Self-Supervised Robot Learning for Spatial Perception

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
Mirko Nava

under the supervision of
Prof. Luca Maria Gambardella
co-supervised by
Prof. Alessandro Giusti

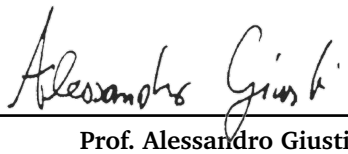
December 2023

Dissertation Committee

Prof. Cesare Alippi	Università della Svizzera italiana, Switzerland
Prof. Piotr Krzysztof Didyk	Università della Svizzera italiana, Switzerland
Prof. Cesar Cadena	ETH Zurich, Switzerland
Prof. Matteo Matteucci	Politecnico di Milano, Italy

Dissertation accepted on 11 December 2023

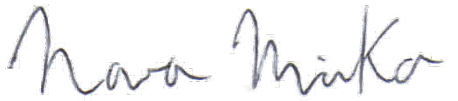
Prof. Luca Maria Gambardella
Research Advisor
Università della Svizzera italiana, Switzerland



Prof. Alessandro Giusti
Research Co-Advisor
Università della Svizzera italiana, Switzerland

Prof. Walter Binder
PhD Program Director

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.



Mirko Nava
Lugano, 11 December 2023

To the people I hold dear in my heart,
for people are what matters the most.

Acknowledgements

There are many people I would like to acknowledge who helped me in countless ways throughout this wonderful journey. To begin with, I want to express my sincere gratitude to Luca Maria Gambardella and Alessandro Giusti for being my advisors: thank you for showing me how to transform abstract ideas into concrete and formal research questions, present results to the public, and for teaching me the importance of always asking why, with curiosity being one of the main driving forces for my research. Without your guidance, support, and patience, my first steps in the academic world would not have been possible.

Special thanks to Mauro Pezzè for being responsible for the double Master degree program with Università degli Studi di Milano-Bicocca (UNIMIB): by being accepted into the program, I got to study at Università della Svizzera italiana (USI), attending courses held by internationally renowned professors and experiencing a different way of teaching while living in the wonderful Lugano, a city I grew to love. Furthermore, I had the opportunity to do my Master thesis with my advisor and get passionate about robotics and artificial intelligence. Without this experience, I probably would have never thought about moving to Switzerland or applying for a Ph.D. altogether – something which, I believe, has drastically changed my life.

Many thanks to my fellow double-degree mates Dario Mantegazza and Gabriele Abbate, which were always there to discuss the silliest ideas I came up with and, overall, for being great friends. Many thanks should also go to the Robotics Lab at the Dalle Molle Institute for Artificial Intelligence (IDSIA). This endeavor would not have been possible without the help and joint effort of my colleagues Antonio Paolillo, Daniele Palossi, Omar Chavez-Garcia, Jérôme Guzzi, Boris Gromov, Elia Cereda, Luca Crupi, Simone Arreghini, and Nicholas Carlotti.

These acknowledgments would not be complete without thanking all the wonderful people I had to pleasure of interacting with in IDSIA, the staff and fellow Ph.D. students at Università della Svizzera italiana (USI) and Scuola Universitaria Professionale della Svizzera italiana (SUPSI). I thank my parents, my beloved girlfriend Cristina, and the many friends I made throughout the years for motivating me to push through the bad times and for being responsible for many of the good times.

This work was supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics. As a member of NCCR Robotics, I had the pleasure of meeting, sharing ideas and collaborating with colleagues on challenging research problems, some of which I hold as dear friends.

Finally, I thank Cesare Alippi, Piotr Krzysztof Didyk, Cesar Cadena, and Matteo Matteucci for serving on my committee, dedicating their time and effort to review my work, and providing valuable suggestions.

Abstract

Nowadays, deep learning techniques are ubiquitous for robot perception tasks, thanks to their ability to recognize complex patterns and handle high-dimensional data. Crucial to the success of a robot learning approach is the amount and quality of labeled training data. Collecting a large amount of labeled training data requires much effort and resources: human experts may be employed to manually label the collected data, requiring many man-hours, or alternatively, one may use dedicated equipment capable of providing the needed labels; however, acquiring and maintaining such equipment is expensive and requires an accurate setup, especially when the system needs calibration to match the expected ground truth. One potential solution is to use a simulator, providing perfect knowledge of the state of the environment. This solution, in turn, brings its own challenges related to the reality gap, i.e., the many differences between simulated and realistic data. Robots ought to work in the real world, where gathered information is complex, imprecise, and noisy by nature, whereas simulated data is often too simplistic for training.

