. Such localization allows one to directly retrieve relative positions of the agents but does not provide relative heading. This problem can be tackled with hybrid approaches, where the heading of each agent is estimated with respect to the global frame using their odometry models. While the presence of odometry systems is relatively common in robots, they are not easily available for humans. In this case, an electronic compass (magnetometer) worn by the user can be used to estimate their heading with respect to the heading estimated by the compass of a robot. Indoors, where the GNSS is not available, motion capture systems or ultra-wideband (UWB) localization can be used.

In our first work [24] we establish a common reference frame between the user and multiple ground robots in two steps:

1. The user fixes their heading with respect to a known absolute frame by pointing along its x-axis of the environment and confirms the heading with a hand gesture.

2. The user points at one of the robots which has a known position in the absolute frame of the motion capture system. Since the pointed location is known in the frame of reference of the user we can then find his/her location in the absolute frame by linking it to the robot's location.

The applicability of GNSS is limited indoors and in the areas with high localization errors, such as in cities with a dense building. On the other hand, high precision motion capture localization systems are limited to designated indoor areas because they require various infrastructure (electricity, networking, scaffolding, etc.).

4.2 Fiducial markers

A more practical solution is to directly acquire the transformation between the robot's frame and the frame of reference of the human.

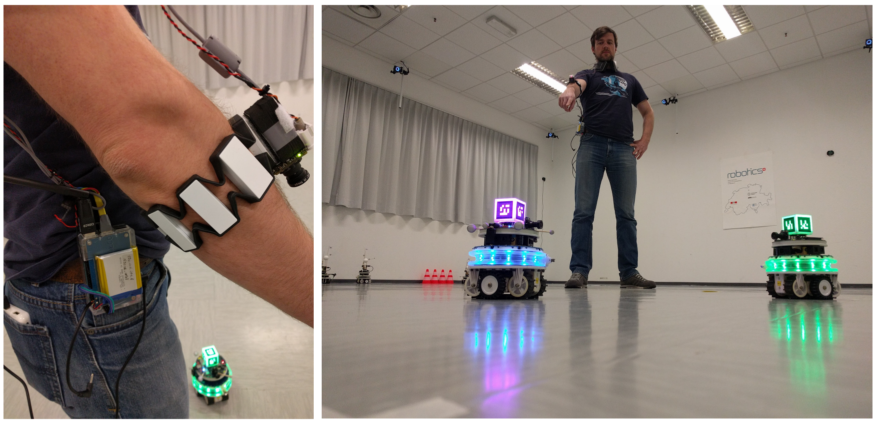


Figure 4.1. Relative localization using a camera and fiducial markers. (*Left*) An RGB-camera is mounted on top of the wearable IMU. (*Right*) The user points at one of the robots to select it, the robot, in turn, changes its beacon color.

We installed a 3D fiducial marker on the robot at $p_m^{\{R\}}$ and mounted a camera on the user's forearm paired with wearable IMU (Figure 4.1) at $p_c^{\{H\}} = T_{hc}o$, where T_{hc} is a transformation from the human frame to the camera frame.

Using computer vision algorithm for 3D pose reconstruction[1] we can estimate the coordinate transformation T_{cm} between the camera and the marker. Since the dynamic position of the camera is known in the human reference frame we can acquire the necessary coordinate transformation from the user to the robot:

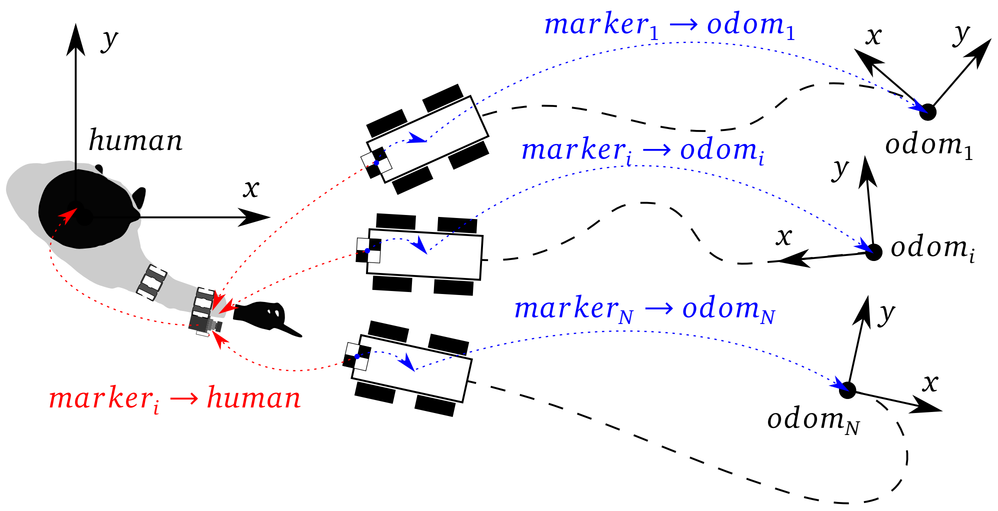
$$\begin{cases} p_m^{\{R\}} &= T_{rh} p_m^{\{H\}} \\ p_m^{\{H\}} &= T_{hc} T_{cm} \mathbf{0} \\ p_m^{\{R\}} &= T_{rm} \mathbf{0} \end{cases} \tag{4.3}$$

$$T_{rm} = T_{rh} T_{hc} T_{cm} \tag{4.4}$$

$$T_{rh} = T_{rm} T_{cm}^{-1} T_{hc}^{-1} \tag{4.5}$$

where T_{hc} is a transformation from the human frame $\{H\}$ to the camera frame, T_{rm} is a transformation from the robot frame $\{R\}$ to the marker attached to it, and $\mathbf{0} = [0, 0, 0]$ is the origin point.

[1]Via `ar_track_alvar` package of the Robot Operating System (ROS).



$marker_1 \rightarrow odom_1$
y
x
y
$human$
x
$odom_1$
$marker_i \rightarrow odom_i$
y
x
$odom_i$
$marker_N \rightarrow odom_N$
y
x
$odom_N$
$marker_i \rightarrow human$

Figure 4.2. Relative localization for multiple robots using fiducial markers. *Checker boxes* depict markers; *colored dashed arcs* constitute the coordinate transformation tree with the arrows pointing to corresponding parent frames; *blue arcs* are the robots' individual transformation trees rooted in their odometry frames; *red arcs* are the transformations acquired during the localization process and rooted in the human frame. The *long-dashed curves* are the robots' traveled paths.

4.2.1 Application to multiple robots

The approach naturally extends to multiple robot setups. When there are several robots nearby, the user can individually localize them by pointing the camera at each of their markers. In this case, all the robots will become a part of the same coordinate transformation tree rooted at the human frame $\{H\}$. It allows the system not only to acquire relative transformations between the human and the robots but also to localize the robots relative to each other. After that, the robots need to keep updating their positions in their local frame, e.g., an odometry frame. We briefly show the described process in Figure 4.2.

4.2.2 Experiments

To proof the viability of the proposed relative localization method, we designed an experiment where a user was required to repeatedly select a robot and indicate a given target location. The locations are placed on a floor in a 5x5 grid pattern with 1.2m spacing, making the farthest locations to be approx. 6.8m away. Out of the 25 locations one was reserved for the human and one for the robot. We considered the robot location a target as well, which allowed us to collect 24 data points per trial. Also, for the ease of pointing, each location was marked with a small plastic ball ($\varnothing 3.5cm$) and a paper label with the location number. The user was instructed to point with a straight arm.

To initiate a trial and proceed through a set of actions in the experiment user had to press a single button on the game controller. With every button press the system generates voice feedback with a status or instruction and proceeds in the following way:

1. With the first button press the system initializes a random permutation of the 24 targets and informs the user accordingly voice feedback: *"Initialized"*.
2. Then, with every consecutive button press the system picks the next target location from the list and instructs the user: *"Localize the robot, then point to target N"*, where N is the target number. At this stage, the user points at the robot to localize it and the robot changes its LED-beacon color for confirmation. The estimated robot pose is utilized to create a coordinate frame transformation tree which is then published to the ROS network using `tf`. We denote the estimated robot frame as $\{R\}^*$.
3. The user points at the given target and presses the button again. System replies: *"Recording... Done"*. The estimated pointed location $p_t^{\{H\}}$ is converted to the robot's local frame using the estimated frame $\{R\}^*$. The resulting point $p_t^{\{R\}}$ is stored in the output dataset along with the target number.
4. The sequence 2–3 repeats until the list of targets is exhausted.
5. At the end of a session the system informs the user: *"That's it. Thank you!'*

4.2.3 Metrics

To assess the accuracy of the estimated pointed locations, we convert the points $p_t^{\{R\}}$ back to the human frame $\{H\}$, but using the *true* pose/frame $\{R\}$ of the robot: $x^{\{H\}} = T_{rh}^{-1} p_t^{\{R\}}$, where $x^{\{H\}}$ is the location where the robot would have arrived using estimated frame $\{R\}^*$ and estimated pointed location $p_t^{\{R\}}$, and T_{rh} is the *true* transformation matrix from $\{H\}$ to $\{R\}$.

We compare the estimated target location $x^{\{H\}}$ with the true target location $t^{\{H\}}$ and report three position errors. For the clarity we further drop the $\{H\}$ superscript:

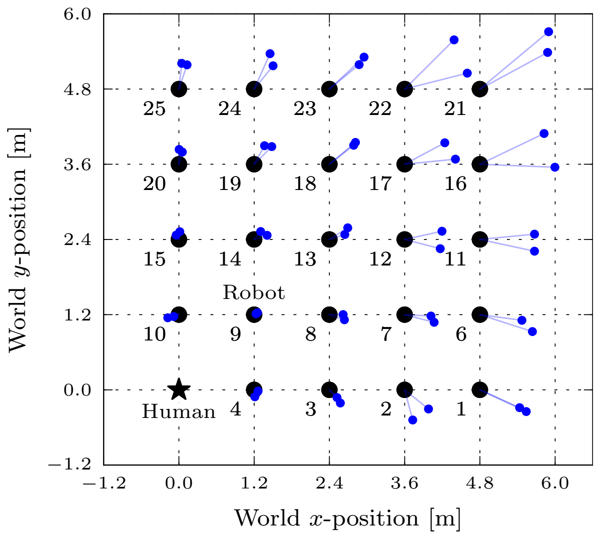


Figure 4.3. Distribution of the euclidean position errors ε_e (blue dots and bars) on a map of the experimental environment. Large black dots represent the targets, the numbers are the target labels, and the star is the location of the human. The robot is located at target 9 and oriented towards target 4.

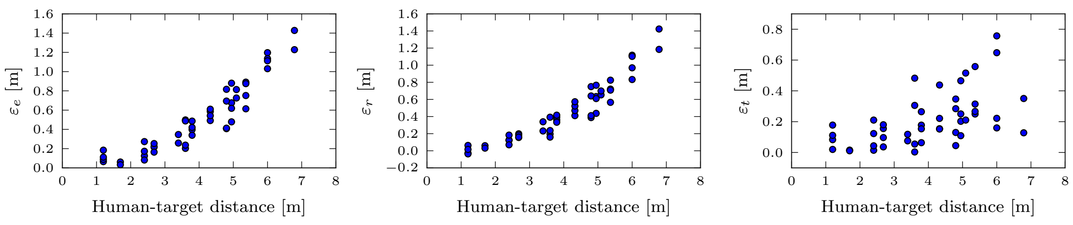


Figure 4.4. Scatter plots of the position errors.

- Euclidean distance: $\varepsilon_e = \|x - t\|$
- Radial distance: $\varepsilon_r = |\rho_x - \rho_t|$
- Tangential distance: $\varepsilon_t = |\alpha_x - \alpha_t|\,\rho_t$

Where ρ_x, α_x and ρ_t, α_t are the polar coordinates of x and t, accordingly, with respect to the origin at $\{H\}$.

4.2.4 Results

We analyzed the errors of the proposed relative localization technique in the controlled indoor environment and found the maximum position error in the navigation task to be $1.4m$ at the distance of $6.8m$ [26].

Following the described procedures we performed two trials and calculated the errors. The position errors are presented in Figure 4.4, while the distribution of the euclidean position errors on a map of the experimental environment is depicted in Figure 4.3.

As can be seen from the plot, the errors grow with a distance giving the maximum position error (ε_e) of approx. $1.4m$ at $6.8m$ distance. The errors grow non-linearly and at the half of the maximum distance are almost three times smaller than the maximum reported error, giving approx. $50cm$ offset from the target position.

A slight asymmetry of the errors along the rows of targets 1, 2, 3, 4 and 10, 15, 20, 25 is probably associated with the simplified model of human body we have used, i.e. we considered the shoulder is fixed, while in practice the user could have turned his torso differently for these two sets of targets.

This approach works relatively well in indoor environments but will have all the problems of computer vision systems in varying light conditions outdoors.

4.3 Predefined heading

The major problem of finding the relative localization between the user and a robot is to fix their relative heading. The distance from the human to the robot can be easily found through a pointing gesture given the fact that the user points at the robot. In certain cases, however, we can assume the heading is already defined by asking the user to always start the interaction from the robot's back. We applied this approach in several experimental studies and it showed its feasibility [28].

4.4 Localization from motion

Another more sophisticated approach, that does not require any additional hardware, is to perform relative localization from the motion of the robot and the synchronized with it the motion of the arm that points at the robot [27].

The system works in the following way. In the first phase, the operator points at the robot they want to interact with, to "select" it (Figure 4.5, left). 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 moves on some trajectory, while

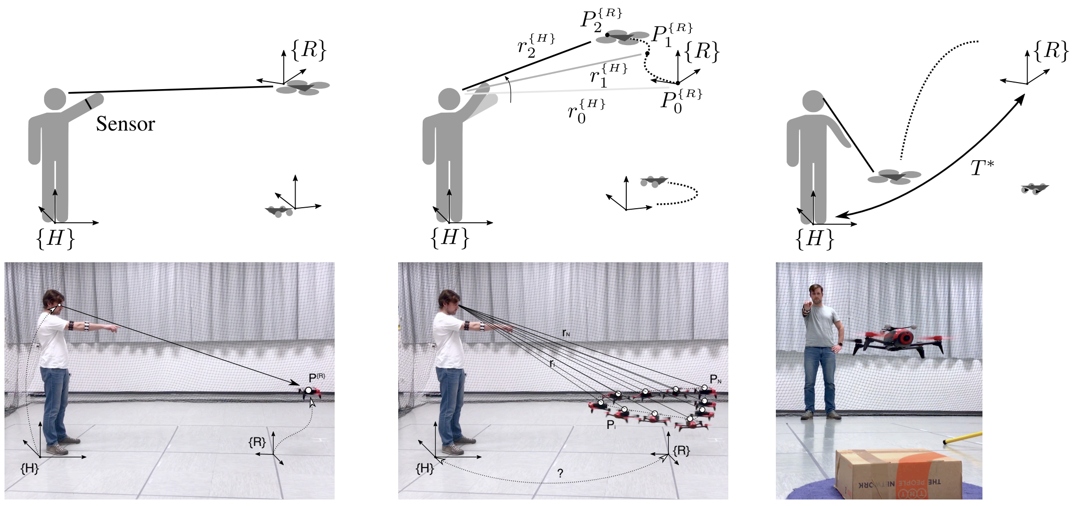


Figure 4.5. (*Left*) The user triggers interaction. (*Center*) All nearby robots start moving along different paths, and the operator keeps pointing at the desired robot. (*Right*) After a few seconds, the system identifies the target robot and reconstructs the transformation between the robot and the user; then the interaction can continue in an application-dependent way, e.g. landing at a precise spot.

the operator points and follows it with their arm (Figure 4.5, center). 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 (Figure 4.5, right). Another feedback marks the end of the second phase. This 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 conducted a series of experiments to assess the accuracy of the proposed method: the user was following the procedure described above to localize the drone which then was commanded by the system to land at the user's origin. By comparing the position the user was standing at with the position the drone was landing at we estimated the average distance between the two locations being less than 40 cm[2].

[2]See the video here: `https://youtu.be/VaQ3aZBf_uE`

4.4.1 Formal definition

We use the same head-finger human pointing model that we previously defined in Chapter 3.

Similarly to the previous definitions, the robot's reference frame $\{R\}$ is an arbitrary and fixed reference frame in which the robot knows its position; in practice, it is useful to assume it to be the robot odometry frame, however, our model does not require it. This frame can equivalently be replaced with the world frame $\{W\}$.

Our goal is to estimate the pose (position and orientation) of the robot's frame $\{R\}$ in the reference frame of the operator $\{H\}$, i.e. the coordinate transformation T_{hr}. To do that 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_{hr}^* between the two frames is estimated using an optimization procedure.

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

$$\mathscr{R} = \{r_1^{\{H\}}, \ldots, r_N^{\{H\}}\},$$

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 $\mathscr{C}$:

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

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 $\mathscr{C}$:

$$\theta\left(T, \mathscr{C}\right) = \frac{1}{N}\sum_{i=1}^{N} \angle(r_i^{\{H\}}, q_i^{\{H\}}) \tag{4.6}$$

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

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

$$T_{hr}^* = \arg\min_T \theta(T, \mathscr{C}) \tag{4.7}$$

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

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 4.4.4, we test the influence of τ on the resulting error.

Constraints on the transformation We express the transformation T_{hr}^* as a composition of translation and rotation, where the translation is defined as a three-dimensional vector $t = \left[t_x, t_y, t_z\right]$ and the rotation as $\gamma = \left[\gamma_x, \gamma_y, \gamma_z\right]$. 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:

$$\rho = \left[t_x, t_y, t_z, \gamma_z\right] \tag{4.8}$$

4.4.2 Implementation

Similar to other experiments in our work, we equipped the operator with a wearable inertial sensor on their forearm (in this particular implementation we used Myo Armband [[77]]). We placed an additional sensor on their upper arm to detect discrete pointing events.

A single sensor on the forearm is fed to head-finger pointing model that was described in Chapter 3, while the data from both sensors serve as an input to the pointing detector [[7]] that is used for triggering the start of the robot motion.

We assume that the robot and the operator are networked and the robot is able to track its 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 $\mathscr{C} = \{(r_i^{\{H\}}, P_i^{\{R\}})\}$. 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. 4.7) is solved with the quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shanno [59] as implemented in the `optimize.minimize` function from the SciPy library [39], 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) [66] 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.

4.4.3 Experimental setup

We implemented our system in an indoor flying arena of approximately 6×6 m size. The arena is equipped with the Optitrack motion capture (MOCAP) system, which 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[3]. We equipped it and the operator with markers to acquire the ground truth for our experiments.

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

[3]This model has been discontinued in 2018.

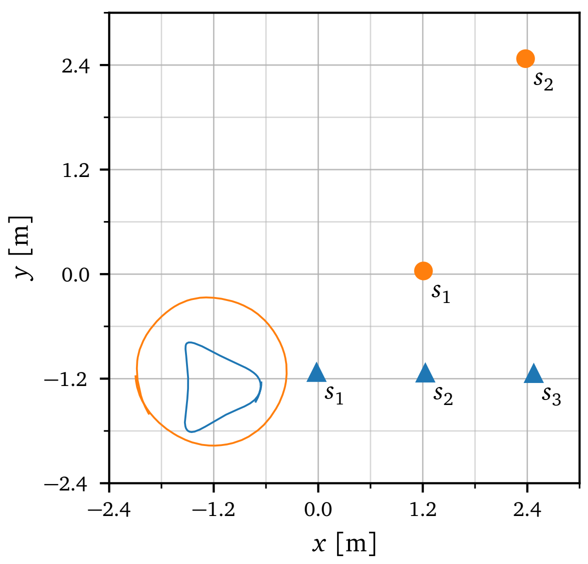


Figure 4.6. Experimental setup for data collection (ground truth): two trajectories (circular and triangular) are represented by thin orange and blue lines, the operator positions for each of the five sessions are represented by the circular and triangular markers. Labels identify the respective sessions in Figure 4.8.

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 $\mathscr{C}$ of pointing rays $r_i^{\{H\}}$ obtained from IMU data and corresponding robot positions $P_i^{\{R\}}$ recorded by the motion capture system; 2) the set of ground truth operator positions $\mathscr{U} = \{U_1^{\text{gt}}, \ldots, U_N^{\text{gt}}\}$ recorded by the motion capture system. Samples of a single loop of the trajectories, and the operator positions in each session are depicted in Figure 4.6.

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

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 capture system, then we perturbed it 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

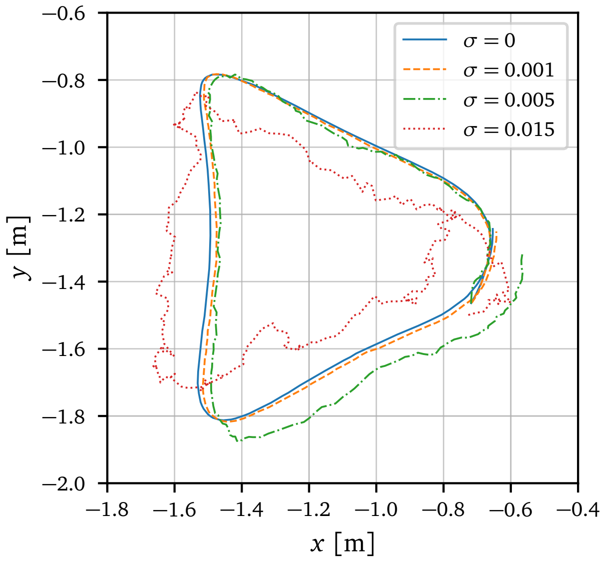


Figure 4.7. Simulated odometry estimates of the drone trajectories: (blue/solid) *perfect* ($\sigma = 0$), i.e. ground truth; (orange/dashed) *good* ($\sigma = 0.001$); (green/dot-dashed) *bad* ($\sigma = 0.005$), (red/dotted) *terrible* ($\sigma = 0.015$).

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 4.7; quantitative data averaged on a large number of simulations are reported in Table 4.1. 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.

	Trajectory length [m]	Trajectory extent [m]	Mean VO error [m] for $\sigma =$				Max VO error [m] for $\sigma =$			
Trajectory duration [s]			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 4.1. 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 $\sigma = \{0, 0.001, 0.005, 0.015\}$.

4.4.4 Quantitative evaluation

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 4.4.3 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 minimization procedure (Eq. 4.7) is then executed using such input data, which returns an estimated transform T^*_{hr}.

Error metric

For a given experiment, we are interested in measuring the error of the estimated transform T^*_{hr} with respect to the true transform between the human and robot frames.

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 employing the motion capture system, since the operator wears a hat with a marker. If the reconstructed transformation T^*_{hr} 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.

Accuracy results

In Figure 4.8 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 4.6) 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 4.1). 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 estimation results.

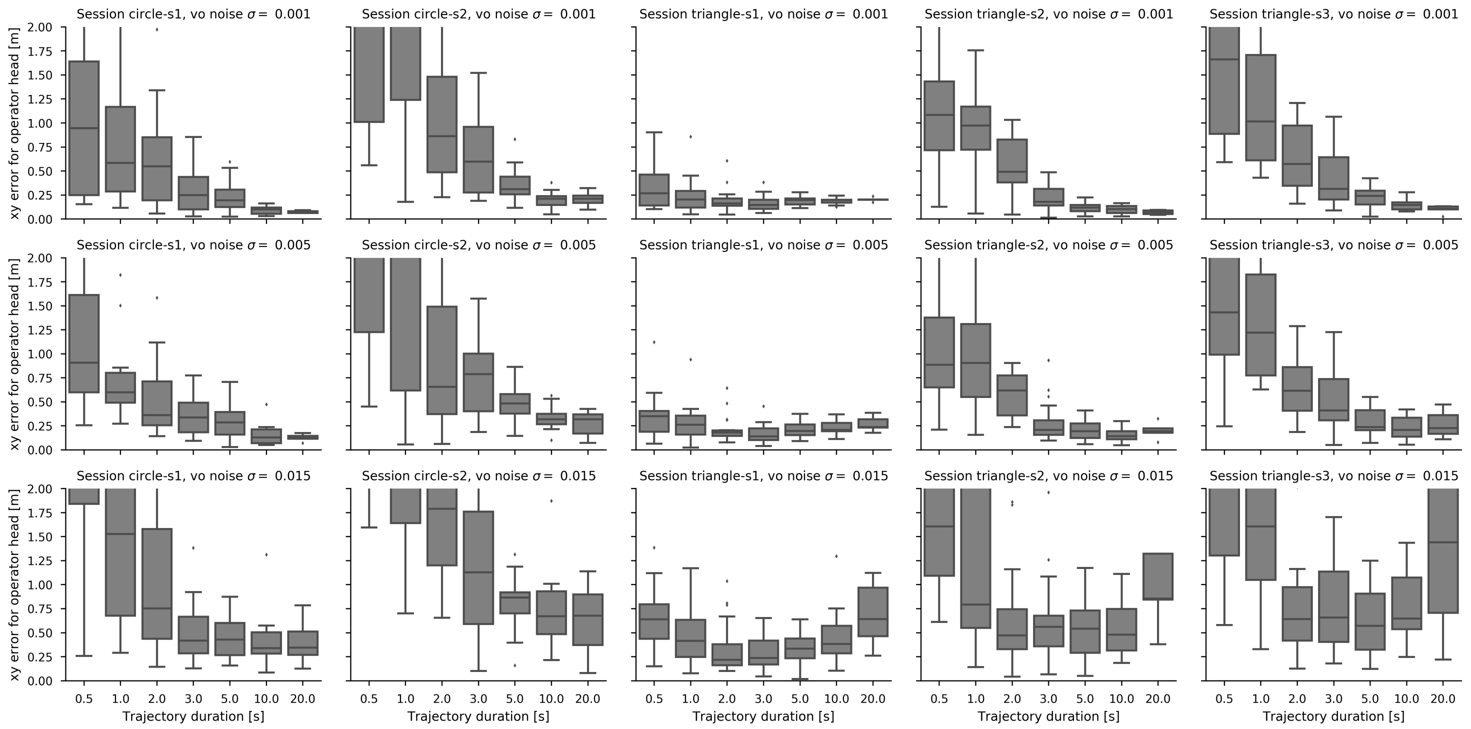


Figure 4.8. We report one row for each value of the VO noise $\sigma \in \{0.001, 0.005, 0.015\}$, and one column for each session. Each plot reports the distribution (box) of the error metric (y axis) as a function of the duration of the trajectory (x axis) over 20 replicas. Results for ideal odometry ($\sigma = 0$) are indistinguishable from for plots with $\sigma = 0.001$ (first row) and are therefore not reported.

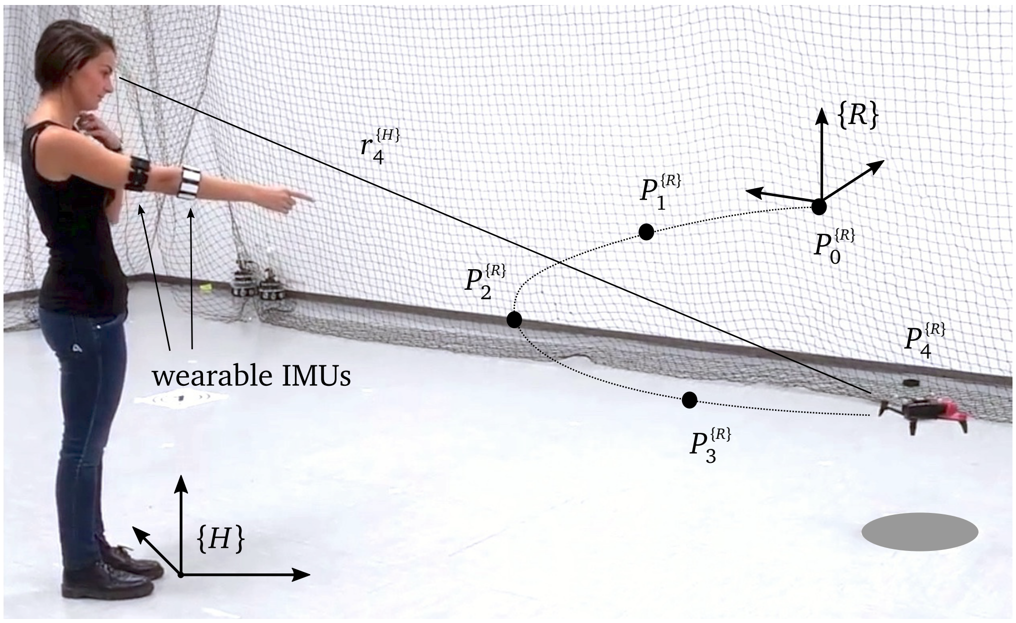


Figure 4.9. Implementation of the proposed method in a real-world experiment.