In this dissertation, we discuss and propose novel approaches for self-supervised robot learning, where the robot autonomously collects data and uses it to supervise the training or fine-tuning of a deep learning model. Specifically, we focus on spatial perception tasks, which entail the robot's ability to interpret complex visual data to estimate the geometrical properties of the environment, including the location of humans, obstacles, robots, and other relevant objects. Self-supervised robot learning is compelling because it allows the robot to collect large quantities of training data without requiring the involvement of humans; in fact, the robot may collect data in all the environments it can explore, even the one in which it will be deployed. The model is trained on the task at hand using the collected data; in addition, the model may be asked to solve an auxiliary task, named pretext, to learn better features and improve its performance. By introducing the pretext task, we limit the need for labeled data required to achieve an adequate level of performance.

In the following, we describe our work in the field, from the design of approaches and their implementation, to the validation on held-out testing data and in-field experiments. Our contributions to the state of the art concern three areas within self-supervised robot learning. First, we propose novel ways to derive supervision from sensors mounted on the robot, combining multiple sensors' readings collected in a time window. Second, we tackle some of the shortcomings in current self-supervised robot learning approaches by taking advantage of partially labeled examples and dealing with the noise affecting sensor's readings. Third, we introduce novel self-supervised pretext tasks tailored to robotics and aimed at improving the performance of spatial perception models. Finally, we present three potential research avenues for alleviating the challenges associated with large-scale data collection and for the improvement of perception models.

Contents

Figures

Tables

Chapter 1

Introduction

We live in a vast and intricately complex world, overflowing with an immense amount of information. Despite all of this complexity, animals as well as humans routinely carry out most of their tasks effortlessly, without much thought given. Consider tasks in your daily routine, even mundane ones, such as making breakfast: from grabbing some food and a drink from the fridge, gathering tools such as cutlery and dishware, to plating the breakfast and pouring the drink inside a glass. These ordinary actions rely on a stream of information coming from our senses, including the visible spectrum of light we see, the range of sounds we hear, and the smallest of pressures we feel on our skin. Although we can handle various tasks with ease, the most technologically advanced robots still struggle to do the same. Robots encounter problems due to their inadequate perception skills, lacking a basic understanding of the world around them by interpreting and making sense of sensory information acquired from the environment. Examples include understanding the geometry of the scene, telling which objects are present and where they are located in space.

While in the past, robots were traditionally adopted in laboratory and controlled industrial plants, nowadays they are starting to be taken outside, performing activities in fields such as transportation, healthcare, agriculture, or in our households for cleaning and companionship; each situation being different from the others, but all sharing a less structured, less predictable environment. Modern robots, therefore, need to make sense of this complexity to be able to successfully accomplish their duties. Consequently, they require powerful systems capable of processing high-dimensional and complex data to construct an internal representation of the robot's surroundings. Furthermore, the learned skills should be applicable to different environmental conditions, e.g., low light, bright sunlight or noisy and loud environments, demonstrating a sufficient level of generality. Deep neural networks can handle high-dimensional and complex data; as such, they have become the new de facto standard for perception tasks. Using deep neural networks, however, is not an easy task and comes with the cost of requiring large amounts of labeled training data and the computing resources needed to store and train using this data. Collecting labeled data is expensive and requires a lot of effort and resources in terms of time and dedicated equipment. Additionally, said data must be collected from a multitude of environments in order to result in models that demonstrate an adequate generalization, adding further cost to the training.

This thesis is about designing approaches for robots to learn perception skills, limiting the need for expensive labeling procedures without sacrificing much in terms of performance.

Specifically, we focus on spatial perception tasks where the robot has to interpret high-dimensional, visual data to estimate geometrical properties of the environment, like the location of humans, obstacles, robots and other objects of interest.

1.1 Problem Statement

Consider the task of detecting obstacles from frames streamed by a front-facing camera mounted on a self-driving car. The ideal scenario is to have at our disposal a large dataset consisting of camera frames annotated with the exact location of obstacles; with this data, one trains a deep neural network model to predict the obstacles visible in camera frames and deploys it onto the car. In many real-world scenarios, however, knowing the location of obstacles is problematic: it requires perfect knowledge of the environment the robot is exploring, either by constructing a map before-hand or by using simulation; or alternatively, one may ask human experts to label the frames with the position of obstacles. These solutions are viable, and their implementation accomplishes the goal we set out to solve; in spite of their effectiveness, they require a great amount of resources and an equal amount of effort: constructing an environment map requires the purchase of expensive equipment external to the robot, including its setup and calibration, then one collects data in the mapped environment and repeats the whole process for multiple environments; relying on human experts to label the camera frames requires many paid man-hours, being in general expensive, even when services such as mechanical turk [42] are adopted; finally, leveraging the perfect knowledge of a simulator to generate synthetic data brings its own problem of adapting the learned models from simulation to reality, crossing the so-called reality gap, another research area which so far has not found a general and task-independent approach [153]. Due to the previously mentioned problems, only a handful of groups and companies have the necessary resources to embark in this resource-intensive and expensive collection process, with outcomes that are still far from perfect, as reported in a United States - Department of Transportation report [2] and portrayed in Figure 1.1. Of course, this companies do not share this data publicly, and even then, it is tied to a specific task out of the many for which such data would be really needed.