4.4.5 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 position where the operator was standing, i.e. at the origin of the operator frame $\{H\}$. Figure 4.9 illustrates the implemented system, and the supplementary video[4] demonstrates many consecutive iterations of the procedure.

Figure 4.9 and the supplementary video also demonstrate our relative localization approach integrated with our previously-published system [28], 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 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.

[4] https://youtu.be/VaQ3aZBf_uE

4.4.6 Comparison to other methods

To the best of our knowledge at the time of writing, there were no other similar methods that would use the motion of pointing arm to localize the human relative to a robot. The only similarities we were able to find are related to extrinsic camera calibration and the perspective-n-points (PnP) problems.

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 [34] camera which observes a known calibration pattern (extrinsic camera calibration), and; reconstructing the pose of a known object [46] when observing the image coordinates of some of its points (perspective-n-points). In both cases, image points can be back-projected 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.

4.5 Dynamic human position

Our main motivation to use inertial sensors so far was to perform gesture sensing 'on-board' the human to avoid problems of continuous visual attention from the robot to the user and its limited field of view.

One important limitation of approaches based on inertial sensors, however, is that relative localization has to be performed every time the user moves to another location. That means the user cannot move away once it is localized by the system.

We tackle this problem by equipping the user with a visual-inertial odometry system (VIO, or sometimes simply VO) that estimates its egomotion using both inertial and vision sensors. The vision part allows the system to compensate for a drift induced by the inertial part and additionally provide odometry information. This type of sensors is available off-the-shelf as part of many modern smartphones. Two major mobile operating systems Apple iOS and Google Android are shipped with augmented reality libraries ARKit [1] and ARCore [23] (respectively) that provide phone's odometry information out of the box.

Instead of wearing a bracelet, the user holds a smartphone like a TV remote in the hand of their pointing arm (Figure 4.10). The phone estimates its pose T_{wp} with respect to an arbitrary world frame $\{W\}$. Since the phone's position $p_p^{\{H\}}$ on the user's body is known, we can calculate the coordinate transformation between the human frame $\{H\}$ and the world frame $\{W\}$ using the human pointing model (Section 3.1). Assuming the phone position p_p is equal to the position of the direction point p_d of the pointing model, we can reuse the coordinate transformation chain in Eq. 3.6 and find T_{wh}. First, we convert the direction point $p_d^{\{H\}}$ from human frame $\{H\}$ to world frame $\{W\}$ using the coordinate transformation T_{wh}:

$$p_d^{\{H\}} = T_{hs} T_{sf} \mathbf{0} \tag{4.9}$$

$$p_d^{\{W\}} = T_{wh} p_d^{\{H\}} \tag{4.10}$$

next, assuming $p_p = p_d$:

$$\begin{cases} p_p^{\{W\}} = T_{wp} \mathbf{0} \\ p_p^{\{W\}} = T_{wh} T_{hs} T_{sf} \mathbf{0} \end{cases} \tag{4.11}$$

We can now find the coordinate transformation of the human frame in the world frame:

$$T_{wh} = T_{wp} T_{sf}^{-1} T_{sh}^{-1} \tag{4.12}$$

4.5.1 Implementation

We implemented a system where the user guides a miniature drone Crazyflie 2.0 parallel to the ground using an off-the-shelf smartphone Apple iPhone 8. The phone directly connects to the drone using a wireless Bluetooth LE link and controls its flight functions. To acquire the visual-inertial odometry, i.e. the coordinate transformation T_{wp}, we use the standard augmented reality library ARKit [1] provided by Apple as part of their iPhone's operating system iOS.

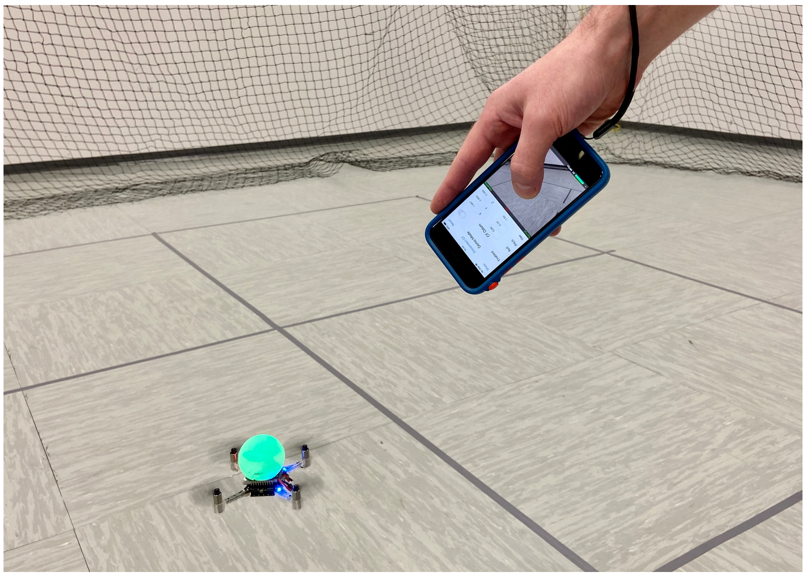


Figure 4.10. A prototype of the system that uses a commodity smartphone (Apple iPhone 8) to dynamically track coordinate transformation between the user and the drone (Crazyflie 2.0).

4.5.2 Public testing

The system has been demonstrated at several public events where the audience had a chance to try our system.

Overall appreciation was very high and people did not have major issues using the system. During these trials we made several interesting observations:

- Many young users attempted to use the phone the same way they would use it in games: e.g. tilting the wrist up to make the drone come closer. Probably the users were trying to control the velocity of the drone instead of directly using pointing gestures to specify a position.
- The visual odometry performs poorly when users start to walk and the phone's camera is facing directly down. Perhaps, this happens because most of the camera view is occupied by walking legs and it makes it difficult for the visual odometry system to choose the right visual features.
- The ability to walk while controlling the drone significantly reduces the arm's fatigue as it allows to keep the arm elevation lower by keeping the drone closer to the user.

4.6 Conclusions

Relative localization is an essential block of pointing-based human-robot interaction, it allows the system to transform pointed positions described in a human reference frame to a frame of reference of the robot. In this chapter, we described several practical approaches to this problem and proposed a novel method that allows one to localize a moving robot with respect to a user that points at it by comparing the robot's trajectory and the user's pointing. This method can also be used to identify a robot being pointed at among several robots (we describe this method in detail in Chapter 5).

An important limitation of the methods based only on inertial sensors is an inability to dynamically track the position of a user, for example, when they walk away from the initial location. We proposed and implemented a method that uses a conventional smartphone and a stock visual-inertial odometry library to solve this task: instead of a bracelet on the wrist, the user holds and points with a smartphone in their hand. Public testing of the system proved the viability of the approach.

Chapter 5

Robot Selection and Identification

Robot selection and identification serve as a tool to engage a user with a robot before they can start the interaction. It allows one to select a robot within a group and make sure that commands issued by the operator are addressed to a specific robot. In the previous chapters, we assumed a robot and a user are already engaged and the interaction is already taking place.

In this chapter, we show various ways to engage a robot and a user. In particular, we demonstrate how our relative localization method presented in Chapter 4 can be efficiently used to solve this problem.

5.1 Point to select

When the robots and the user are co-localized, a simple pointing at a robot allows us to identify it by comparing the coordinates of the target point p_t with coordinates of all the robots in a group. The robot whose position is the closest to p_t and falls within a threshold, e.g., a radius of the robot's circumference, is the one the user is referring to.

However, should we engage the robot immediately after such a check succeeds? Probably not, as on the way to the desired robot, the pointed location p_t may traverse through other robots before it finally reaches the one the user needs.

This problem shows that the selection requires a triggering event. We can generate such an event in multiple ways. One straightforward and robust approach is to use a push-button: the operator points at the desired robot and presses a button held in either hand, at this very moment the system identifies the pointed robot and selects it.

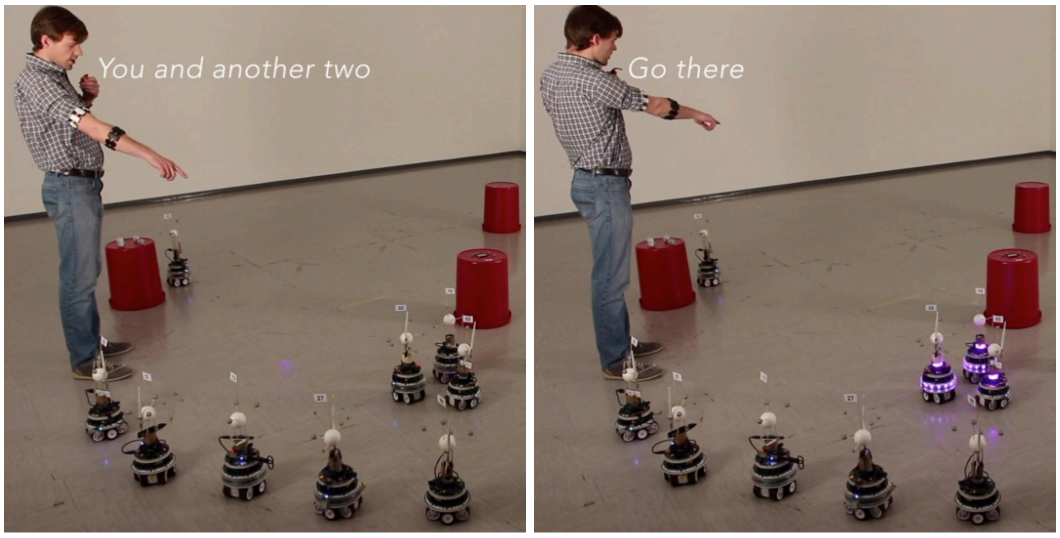


Figure 5.1. Selecting (*left*) and commanding (*right*) robots using natural speech accompanied by pointing gestures.

The main drawback of this approach is that the user needs to hold an additional controller in their hand.

The speech modality allows us to relax this constraint.

5.1.1 Speech

Using natural speech to command robots is a common and very compelling approach. We used speech-to-text cloud services[1] to convert speech utterances to text representations and then matched them against templates to extract the commands and associated data, e.g. robots' numerical identifiers. The system can recognize and implements the following commands [24]:

1. *"You!"* — selects a robot the user is pointing at. These commands can be used consecutively one after another to cherry-pick individual robots and add them to a group;

2. *"You and another N!"*, where N is a positive integer — selects a robot the user is pointing at and another N closest to it robots;

3. *"Number K!"*, where K is a numerical ID of a robot — selects the named robot and implicitly *localizes* the user that points at it;

[1]Google Speech and Microsoft Bing Speech APIs.

4. *"Go there!"* — tells all selected robots to move to a pointed location;

5. *"Go D meters in this direction!"*, where *D* is a positive real number — tells all selected robots to move a specified distance in a pointed direction.

An example of executing the command 2 (selection) and the command 4 (motion) using natural speech and accompanying pointing gestures are shown in Figure 5.1.

Apart from typical problems of speech recognition systems, such as failures in noisy environments and difficulties with non-native accents, the major issue we experienced with this approach was a large latency of cloud-based services. Although the latencies were under 1.5 s, such delays were significantly affecting the user experience and interaction efficiency.

5.1.2 Pointing detection using fixed delays

Another approach that does not require additional handheld controllers is to detect an event of pointing.

One way to achieve this is to trigger selection events when the pointing arm does not move for a fixed time. We implemented this modality using a sliding window technique where we keep the last 1.5 s of the IMU orientation in a first-in-first-out (FIFO) buffer. With every new sample of orientation received from the IMU, we calculate the maximum deviation of all samples in the buffer from its mean value. When the maximum deviation exceeds a certain threshold we trigger an event.

This technique proved to be one of the most efficient, robust, and easy to implement, but it suffers from a similar issue as the speech-based solution—induced latency of interaction.

5.1.3 Pointing detection using neural networks

We as humans clearly distinguish the actual pointing from an abstract gesticulation, but we often understand this from a context, e.g., from a discussion when a pointing gesture is expected. This context may not be accessible to a computer system; however, it turns out that such a detection is possible using an IMU data.

Using our wearable IMU setup, Broggini et al. [7] developed a system that uses *two* IMU sensors placed on the upper arm and forearm to detect pointing events. The system accumulates orientation and acceleration data in a FIFO buffer and then feeds it to a 1D convolutional neural network (CNN) that predicts the probability of the current buffer being a pointing gesture. When the probability exceeds a threshold we trigger an event.

We successfully applied this system in several tasks both for engaging a user with a robot and to trigger particular behaviours, such as to start the localization from motion [27] and to perform landing at a precise spot [30].

Although the solution works well it requires two IMU sensors that might be impractical in many real-life applications.

5.2 Selection by identification

The described above solutions rely on a strong assumption that the robots and the user are already localized, i.e. the coordinate transformation T_{hr} is known. This is not always the case: for example, when there are several robots deployed to perform an autonomous mission and there is a human operator who wants to take the control of one of these robots, he/she stands somewhere near the robots, but his/her location is not known yet.

In this case, the selection and identification can be combined with the localization techniques we described in Chapter 4.

5.2.1 Fiducial markers

Fiducial markers simultaneously provide an identity of the robot and its relative pose with respect to the operator. Once the robot is identified the system can generate the selection event to engage the user with the robot.

5.2.2 Motion based

Our localization from motion method described in Chapter 4 can be also efficiently used to identify pointed-at robots in multi-robot setups.

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 is the one which yields the lowest residual error. A schematic representation of this process was presented in Figure 4.5.

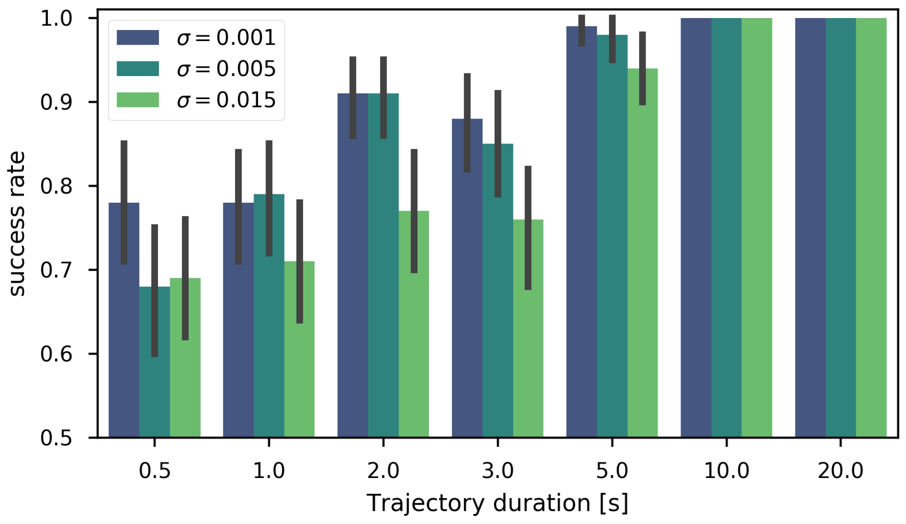


Figure 5.2. The success rate in identifying the correct robot among two (see text). 100 replicas per bar. Error bars represent 90% confidence interval.

Robot identification results

Using the data acquired in Section 4.4 we perform experiments to identify which of the two robots the operator was pointing at. 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 sets 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%). In Figure 5.2 we report the corresponding success rates of the identification.

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.

5.3 Conclusions

In this chapter we discussed several approaches for robot selection and robot identification: using simple voice commands, fiducial markers, and an extension of our localization from motion method described in Chapter 4.

Chapter 6

Efficient Interaction Feedback

Feedback plays a fundamental role in control systems, including such natural systems as the human body; it allows the system to react timely and adapt a control loop to a changing environment or deficiencies of the system itself. By providing adequate feedback to the user of a technical system, we can use vast human abilities to compensate for inaccuracies.

In the literature review in Chapter 2, we have shown that visual feedback, in particular, plays a significant role in the way people use pointing.

In this chapter, we discuss various types of feedback and its implementation applied to our experimental setups. Our key insight and contribution are to use a moving robot itself as visual feedback. Analogous to a mouse cursor on a computer screen, the user adjusts pointing according to the actual position of the robot and therefore, can compensate for errors. Since it is not practical to use such feedback with slow-moving robots, e.g., ground vehicles, we propose another solution based on a laser pan-tilt unit that shines a dot on the floor to provide a similar kind of feedback. We evaluate these solutions in user studies.

6.1 Voice and lights

In the course of our research, we attempted to use various types of feedback to communicate the state of the system to the user. We started with a text-to-speech interface and colored lights in [24] using ground robots. The system would pronounce the ID of a robot when it is selected by the pointing gesture it would light up with purple color as well. While the speech feedback improved the user awareness of the system state, compared to the visual feedback it turned to be inefficient—the long utterances like “Robot number 12 is selected” bear very little information for the user, but take a lot of time to pronounce, i.e. the confirmation that the pointed robot was selected can be done much faster by lighting up its LED with a certain color. Also, very quickly it became obvious that audio feedback has limited applicability to the drones because of the high background audio noise produced by the rotors of the robot. On the other hand, the visual feedback with LED lights shown to be extremely efficient because it could be switched on almost instantaneously. This allows us to communicate the system state to the user immediately and therefore allow them to correct their action in a timely manner[1].

We further exploited these ideas and build a custom LED-ball that we fitted on a lightweight drone[2] [31]. Using the LED allows the system to report various states and stages of the interaction: during the localization phase the LED is turned blue; once the user is in control the LED switches to a blinking green color; when the user about to disengage from the drone the LED dims from yellow to off state; once the drone is on its own again, the LED turns back to blue.

6.2 Laser pointer paired with camera

In the system with fiducial markers (see Section 4.2) we used a narrow lens on the camera and therefore it was difficult for the user to aim at the marker if they started far away from it. In this case, the feedback was given only at the time the user successfully aimed at the marker (the robot would light up). We paired the camera with the laser pointer and it proved to be a very efficient yet simple solution.

6.3 Haptic feedback

Last, but not least important, we implemented haptic feedback through the armbands. We use vibration to notify the user of various events. For example, the armband would vibrate every second to count down the landing timeout [28].

Recently we have also implemented haptic feedback in a smartphone-based pointing interface. Modern phones have very few hardware buttons and those are usually dedicated to specific operating system functions, such as changing the sound volume, answering a call, etc., and are not available to developers. On the other hand, touch screens allow one to build versatile graphical user interfaces;

[1]See the video section “Fast tracking and selection...” (1m 41s) at: `https://youtu.be/FWMCxARQYhY`

[2]See the video demonstration here: `https://youtu.be/yafy-HZMk_U`

however, the problem with touch interfaces is that they do not provide any tactile feedback and make it practically impossible for the user to know if a software button on the screen was actually pressed or not. It is especially true for our interface where the user holds a phone in the hand of an outstretched arm—the phone's screen is parallel to the arm and at a shallow angle with respect to the line of sight.

To solve this problem we use the 3D Touch feature of the Apple iPhone 8 and sharp vibrations provided by Apple Taptic Engine. 3D Touch technology is Apple's brand for a force-sensing touch screen technology. It allows one to measure the relative force with which the user presses on the screen. The Taptic Engine is a linear vibration actuator that is capable of generating very short and strong vibrations in response to user actions. We use these technologies to simulate the feel of a real hardware button, such that it clicks only when the pressing force exceeds a certain threshold. That allowed us to implement a software button that has four states:

1. *Initial* — the user is not interacting with the button;

2. *Touched* — the user's finger is on the button, but it is not pressed;

3. *Pressed* — the button is pressed with force and is being held down;

4. *Released* — the button was released but the user's finger is still touching it;

When the button transition from the state *Touched* to *Pressed* and from *Pressed* to *Released*, the user feels a click that tells them that the system has changed its state. What is more, the software implementation of such feature allows us to block the button to notify the user the state change is not possible or not allowed, i.e. the button will not give any haptic feedback in this case and the user will know there is something wrong, e.g. the drone is not connected to the phone.

The button with these states allows the user to engage and interact with the drone without ever lifting their finger from the phone's screen.

We implemented an iPhone application to control a nano quadcopter Crazyflie 2.0. The user holds the phone as a TV remote with their palm under the phone and a thumb resting on top of the screen. To start the interaction the user points at the drone with the phone, presses the screen with force and holds the finger, the phone generates the haptic click to show the command has been sent to the drone. The color of the LED ball on the drone changes its color to blue to signify it is taking off. Once in the air, the LED color changes to green showing that the user is now in control. From that moment the drone flies wherever the user points to. When the user relaxes the hold, the button releases and generates the click, the drone's LED switches back to blue showing it is now in hovering mode and does not follow the user's pointing anymore. The user now can freely move around without affecting the drone's position (but have to keep touching the screen). To engage with the drone again the user has to point at the flying drone and press on the screen with force, the LED color will change back to green and the user can start pointing to drive the drone. Whenever the user feels unsafe or wants to land the drone they just have to lift their finger from the screen.

We have tested these features “in the wild” with lay users. Contrary to our expectations it was sometimes quite difficult to convey how the virtual button interface works. The users had to learn it by trial and error; however, once they were accustomed to the interface they could control the drone quite efficiently.

Unfortunately, in the latest versions of Apple phones, the 3D Touch feature was discontinued and we will need to find an alternative interaction pattern.

6.4 Feedback from motion

Although the idea of using the motion of a robot itself as visual feedback is somewhat obvious, it has a significant impact on the performance especially when the pointing reconstruction itself lacks precision.

Flying robots, such as quadcopters, are quite agile and thus suitable for this type of feedback. In our work [28; 30] we implemented two types of feedback: *discrete* to signify events and *continuous* feedback to show where the system thinks the user is pointing at. We used a “jump” for a few centimeters up (discrete feedback) to confirm that the drone has been selected and that it is the user who controls it now. Once the control has been transferred to the user the drone would follow their pointing gestures in real-time (continuous feedback), such that the user can immediately see whether there are any discrepancies between the intended control and the actual state of the robot. These two feedback techniques proved to be very efficient. The live feedback proved especially useful for the cases of accumulated IMU errors.

The simplified approach used to reconstruct the pointing ray, and inaccuracies in the estimation of T^*, cause errors in the reconstructed point: in practice, these errors are adjusted for by the operator as long as we provide real-time visual feedback.

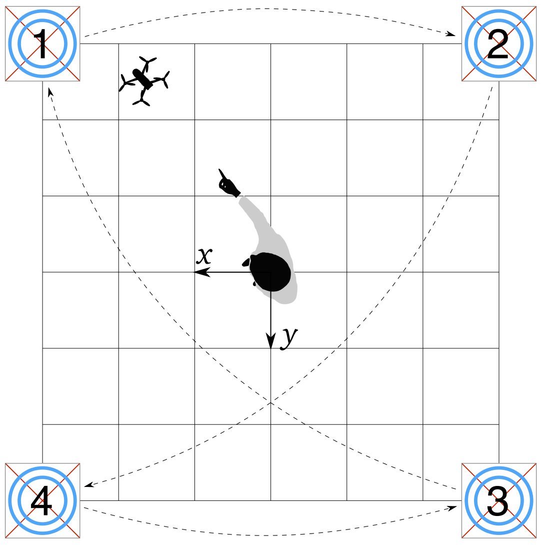


Figure 6.1. The scheme of the experimental environment: targets are fixed at the corners of a square with a 3.6 m edge (one tile is 0.6 m). The human position is fixed at the center. The sequence is defined by arrows and starts at target 1.

6.4.1 Experiments with fast robots

To assess the viability of this approach we set up an experiment where the users were required to land a quadcopter (Parrot Bebop 2 [60]) at a given location using two different interfaces: pointing gestures and a regular joystick.

The scheme of the experimental environment is depicted in Figure 6.1 and represents a flying arena with four predefined targets. The targets are placed at the corners of a square with an edge of 3.6 m. The sequence of the targets is predefined and is shown on the scheme with the arrows. The subjects were asked to stay in the middle of the arena, however, they were allowed to step aside to avoid collisions with the drone.

The experimental environment represents a flying arena with four predefined targets. The targets are placed at the corners of a square with an edge of 3.6 m and numbered in a clockwise order. The sequence of the targets is predefined as 1–2–4–3–1, i.e. edge segments alternate with diagonal ones. The subjects were asked to stay in the middle of the arena, however, they were allowed to step aside to avoid collisions with the drone.

The arena is equipped with the Optitrack motion capture system that provides precise information about the drone's position. This information is used, both, to control the safety margins and to implement autonomous flights. Note, that in general the robot is localized in an arbitrary frame, e.g. in its odometry frame or, as in our case, in the motion capture frame; however, the location of the operator with respect to robot's frame is not known until the "Robot identification and pose reconstruction" step has taken place.

The drone closed-loop controller is built around `bebop_autonomy`[3] ROS-package and accepts velocity and 6D-pose commands. While the joystick interface generates velocity commands, the pointing interface supplies the pose commands.

Subjects

Five people between 25 and 36 years old have volunteered to participate in the experiment. The majority have reported either no experience in piloting RC-vehicles or a little experience ("tried it a few times").

We conducted two sessions per person: with the joystick and with the pointing-based interface, each consisting of three runs. Each run starts and ends at target 1 and therefore provides four segments. This totals to 12 segments per person or 120 target-to-target segments for all the participants for both interfaces. Three subjects started with the pointing interface and the rest with the joystick.

Before each experimental session, individually for each subject, we conducted a training session with the same interface that they were given later. Each training session consisted of two runs.

Two training sessions plus two experimental sessions took approximately 40 minutes per person, including all the explanations, service times (e.g. replacing the drone's battery), etc.

Experimental sequence

At the beginning of each session, the drone is placed at target 1. Once the supervisor starts the session the drone takes off automatically. Once it is airborne and stable, it aligns itself with the first target of the segment and turns its back to the user, such that the user's controls, both, for the joystick and for the gestures, are aligned, meaning, e.g., that pushing joystick forward would drive the drone away from the user, in the direction they look to. This way we ensure that the landing errors are not accumulated over the course of the experiment and that all the subjects start in equal conditions, both, when controlling the drone with pointing gestures and with a joystick.

From this moment the drone is ready to interact with the user. Once the user lands the drone, it will perform the automatic take-off procedure with a delay of

[3] `http://wiki.ros.org/bebop_autonomy`

5 seconds from the moment it has been landed. This procedure repeats until the list of targets is exhausted.

Since the interaction pattern with the joystick and the gesture-based interface are slightly different we report them separately.

Pointing-based interface The user points at the floor beneath the drone, i.e. at the crosshair of the target, to select it[4]. The Myo on the user's arm vibrates and the drone 'jumps' to signify that it is now being controlled by the user. At the same time, the control station gives voice feedback through a loudspeaker, telling the user the next target they should bring the drone to. Immediately after that, the drone starts to continuously track a newly given location. Once the user is ready to land the drone, they have to maintain the pointed location for approximately half a second. The system starts to count down and makes the upper arm Myo to vibrate every second. The user has about $3s$ to change their mind. To adjust the landing position they just have to start moving the arm away and the countdown will be canceled.