A desirable solution would feature the following properties: *inexpensive*, relying on hardware that is already fitted on the robot or inexpensive one to be mounted onto it while also making a thoughtful choice in terms of energy consumption and weight; *realistic*, relying on a system to generate data that resembles the real world, indistinguishable from that collected with the sensors fitted on the robot; *effective*, able to generate large quantities of data in different and varied environments, resulting in adequate model generalization and forgoing as little performance as possible with respect to supervised approaches. In the next section, we propose one such solution as the core of this dissertation.

1.2 Self-Supervised Robot Learning for Spatial Perception

While supervised learning has proven to be effective in many domains, it heavily relies on labeled examples provided by humans, which may not capture the range of possible scenarios a robot may encounter in the real world. This lack of diversity in training data hinders the generalization of learned perception models, resulting in subpar performance, potential safety risks for users and passersby, and slows the adoption of robots across various industries. Additionally, hand labeling is a very time-consuming and expensive process, and the alternative

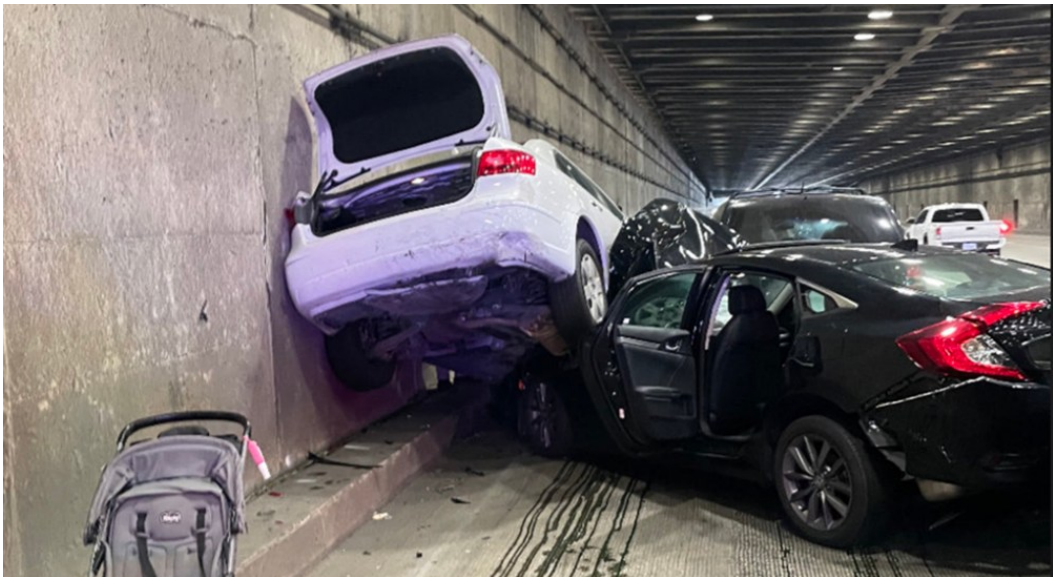


Figure 1.1. Access to labeled data is one limiting factor in building autonomous systems: self-driving cars are not yet ready to be considered reliable; in the United States alone over a ten month period, there have been 392 reported incidents [2] where these cars have failed to respond appropriately to the environment. Photo of a car crash caused by a self-driving car on San Francisco's Bay Bridge, November 2022, found on The Intercept journal[1].

of using external dedicated equipment to provide labels also involves a large upfront cost and, very often, this equipment requires a complex setup and calibration phase that renders moving the equipment to different environments impractical.

The two different meanings of self-supervised learning

In the machine learning literature, a common solution for this issue is to take advantage of cheaper unlabeled data by solving an additional task, named pretext, whose ultimate goal is to supervise the model into learning a general feature space. With the feature space learned, the model is then trained on the task we are interested in solving, named downstream or end task, using a more expensive, generally small dataset featuring labels for this task. In robotics, instead, the same issue is addressed by designing approaches that allow the robot to autonomously collect its own labelled training data without human supervision; this data is then used to train a model in the classic supervised learning manner. Moreover, since the required sensors are fitted on the robot, data can be collected in all environments the robot is capable of exploring, leading to a varied dataset and, therefore, model generalization. While both solutions entail the idea of a cheaper alternative to supervise the model without the involvement of humans or external systems, there are clear differences between the two. Specifically, in the machine learning sense, self-supervised approaches are aimed at learning a general feature space that suits as many end tasks as possible; in the robotics sense, it is specifically focused on the end task of interest and further constrains the unlabeled data to be collected by the robot. To

[1] https://theintercept.com/2023/01/10/tesla-crash-footage-autopilot

reconcile the two different meanings, we propose a definition of self-supervised robot learning that combines the two, in which a robot collects its own training data that can be unlabeled and used for a pretext task or labeled and used for the end task; with the former scenario, namely the pretext one, implying that the robot additionally collected a small labeled dataset for the end task.