Joystick The behavior of the system in this mode is similar, however, the drone is selected automatically and performs the same 'jump' motion as in pointing-based interaction mode, meanwhile, the control station gives voice feedback with the next target number. The user then moves the drone to the required target and presses the button on the joypad to land the drone.

Performance Metrics

We define a set of performance metrics to compare the performance of two interfaces:

- *Landing error*. Euclidean distance between the requested landing target P_i and the actual position p_a the drone have landed to: $\varepsilon = |p_a - P_i|$.
- *Time to target*. Time that passed from the moment t_0 the drone has moved $20cm$ away from its starting pose till the moment t_1 the user gave command to land it: $\tau = t_1 - t_0$.
- *Trajectory length*. Line integral of the trajectory curve from the start position P_{i-1} to the actual landing position p_a: $\rho = \sum_{k=1}^{N} ||p_{k+1} - p_k||$, where

[4]Although the drone can be selected in the air, this brings an additional error to the relative localization and may deteriorate user experience.

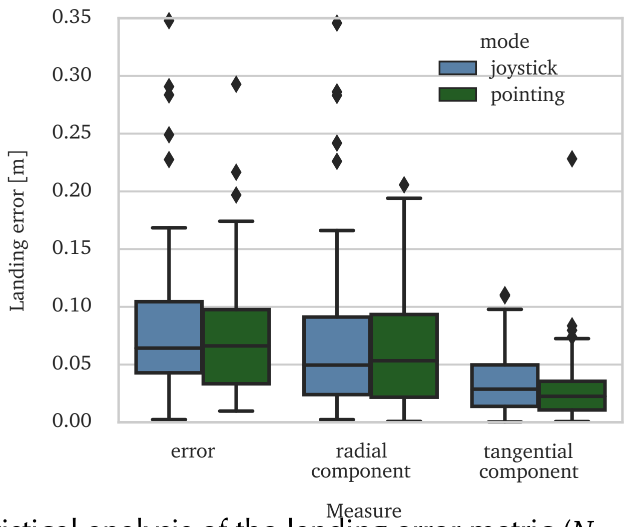


Figure 6.2. Statistical analysis of the landing error metric ($N = 60$ for each interface).

$p_k \in \{p_1, ...p_N\}$ is a set of all the acquired positions of the drone between P_{i-1} and p_a.

We collect the data with a standard ROS tool `rosbag` and analyze it offline.

Results

Landing Error Figure 6.2 reports the landing error metric. We observe no statistically significant difference between the results with the two interfaces; we separately report the error (left pair) and its decomposition in a radial (central pair) and tangential (right pair) component with respect to the user's position. It's interesting to note how the radial component dominates the error in both interfaces, which is expected since the radial component corresponds to the depth direction from the user's perspective: along this direction, the quadrotor's misalignment with respect to the target is much more difficult to assess visually; the landing error is dominated by a perception (rather than control) issue.

Time to Target Figure 6.4 (*left*) reports the time to target metric, separately for each of the four segments. We observe that the pointing interface yields a better average performance; the difference is statistically significant under the Student's t-test ($p < 0.01$) for 3 of the 4 segments and the mean over all segments.

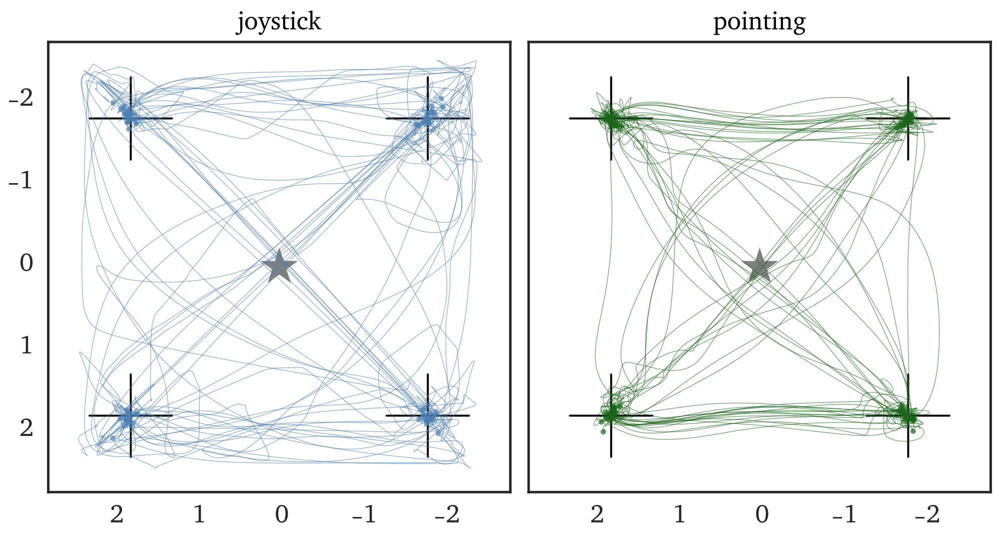


Figure 6.3. Visualization of all trajectories ($N = 120$) flown with joystick (*left*) and pointing (*right*) interfaces. Fixed human position is denoted by a star. Targets 1–4 are denoted by a thin cross (clockwise from top-left).

Trajectory Length Figure 6.4 (*right*) reports the trajectory length metric, separately for each of the four segments. We observe that the pointing interface consistently yields shorter trajectories than the joystick interface; the difference is statistically significant under the Student's t-test ($p < 0.01$) for all four segments. On average over all segments, the joystick interface yields a 66% longer trajectory than the straight distance between the targets; the pointing interface yields a 33% longer trajectory.

One can also observe this phenomenon in Figures 6.3 and 6.5: the trajectories flown with the pointing interface are smoother and more direct and tend to converge faster to the target (Figure 6.6).

6.4.2 Variants for slow robots

Because of relatively low speeds at which ground robots operate, it is impractical to encode their state through the change in their trajectories. We adapt the approach of robot motion feedback for slow robots by implementing a variant in which a laser dot is projected by a robot-mounted laser turret (ScorpionX MX-64 Robot Turret, InterbotiX Labs). The position of the dot on the ground plane is precisely controlled in the robot frame $\{R\}$. This dot takes the role of "a fast-moving robot" during the whole interaction.

While available for interaction, the robot shines the laser on the floor in its vicinity (Figure 6.7), continuously tracing some pattern (such as a circle or 8-

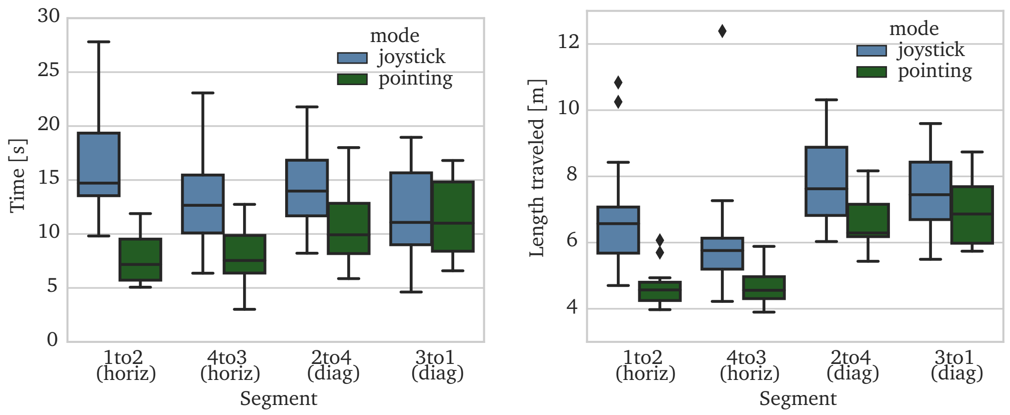


Figure 6.4. Statistical analysis of the time to target (*left*) and trajectory length metrics (*right*) ($N = 15$ for each interface and segment).

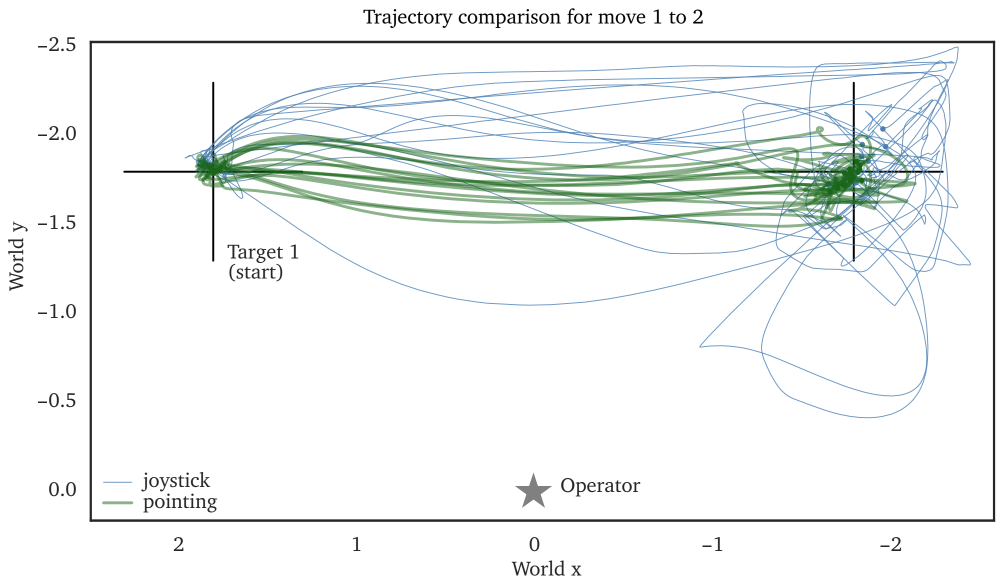


Figure 6.5. Comparison of all trajectories flown from target 1 (left) to target 2 (right) for joystick (blue, $N = 15$) and pointing (green, $N = 15$) interfaces.

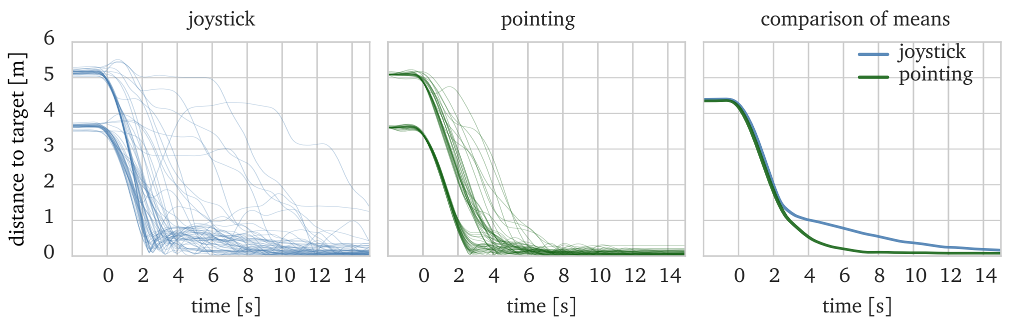


Figure 6.6. Evolution in time of the distance to the target, for each trajectory flown: (*left*) joystick interface ($N = 60$), (*center*) pointing interface ($N = 60$), (*right*) average over all trajectories for each interface.

shape). To take control, the operator should follow the laser dot that the robot is projecting instead of at the robot itself. For triggering, relative localization, and robot identification, the position of the laser dot takes the role of $P^{\{R\}}$ for the specific robot. During real-time feedback, the laser is projected on the reconstructed point that the user is indicating.

Experiments

We assume that relative localization is obtained exactly and focus instead on measuring the impact of real-time feedback on the reconstructed pointing location.

In all the experiments we collect the data with a standard ROS tool `rosbag` and analyze it offline.

Impact of real-time feedback on pointing accuracy In order to evaluate the impact of visual feedback on the accuracy of pointed locations, we implemented an experiment using the laser turret mounted on a fixed platform, that would provide the feedback accurately. Ten people volunteered to participate in this study. Each participant wears the IMU-equipped bracelet and has their body measurements taken to set up the parameters of the system.

Three targets are laid out at known positions as shown in Figure 6.8: with the user standing at a known relative position to the turret. The user's heading is fixed with respect to the turret.

The experiment proceeds using the following sequence, which is advanced by audio prompts played at known times: 1) the user is asked to relax their pointing arm and wait for an audio signal; 2) after 5 s the system plays a beep and the

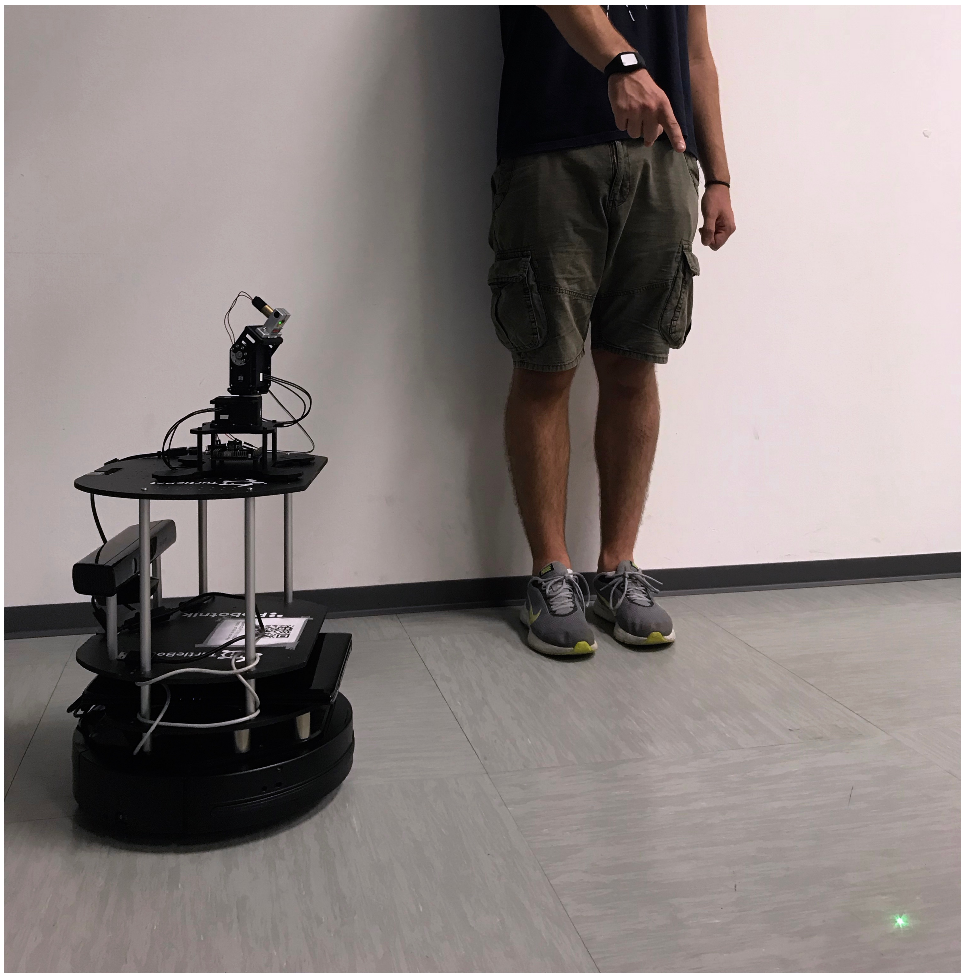


Figure 6.7. Interaction using pointing gestures with a slow-moving robot equipped with a laser turret for real-time visual feedback.

user is asked to point at the first target and hold their arm still; during this time, no feedback is given; 3) after another 5 s and a beep, the user can relax the arm and wait for another command; 4) after next 5 s, the user is asked to point to the same target again; now, however, the laser provides real-time feedback by shining at the pointed location. The procedure is then repeated for another two targets.

Results

We recorded and analyzed the pointing locations and the timings of the audio prompts for each participant. The collated trajectories of the pointed locations

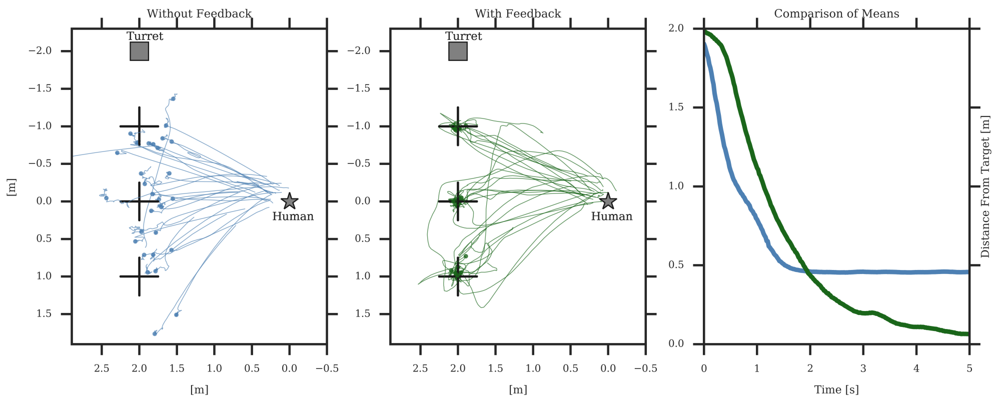


Figure 6.8. Effects of real-time visual feedback on pointing accuracy.

and the evolution of the distance to target are visualized in Figure 6.8. The data shows that, without feedback, users quickly reach an average distance from the target of 0.5 m but do not improve any further; this is expected as the system has intrinsic inaccuracies (for example in the reconstruction of pointing rays $r^{\{H\}}$) which the user is unable to see and correct. When the feedback is provided, distance decreases to almost 0 within 5 seconds.

This demonstrates that real-time feedback (provided with a laser or with the robot's own position) is a key component in our approach.

6.5 Conclusions

In this chapter, we considered and implemented various types of feedback necessary for efficient human-in-the-loop control. We used voice, lights, vibrations, and motion of robots themselves to indicate the current state of the system. We found that although voice messages could provide information-rich feedback they introduce a significant delay to the interaction process. On the contrary, using colored lights installed on robots a state of the system can be communicated to the user almost instantaneously.

Our main contribution is a systematic study on the effects of visual feedback on pointing accuracy and its influence on the time necessary to point at a target. We found that without the visual feedback subjects tend to conclude the pointing faster but with a larger static error; being provided with the continuous visual feedback the resulting pointing error converges to almost zero but takes more time. These findings show that an imprecise pointing estimation and simplified human pointing models can still be efficiently used for precise pointing tasks when a user is provided with continuous visual feedback. We implemented two types of continuous visual feedback: for fast-moving (flying) and slow-moving (ground) robots. The fast-moving robots provide feedback with their own motion, i.e. the robot continuously moves to an estimated pointed location and therefore the user can timely adjust their pointing to compensate for any errors. In the case of slow-moving robots, we provide similar feedback with a dot projected by a pan-tilt laser unit installed on the robot.

Chapter 7

Conclusions

In this work, we have developed an interface for efficient human-robot interaction that is based on pointing gestures—a natural way to communicate directions, locations, and objects in the user's proximity to flying and ground robots. What makes pointing gestures especially powerful, as compared to other interfaces, is the way they define targets: pointed directions and locations are always relative to the user and therefore the cognitive effort to issue commands is minimal.

Our approach is based on inertial wearable sensors that in the past few years have become a commodity and now are present in the vast majority of smartwatches, fitness trackers, and smartphones. An important property of such sensors is that data acquisition is performed directly on a human body and therefore does not depend on the robot's sensing capabilities, does not degrade with distance, and does not require the robot's continuous visual attention.

In the course of our work, we have identified and addressed the following research questions that arose in the context of efficient human-robot interaction using pointing gestures.

Human pointing model A human pointing model maps the posture of the human body to a pointed direction that is expressed as a ray. Although human pointing is a complex concept that relies on many parameters such as interaction context, e.g. distance to indicated objects and their size, physical peculiarities of a user (eye and hand dominance, body proportions), it can be approximated with simplified models and still be efficiently used in many practical tasks. In Chapter 2 we reported state of the art human pointing models and practical applications of these models in HRI. The eye-finger model we have adopted in this work assumes the user is standing upright and points with a straight arm, their shoulder and head are vertically aligned and are at known heights. In various experiments, we have shown that a single inertial sensor on the user's wrist or in the hand of their pointing arm is sufficient to control robots on complex 3D trajectories.

Estimation of pointed locations The act of pointing by itself does not uniquely identify a pointed location. Using a human pointing model one can define a ray in 3D space that originates at the operator's body and extends to infinity along a given 3D direction: the desired target position might lie anywhere on the pointing ray. When the desired target location lies on or close to a surface of the environment, one can simply intersect the pointing ray with that surface to define the 3D position. For example, when controlling a ground robot on flat terrain, one can easily find a target location by intersecting the pointing ray with the ground plane. However, this is not possible if one needs to define a target position in a free 3D space.

We developed a method that solves this problem and allows one to also define targets mid-air. This modality is especially useful to control flying robots whose waypoints may not be bound to the shape of a work environment. The method relies on a set of virtual workspace surfaces that constrain the motion of a robot in 3D space and essentially reduce the control space to a 2D case. To aid the legibility of the system's state, each virtual surface is initialized to pass through a current robot's position: knowing the shape of the surface, the user can easily visualize and predict the robot's future position. To switch between virtual surfaces, we use an additional discrete input—a push-button. We found that a vertical cylinder and a horizontal plane are the most appropriate shapes that allow one to define target positions in the entire 3D space. In Chapter 3 we described the method in detail and reported its experimental validation. The results show that the proposed interaction modality compares favorably with a baseline joystick interface.

Relative localization Because pointed locations are defined in the frame of reference of the human they have to be eventually converted to the frame of reference of the robot. To do that one should know a coordinate transformation between the two frames. In Chapter 4 we have proposed a method that finds such a transformation based on the comparison of the mutual motion of the human and the robot: the user points at the moving robot and keeps following it for a few seconds; the system accumulates a list of recent robot positions defined in its odometry frame and corresponding pointing rays defined in the human's frame;

using an optimization procedure we then find a coordinate transformation that describes the match between robot's positions and pointing rays the best. The experimental evaluation of the method shows that pointing at a moving robot for just 3 seconds is enough to determine the relative coordinate transformation between the user and the robot.

An important drawback of IMU-based pointing reconstruction is an inability to track the user's position. In Chapter 4 we have shown that this limitation can be solved by replacing a wearable inertial sensor on the user's wrist with a smartphone equipped with a camera and a visual-inertial odometry system.

Robot selection and identification To start the interaction with a robot the system needs to know which robot the user is referring to and then engage with it. In Chapter 5 we proposed to use the relative localization method described in Chapter 4 to solve this task: we trigger the robot selection whenever the localization residual error drops below an arbitrary threshold. This method also naturally extends to multi-user multi-robot setups: the system keeps track of all the pointing users and moving robots and matches the operator–robot pairs the motion of which yields minimum residual errors. We discussed this approach and reported its experimental evaluation in Chapter 5.

Efficient interaction feedback Feedback plays a fundamental role in control systems and especially crucial for systems with a human in the loop. We have studied and implemented various feedback types to facilitate a more intuitive and efficient human-robot interaction. In Chapter 6 we reported results of a user study that show the influence of visual feedback on the human pointing accuracy and the time necessary for pointing to converge. We found that continuous visual feedback allows one to point more accurately and compensate for inaccuracies induced by the simplified human pointing model and the drift of wearable inertial sensors, but we also found that such feedback results in longer times necessary to acquire a target. We implemented two types of continuous visual feedback: (1) for fast-moving robots, such as quadrotors, we use the robot's position itself to indicate the pointed location; (2) for slow-moving ground robots we substitute the robot motion with a laser dot that continuously follows the pointed location.

7.1 Current limitations and future work

The present work is based on several important assumptions that we discuss below. We also propose how current limitations can be tackled in future work.

Human pointing model In Chapter 3 we described an eye-finger pointing model that we used in this work. Our implementation of this model does not account for such important parameters as relative orientations between the user's torso, head, and pointing arm. The model also assumes the shoulder is aligned with the line of sight, as if the user is always looking to the side; thus, when it is not the case, the lateral pointing error may significantly increase.

As we have seen in many experiments where users were provided with live feedback through the robot's own motion, compensating for such an error is not a big problem. However, the live feedback means the user is always attached to the robot and that results in fatigue that develops within 5–10 minutes, especially when the robot is far away and the arm's elevation is high. One solution to this problem would be to use *discrete pointing gestures* instead of continuous gestures: the user needs to indicate only the final target location for the robot to go instead of specifying an entire trajectory.

Another limitation of the present pointing model is that it assumes the user is always standing up while interacting with a robot. This may be undesirable in household environments, for example, when the user sits on a couch and wants to command their vacuum cleaning robot.

These problems could be addressed by using a more sophisticated kinematics model of the human body; however, this solution requires more sensors to be worn by the user which inevitably will induce more errors. In this case, advanced calibration techniques proposed by Droeschel et al. [15] and Mayer et al. [49] can be used.

Instrumented human The pipeline we have described in this work relies entirely on an instrumented human, i.e. the user needs to wear inertial sensors that capture kinematic parameters of their body. On one hand, that means a passer-by could not engage with the robot; on the other hand, that allows the system to have refined personal access capabilities. The devices we have used throughout this research are wireless sensors based on a commodity Bluetooth Low-Energy (Bluetooth 4.0) technology with industry-proven authentication and security capabilities. Uninstrumented human-robot interaction is typically based on computer vision systems that are at the moment cannot be used for user authentication from a distance.

Both visual and inertial sensing have their benefits for pointing-based interaction. Although in this work, we left out the comparison of the two sensing methods, it would be interesting to see their differences. An experimental study for the robot-centric vision system can be implemented using the same experimental setup that we used in our paper on quadrotor landing [30], where the user was tasked to move the drone between predetermined positions on the floor. We then can compare the target location with the actual landing position to estimate the pointing accuracy of the vision-based interface. However, the peculiarities of the visual sensing may not allow implementing the same continuous control as used with the inertia-based interface, and thus, will require to use discrete pointing instead.

Pointing in 3D The proposed 3D pointing method with virtual workspace surfaces (Chapter 3) relies on a strong assumption of continuous real time feedback. While this assumption holds for fast-moving robots in close proximity, this solution unlikely to be convenient for the robots that are far away or that move slowly. For example, one may need to position a payload using a tower crane at a construction site. In this case, relying on continuous visual feedback would be impractical as it will quickly result in arm fatigue.

A possible solution is, again, to use discrete pointing gestures. Alternatively, one may use pointing gestures in conjunction with an augmented reality head-up display and a model predictive control system: the user will see a predicted position of the payload that accounts for its dynamics but compensates the time necessary to reach the target location. Once the user is satisfied with the target position, they can commit the motion command.

Human studies Throughout this work, we have conducted multiple experimental human studies that compare the performance of a baseline control interface (joystick) with our pointing interface. Although the results show certain advantages of pointing over joystick control, the differences are not as dramatic as one would intuitively expect. We see two reasons that could have led to these results. First, the task we chose for our experimental studies, i.e., a precise target acquisition, does not represent the type of task a pointing-based interaction would benefit the most. Second, the number of subjects who participated in the studies was not sufficient to highlight the advantages of pointing over the joystick interface. Future work should approach these issues by a more careful experimental design and more extensive studies on a larger population.

7.2 Beyond robotics

In this work, we have shown how pointing gestures can be efficiently used to control robots. Since pointing is a very generic interface it could also be effi-

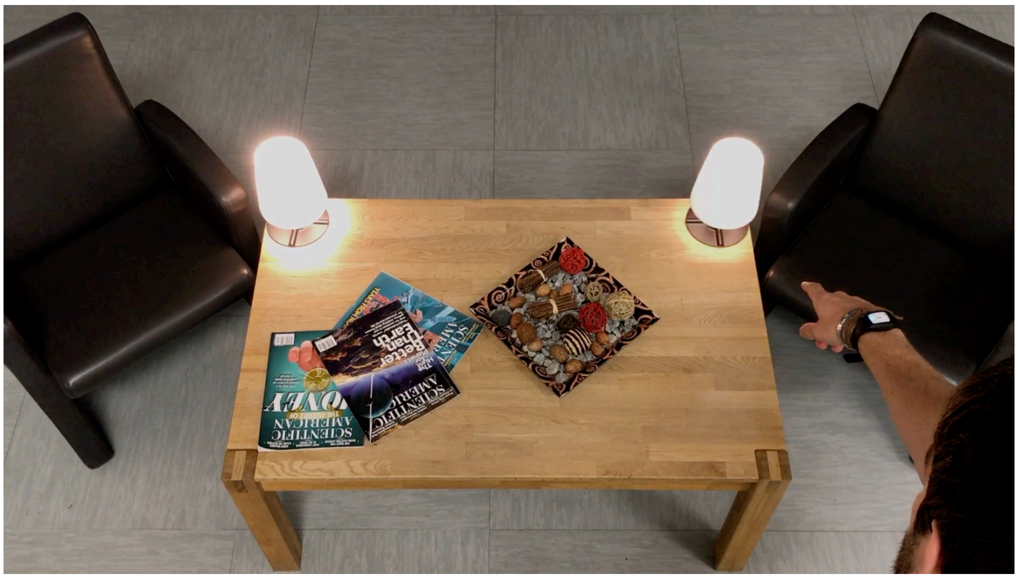


Figure 7.1. A prototype of the pointing-based smart lights interaction system using off-the-shelf IKEA Tradfri smart lights [37].

ciently used to control other devices. We explored a possibility to use pointing to interact with smart lights—a type of lights (bulbs) that can be switched on/off or dimmed remotely via a wireless network.

A typical way to interact with smart lights requires the user to assign names to all available devices and then use a given name to access a light's functionality with either a mobile application or with a voice assistants like Amazon Alexa.

We built a prototype of a system that allows the user to refer to smart lights with simple pointing gestures. The interaction proceeds in the following way: the user points at the light they want to control, the light turns to 30% brightness showing it is being pointed at; if the user keeps pointing at the light for a few seconds, the light switches to the maximum brightness and latches in this state; otherwise, if the user points away before the light latches the state, the light simply switches off. A prototype of the system is shown in Figure 7.1.

An important advantage of using pointing to control smart appliances is that one neither needs to remember devices' names nor to find which switch controls each appliance: the device is the one the user is pointing at.

Similar to robotics applications, pointing gestures naturally combine with speech commands. For example, using modern smart home voice assistants, one could simply point at a light and say: "Alexa, dim *that* light to 50 per cent!"

Pointing can also be conveniently used to set the level of smart blinds, such that one could say: "Alexa, lower *that* blind until *here*!" instead of "Alexa, set kitchen blind 55 per cent closed!”

Appendix A

Publications

Estimation of pointed locations We first discussed the pointing reconstruction using inertial sensors in a workshop paper [A1] where we suggested to use two sensors: one on the upper arm and another on the forearm. To estimate pointed locations we use the *shoulder-finger* pointing model and intersect the pointing ray with the ground plane. We further elaborate on this setup in a conference paper [A2] and assess the accuracy of the IMU-based arm pose reconstruction using a motion capture system. Further, we simplify the setup and use a single inertial sensor to estimate pointed locations (using the *head-finger* model), where we assume the user is always pointing with a straight arm [A3; A4]. In a conference paper [A5] we use a similar setup but with a more compact sensor placed at the user's wrist. Finally, in [A6] we consider a problem of specifying locations in *free 3D space* using virtual workspace surfaces that the user can switch between on demand.

Relative localization In a conference paper [A2] we propose a relative localization method based on fiducial markers. We further study the applicability of this method in a workshop paper [A7] where we experimentally quantify localization errors. In a conference paper [A8] we propose, implement, and validate a novel localization method based on a comparison of the robot's motion and the motion of the user's arm that points at that robot. We further use this method as an integral part of our human-robot interaction pipeline [A5].

Robot selection and identification In a workshop paper [A1] we implemented a system where a single robot or a group of robots is selected using a pointing gesture and an accompanying voice command, we use a trivial coordinate comparison to identify the pointed at robots. In a conference paper [A2] we use unique fiducial markers on each robot for their identification and a voice interface to trigger selection. Our main contribution to a robot identification and selection task is presented in a conference paper [A8] where we use a residual localization error to identify which robot the user was pointing at.

Efficient interaction feedback In a workshop paper [A1] we implemented voice and visual feedback (colored lights) for a robot selection acknowledgment. As a part of the relative localization system based on fiducial markers presented in [A2], we have paired a camera on the user's wrist with a laser pointer to help the user to point at the markers. In all our works we also used haptic feedback (vibrations) to indicate and acknowledge various system states. Our main contribution—an experimental study on the effects of continuous visual feedback on duration and accuracy of pointing—is presented in a conference paper [A5].

Other publications The author also published two videos [A3; A9] and demonstrated live the implemented pointing-based system for quadrotor control [A10] at the Human-Robot Interaction conference (HRI). Additionally, the author contributed to [A11; A12] publications.

Awards

The public demonstration [A10] and the video [A9] describing our system received the Best Demo and an honorable mention (the Best Video according to reviewers) awards at the Human-Robot Interaction 2019 (HRI '19) conference.

Patents

Based on the two research topics presented in this thesis we filed two international patents. Namely, the patent [A13] describes the relative localization method that we have presented and evaluated in a conference paper [A8]; and the patent [A14] describes the control method of devices in 3D space using pointing gestures that we evaluated in a drone control task in a conference paper [A6].

References

[A1] Boris Gromov, Luca Maria Gambardella, and Gianni A. Di Caro. Wearable multi-modal interfaces for mixed-initiative interaction in human multi-robot teams. In *ICRA Workshop on Fielded Multi-Robot Systems Operating on Land, Sea, and Air*, May 2016.

[A2] Boris Gromov, Luca Maria Gambardella, and Gianni A Di Caro. Wearable multi-modal interface for human multi-robot interaction. *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 240–245, Oct 2016. doi: 10.1109/SSRR.2016.7784305.

[A3] Boris Gromov, Luca Gambardella, and Alessandro 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. ISBN 978-1-4503-5615-2/18/03. doi: 10.1145/3173386.3177530.

[A4] Boris Gromov, Luca Gambardella, and Alessandro Giusti. Guiding quadrotor landing with pointing gestures. In *12th International Workshop on Human Friendly Robotics*. Springer International Publishing, oct 2019. to appear.

[A5] Boris Gromov, Gabriele Abbate, Luca Gambardella, and Alessandro Giusti. Proximity human-robot interaction using pointing gestures and a wrist-mounted IMU. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8084–8091, May 2019. doi: 10.1109/ICRA.2019.8794399.

[A6] Boris Gromov, Jérôme Guzzi, Luca Gambardella, and Alessandro Giusti. Intuitive 3D control of a quadrotor in user proximity with pointing gestures. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. to appear.

[A7] Boris Gromov, Luca Maria Gambardella, and Alessandro Giusti. A study on human multi-robot interaction using distal deictic gestures. In *ICRA Workshop on Human Multi-Robot Systems Interaction*, Jun 2017.

[A8] Boris Gromov, Luca Gambardella, and Alessandro Giusti. Robot identification and localization with pointing gestures. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3921–3928, October 2018. doi: 10.1109/IROS.2018.8594174.

[A9] Boris Gromov, Jérôme Guzzi, Gabriele Abbate, Luca Gambardella, and Alessandro Giusti. Video: Pointing gestures for proximity interaction. In

HRI ’19: 2019 ACM/IEEE International Conference on Human-Robot Interaction, March 11–14, 2019, Daegu, Rep. of Korea, March 2019. doi: 10.1109/HRI.2019.8673020.

[A10] Boris Gromov, Jérôme Guzzi, Luca Gambardella, and Alessandro Giusti. Demo: Pointing gestures for proximity interaction. In *HRI ’19: 2019 ACM/IEEE International Conference on Human-Robot Interaction, March 11–14, 2019, Daegu, Rep. of Korea*, March 2019. doi: 10.1109/HRI.2019.8673329.

[A11] Denis Broggini, Boris Gromov, Luca M. Gambardella, and Alessandro 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, February 2018.

[A12] Jeffrey Delmerico, Stefano Mintchev, Alessandro Giusti, Boris Gromov, Kamilo Melo, Tomislav Horvat, Cesar Cadena, Marco Hutter, Auke Ijspeert, Dario Floreano, Luca M. Gambardella, Roland Siegwart, and Davide Scaramuzza. The current state and future outlook of rescue robotics. *Journal of Field Robotics*, pages 1–21, August 2019. doi: 10.1002/rob.21887.

[A13] Luca Maria Gambardella, Alessandro Giusti, Boris Gromov, and Jérôme Guzzi. System for controlling mobile device. International patent No. WO 2019/149921, Aug 2019. URL https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2019149921.

[A14] Boris Gromov, Jérôme Guzzi, Alessandro Giusti, and Luca Maria Gambardella. Method for controlling a device by a human. International patent (pending). Application No. PCT/EP2019/078099, Oct 2019.

Bibliography

[1] Apple. Apple ARKit. https://developer.apple.com/documentation/arkit. [Online; accessed: 2019-11-30].

[2] Antonio Bicchi, Michael A Peshkin, and J Edward Colgate. *Safety for Physical Human–Robot Interaction*, pages 1335–1348. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_58.

[3] Bitcraze. The Crazyflie nano quadcopter. Bitcraze official web-page. https://bitcraze.io. [Online; accessed: 2019-11-20].

[4] Martin Björklund. Effects of repetitive work on proprioception and of stretching on sensory mechanisms. *Medical Dissertation, UME A University*, 2004.

[5] Richard A Bolt. "Put-that-there": Voice and Gesture at the Graphics Interface. *Proceedings of the 7th annual conference on Computer graphics and interactive techniques - SIGGRAPH '80*, pages 262–270, 1980. ISSN 00978930. doi: 10.1145/800250.807503.

[6] Cynthia Breazeal, Atsuo Takanishi, and Tetsunori Kobayashi. *Social Robots that Interact with People*, pages 1349–1369. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_59.

[7] Denis Broggini, Boris Gromov, Luca M. Gambardella, and Alessandro 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, February 2018.

[8] Andrew G Brooks and Cynthia Breazeal. Working with Robots and Objects: Revisiting Deictic Reference for Achieving Spatial Common Ground. *Ges-*

ture, pages 297–304, 2006. doi: http://doi.acm.org/10.1145/1121241.1121292.

[9] George Butterworth. Pointing is the royal road to language for babies. In Sotaro Kita, editor, *Pointing: Where Language, Culture, and Cognition Meet*, chapter 2, pages 9–33. Lawrence Erlbaum Associates, Manwah, New Jersey, 2003.

[10] Jonathan Cacace, Alberto Finzi, and Vincenzo Lippiello. Implicit Robot Selection for Human Multi-Robot Interaction in Search and Rescue Missions. *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 803–808, 2016. doi: 10.1109/ROMAN.2016.7745211.

[11] Jessica R. Cauchard, Kevin Y. Zhai, Marco Spadafora, and James A. Landay. Emotion encoding in human-drone interaction. *ACM/IEEE International Conference on Human-Robot Interaction*, 2016-April:263–270, 2016. ISSN 21672148. doi: 10.1109/HRI.2016.7451761.

[12] A. Cockburn, P. Quinn, C. Gutwin, G. Ramos, and J. Looser. Air pointing: Design and evaluation of spatial target acquisition with and without visual feedback. *International Journal of Human Computer Studies*, 69(6):401–414, 2011. ISSN 10715819. doi: 10.1016/j.ijhcs.2011.02.005.

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

[14] Alex Couture-Beil, Richard T Vaughan, and Greg Mori. Selecting and Commanding Individual Robots in a Multi-Robot System. *2010 Canadian Conference on Computer and Robot Vision*, pages 159–166, 2010. doi: 10.1109/CRV.2010.28.

[15] David Droeschel, Jörg Stückler, and Sven Behnke. Learning to interpret pointing gestures with a time-of-flight camera. *Proceedings of the 6th international conference on Human-robot interaction - HRI '11*, pages 481–488, 2011. ISSN 9781450305617. doi: 10.1145/1957656.1957822.

[16] Gregory Dudek and Michael Jenkin. *Inertial Sensors, GPS, and Odometry*, pages 477–490. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_21.

[17] Matthias Faessler, Elias Mueggler, Karl Schwabe, and Davide Scaramuzza. A monocular pose estimation system based on infrared leds. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 907–913. IEEE, 2014.

[18] Ana Fernández, Luca Bergesio, Ana M Bernardos, Juan A Besada, and José R Casar. A Kinect-based system to enable interaction by pointing in smart spaces. 2015. doi: 10.1109/SAS.2015.7133613.

[19] Mark Fiala. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1317–1324, 2010. ISSN 01628828. doi: 10.1109/TPAMI.2009.146.

[20] Andrea Fossati, Miodrag Dimitrijevic, Vincent Lepetit, and Pascal Fua. From canonical poses to 3d motion capture using a single camera. *IEEE transactions on pattern analysis and machine intelligence*, 32(7):1165–1181, 2010.

[21] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2014.01.005.

[22] Claude Ghez, James Gordon, Maria Felice Ghilardi, and Robert Sainburg. Contributions of vision and proprioception to accuracy in limb movements. In *The cognitive neurosciences.*, chapter 35, pages 549–564. The MIT Press, Cambridge, MA, US, 1995. ISBN 0-262-07157-6 (Hardcover).

[23] Google. Google ARCore. `https://developers.google.com/ar/discover`. [Online; accessed: 2019-11-30].

[24] Boris Gromov, Luca Maria Gambardella, and Gianni A Di Caro. Wearable multi-modal interface for human multi-robot interaction. *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 240–245, Oct 2016. doi: 10.1109/SSRR.2016.7784305.

[25] Boris Gromov, Luca Maria Gambardella, and Gianni A. Di Caro. Wearable multi-modal interfaces for mixed-initiative interaction in human multi-robot teams. In *ICRA Workshop on Fielded Multi-Robot Systems Operating on Land, Sea, and Air*, May 2016.

[26] Boris Gromov, Luca Maria Gambardella, and Alessandro Giusti. A study on human multi-robot interaction using distal deictic gestures. In *ICRA Workshop on Human Multi-Robot Systems Interaction*, Jun 2017.

[27] Boris Gromov, Luca Gambardella, and Alessandro Giusti. Robot identification and localization with pointing gestures. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3921–3928, October 2018. doi: 10.1109/IROS.2018.8594174.

[28] Boris Gromov, Luca Gambardella, and Alessandro 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. ISBN 978-1-4503-5615-2/18/03. doi: 10.1145/3173386.3177530.

[29] Boris Gromov, Gabriele Abbate, Luca Gambardella, and Alessandro Giusti. Proximity human-robot interaction using pointing gestures and a wrist-mounted IMU. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8084–8091, May 2019. doi: 10.1109/ICRA.2019.8794399.

[30] Boris Gromov, Luca Gambardella, and Alessandro Giusti. Guiding quadrotor landing with pointing gestures. In *12th International Workshop on Human Friendly Robotics*. Springer International Publishing, oct 2019. to appear.

[31] Boris Gromov, Jérôme Guzzi, Gabriele Abbate, Luca Gambardella, and Alessandro Giusti. Video: Pointing gestures for proximity interaction. In *HRI '19: 2019 ACM/IEEE International Conference on Human-Robot Interaction, March 11–14, 2019, Daegu, Rep. of Korea*, March 2019. doi: 10.1109/HRI.2019.8673020.

[32] Boris Gromov, Jérôme Guzzi, Luca Gambardella, and Alessandro Giusti. Intuitive 3D control of a quadrotor in user proximity with pointing gestures. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020. to appear.

[33] Bjarne Großmann, Mikkel Rath Pedersen, Juris Klonovs, Dennis Herzog, Lazaros Nalpantidis, and Volker Krüger. Communicating Unknown Objects to Robots through Pointing Gestures. In *Advances in Autonomous Robotic Systems 15th Annual Conference, TAROS 2014*, pages 209–220, Birmingham, 2014. Springer. doi: 10.1007/978-3-319-10401-0_19.

[34] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[35] Benjamin 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)*, pages 189–194, oct 2016. ISSN 21530866. doi: 10.1109/IROS.2016.7759054.

[36] Oliver Herbort and Wilfried Kunde. Spatial (mis-) interpretation of pointing gestures to distal spatial referents. *Journal of Experimental Psychology: Human Perception and Performance*, 42(1):78–89, 2016. ISSN 19391277. doi: 10.1037/xhp0000126.

[37] IKEA. Smart lighting. https://www.ikea.com/ch/en/cat/smart-lighting-36812/. [Online; accessed: 2019-12-18].

[38] Aleksandar Jevtić, Guillaume Doisy, Yisrael Parmet, and Yael Edan. Comparison of Interaction Modalities for Mobile Indoor Robot Guidance: Direct Physical Interaction, Person Following, and Pointing Control. *IEEE Transactions on Human-Machine Systems*, 45(6):653–663, 2015. ISSN 21682291. doi: 10.1109/THMS.2015.2461683.

[39] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. http://www.scipy.org/, 2001–. [Online; accessed: 2018-03-01].

[40] Lynette A Jones. Kinesthetic Sensing. In *Human and Machine Haptics*. MIT Press, 2000. doi: 10.1.1.133.5356. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.5356.

[41] Ricardo Jota, Miguel a Nacenta, Joaquim a Jorge, Sheelagh Carpendale, and Saul Greenberg. A Comparison of Ray Pointing Techniques for Very Large Displays. *GI '10 Proceedings of Graphics Interface 2010*, pages 269–276, 2010. ISSN 07135424.

[42] Runchang Kang, Anhong Guo, Gierad Laput, Yang Li, and Xiang Anthony. Minuet : Multimodal Interaction with an Internet of Things. In *SUI '19 Symposium on Spatial User Interaction*, New York, New York, USA, 2019. ACM Press. ISBN 978-1-4503-6975-6. doi: 10.1145/3357251.3357581. URL https://doi.org/10.1145/3357251.3357581.

[43] Aarlenne Z. Khan and J. Douglas Crawford. Ocular dominance reverses as a function of horizontal gaze angle. *Vision Research*, 41(14):1743–1748, 2001. ISSN 00426989. doi: 10.1016/S0042-6989(01)00079-7.

[44] Kazuaki Kondo, Genki Mizuno, and Yuichi Nakamura. Analysis of Human Pointing Behavior in Vision-based Pointing Interface System - difference of two typical pointing styles -. *IFAC-PapersOnLine*, 49(19):367–372, 2016. ISSN 24058963. doi: 10.1016/j.ifacol.2016.10.593.

[45] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. doi: 10.1177/0278364914554813.

[46] Shiqi Li, Chi Xu, and Ming Xie. A robust o (n) solution to the perspective-n-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1444–1450, 2012.

[47] Richard Lowry. Concepts and applications of inferential statistics. `http://vassarstats.net/textbook/`. [Online; accessed: 2019-08-26].

[48] Sven Mayer, Katrin Wolf, Stefan Schneegass, and Niels Henze. Modeling Distant Pointing for Compensating Systematic Displacements. In *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems*, volume 1, pages 4165–4168, 2015. ISBN 9781450331456. doi: 10.1145/2702123.2702332.

[49] Sven Mayer, Valentin Schwind, Robin Schweigert, and Niels Henze. The Effect of Offset Correction and Cursor on Mid-Air Pointing in Real and Virtual Environments. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018. doi: 10.1145/3173574.3174227.

[50] Brian D Mayton, Nan Zhao, Matt Aldrich, Nicholas Gillian, and Joseph A Paradiso. WristQue : A Personal Sensor Wristband. In *2013 IEEE International Conference on Body Sensor Networks*, pages 1–6. IEEE, 2013. ISBN 9781479903306. doi: 10.1109/BSN.2013.6575483.

[51] Mbientlab. Wearable technology for healthcare. Mbientlab official webpage. `https://mbientlab.com/`. [Online; accessed: 2019-08-26].

[52] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 37(3):311–324, 2007. ISSN 10946977. doi: 10.1109/TSMCC.2007.893280.

[53] Mani Monajjemi, Sepehr Mohaimenianpour, and Richard Vaughan. UAV, come to me: End-to-end, multi-scale situated HRI with an uninstrumented human and a distant UAV. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, number Figure 1, pages 4410–4417. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759649.

[54] S. Morrison and J. Keogh. Changes in the dynamics of tremor during goal-directed pointing. *Human Movement Science*, 20(4-5):675–693, 2001. ISSN 01679457. doi: 10.1016/S0167-9457(01)00072-0.

[55] Jawad Nagi, Alessandro Giusti, Gianni A. Di Caro, and Luca 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*, (c):252–253, 2014. ISSN 21672148. doi: 10.1145/2559636.2559833.

[56] Jawad Nagi, Alessandro Giusti, Luca M Gambardella, and Gianni A. Di Caro. Human-swarm interaction using spatial gestures. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3834–3841, 2014. ISBN 9781479969340. doi: 10.1109/IROS.2014.6943101.

[57] Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. Mid-Air Pointing on Ultra-Walls. *ACM Transactions on Computer-Human Interaction*, 22(5):1–62, 2015. ISSN 10730516. doi: 10.1145/2766448.

[58] Kai Nickel and Rainer 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*, pages 140–146, 2003. ISSN 02628856. doi: 10.1145/958432.958460.

[59] Jorge Nocedal and S. Wright. *Quasi-Newton Methods*, pages 135–163. Springer New York, New York, NY, 2006. ISBN 978-0-387-40065-5. doi: 10.1007/978-0-387-40065-5_6.

[60] Parrot. Parrot Bebop 2. `https://www.parrot.com/us/drones/parrot-bebop-2`. [Online; accessed: 2019-11-30].

[61] Maria Pateraki, Haris Baltzakis, and Panos Trahanias. Visual estimation of pointed targets for robot guidance via fusion of face pose and hand orientation. *Computer Vision and Image Understanding*, 120:1–13, 2014. ISSN 10773142. doi: 10.1016/j.cviu.2013.12.006.

[62] A. Pesenti Gritti, O. Tarabini, J. Guzzi, G. A. Di 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*, pages 4096–4103, Sept 2014. doi: 10.1109/IROS.2014.6943139.

[63] Jeffrey S. Pierce, Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. Image plane interaction techniques in 3D immersive environments. *Proceedings of the 1997 symposium on Interactive 3D graphics - SI3D '97*, pages 39–ff., 1997. ISSN 2000-6764. doi: 10.1145/253284.253303.

[64] Katrin Plaumann, Matthias Weing, Christian Winkler, Michael Müller, and Enrico Rukzio. Towards accurate cursorless pointing: the effects of ocular dominance and handedness. *Personal and Ubiquitous Computing*, pages 1–14, 2017. ISSN 16174909. doi: 10.1007/s00779-017-1100-7.

[65] Shokoofeh Pourmehr, Valiallah Mani Monajjemi, Richard Vaughan, and Greg Mori. 'You two! Take off!': Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice commands. *IEEE International Conference on Intelligent Robots and Systems*, pages 137–142, 2013. ISSN 21530858. doi: 10.1109/IROS.2013.6696344.

[66] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[67] Fabrice R Sarlegna and Robert L Sainburg. *The Roles of Vision and Proprioception in the Planning of Reaching Movements*, pages 317–335. Springer US, Boston, MA, 2009. ISBN 978-0-387-77064-2. doi: 10.1007/978-0-387-77064-2_16.

[68] 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*, pages 380–385, Oct 2006. doi: 10.1109/IROS.2006.281797.

[69] Patrik Schmuck and Margarita Chli. Multi-UAV Collaborative Monocular SLAM. *Icra 2017*, pages 3863–3870, 2017. doi: 10.3929/ethz-a-010861169.

[70] Samuel J Sober and Philip N Sabes. Multisensory integration during motor planning. *Journal of Neuroscience*, 23(18):6982–6992, aug 2003. ISSN 1529-2401. doi: citeulike-article-id:409345.

[71] Sphero. Force Band Training Series 2 — Droid Control. Official video guide for Sphero BB-8 robot. `https://youtu.be/Yxj961R5Fc8`. [Online; accessed: 2019-12-16].

[72] Jesus Suarez and Robin R Murphy. Hand gesture recognition with depth images: A review. *Ro-Man, 2012 Ieee*, pages 411–417, 2012. ISSN 1467346047.

[73] Junichi Sugiyama and Jun Miura. A wearable visuo-inertial interface for humanoid robot control. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 235–236. IEEE, mar 2013. ISBN 9781467330558. doi: 10.1109/HRI.2013.6483588.

[74] Daniel Szafir, Bilge Mutlu, and Terrence Fong. Communicating Directionality in Flying Robots. *Proceedings of the 10th ACM/IEEE International Conference on Human-Robot Interaction*, 2(1):19–26, 2015. ISSN 21672148. doi: 10.1145/2696454.2696475.

[75] Hong Z Tan, Brian Eberman, Mandayam a Srinivasan, and Belinda Cheng. Human factors for the design of force-reflecting haptic interfaces. *American Society of Mechanical Engineers, Dynamic Systems and Control Division (Publication) DSC*, 55-1:353–359, 1994. doi: 10.1.1.50.2564.

[76] Janet L Taylor and D.I. McCloskey. Pointing. *Behavioural Brain Research*, 29(1-2):1–5, jul 1988. ISSN 01664328. doi: 10.1016/0166-4328(88)90046-0.

[77] Thalmic Labs. Myo Armband. Official web-page. `https://www.myo.com/`. [Online; accessed: 2017-03-10].

[78] Michal Tölgyessy, Martin Dekan, František Duchoň, Jozef Rodina, Peter Hubinský, and Luboš Chovanec. Foundations of Visual Linear Human–Robot Interaction via Pointing Gesture Navigation. *International Journal of Social Robotics*, 9(4):509–523, 2017. ISSN 18754805. doi: 10.1007/s12369-017-0408-9.

[79] Alexander J B Trevor, John G. Rogers, Akansel Cosgun, and Henrik I. Christensen. Interactive object modeling & labeling for service robots. *ACM/IEEE International Conference on Human-Robot Interaction*, page 421, 2013. ISSN 21672148. doi: 10.1109/HRI.2013.6483627.

[80] Satoshi Ueno, Sei Naito, and Tsuhan Chen. An efficient method for human pointing estimation for robot interaction. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1545–1549. IEEE, oct 2014. ISBN 978-1-4799-5751-4. doi: 10.1109/ICIP.2014.7025309.

[81] Deepak Vasisht, Swarun Kumar, and Dina Katabi. Decimeter-Level Localization with a Single WiFi Access Point. In *NSDI 2016*, pages 165–178, 2016. ISBN 978-1-931971-29-4.

[82] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 00994987. doi: 10.2307/3001968.

[83] Marta Wnuczko and John M. Kennedy. Pivots for pointing: Visually-monitored pointing has higher arm elevations than pointing blindfolded. *Journal of Experimental Psychology: Human Perception and Performance*, 37 (5):1485–1491, 2011. ISSN 1939-1277. doi: 10.1037/a0024232.

[84] Michael T. Wolf, Christopher Assad, Matthew T. Vernacchia, Joshua Fromm, and Henna L. Jethani. Gesture-based robot control with variable autonomy from the JPL BioSleeve. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1160–1165, 2013. ISSN 10504729. doi: 10.1109/ICRA.2013.6630718.