Our contributions

In this dissertation, we propose the use of self-supervised robot learning to address the issues related to labeled data availability, data variety, and data efficiency for perception tasks. In detail, our contributions to the self-supervised robot learning field involve original uses of sensors to derive supervision for the end task, combining multiple ones, and aggregating information collected over a time horizon. Further, we address some of the limitations of current approaches by taking advantage of partially labeled examples, thus increasing the model supervision and proposing a simple yet effective way of handling noisy labels derived from sensors. Additionally, we investigate self-supervised pretext tasks tailored to robot perception tasks as a viable solution to reduce the reliance on labeled data without sacrifices in performance.

This dissertation is organized into chapters and structured according to Figure 1.2: after this introductory Chapter, we report a comprehensive review of the state of the art in Chapter 2, followed by our contributions spanning Chapters 3 through 8, and finally conclude the document with Chapter 9 by summarizing the proposed approaches, achieved results and describing future directions that follow this line of work. In what follows, we describe the three main contribution areas in detail, represented in Figure 1.2 by gray rounded-edge boxes, while the full publication list is provided in Appendix A.

Self-supervised robot learning for end tasks We investigate different supervision sources to train a robot on the end task, leveraging information coming from sensors and demonstrations of a control task. In Chapter 3, we propose to combine odometry and short-range proximity sensors' readings collected over a time window to construct a map-based obstacle label for the end task; given the camera frame collected at time t, we train a perception model to predict obstacles the robot has and will encounter during its trajectory within a time window centered in t. The resulting model is capable of relating hard-to-interpret and long-range camera frames with obstacles detected by straightforward short-range proximity sensors, demonstrating a high-level understanding of the perceived scene and detecting obstacles beyond the physical limit of the short-range sensors. Results computed on a separate testing set show that the approach is viable and leads to an average obstacle localization performance in the central area in front of the robot of 86% in AUC. In Chapter 6, we investigate the use of visual servoing task demonstrations to supervise a perception model in estimating the visual features of an object of interest; by backpropagating through a differentiable control law, the robot learns to imitate the demonstrated task, generating visual features that would result in the same trajectory. This approach enables robots to learn a perception task without explicit ground truth labels; instead, it leverages the supervision provided by expert demonstrations and knowledge of the control problem structure. The deployed model is capable of solving the feature extraction task and drive the robot with the control law to reach the desired position with respect to the object of interest.

The approaches discussed in this paragraph resulted in three publications: the first is a short paper demonstrating the initial findings on the use of odometry and short range sensors, published and presented at the AAAI 2019 conference [112]; the second is an article with novel

experiments on the same idea, published at the IEEE RA-L journal [113]; while the third is an article on the use of task demonstrations to supervise a perception model, published at the IROS 2022 conference [125].

Self-supervised robot learning improvements Autonomously collecting data to be used as training labels also comes with some challenges. Specifically, approaches based on robot exploration, such as the one proposed in Chapter 3, result in many samples featuring only partial labels or being totally unlabelled; moreover, the sensors used during collection are generally affected by noise, causing a decrease in performance. In Chapter 4, we tackle the first problem by introducing a consistency term inspired by human perception and their prior beliefs. Specifically, we propose a state-consistency loss that force models to have consistent predictions when shown different views of the same environment, some of which featuring occlusions. The idea is that by leveraging information present in one view, we supervise the robot in learning how to interpret the other view, even if some parts of the space are occluded; furthermore, knowing that a single state underlies the two views, we force model's predictions to match and do so even when no labels are available. In the evaluation, the perception model trained using additionally this loss results in improvement of up to 33% in R^2 when compared with one trained only on the end task loss. In Chapter 5, instead, we address the noise affecting the sensors' readings used to generate training labels. We consider the approach combining different information sources, such as odometry and fiducial marker detectors, both affected by noise. Our approach assumes the noise to be gaussian and with known parameters; we employ a Monte Carlo approach to approximate different possible realizations of the collected data by adding the sampled noise contribution. Using this examples, we train a perception model to solve three different spatial tasks, showing a consistent and significant reduction of errors and a more stable inference, reducing the average pose estimation error on a separate testing set by 5cm and 8 degrees.

The approaches discussed here resulted in two publications: the former is an article exploring the benefits of using a state-consistency loss in two different case studies, published at the IEEE RA-L journal [114], while the latter is an article addressing the problem of noise-affected sensors with a simple yet effective solution, published at the IEEE RA-L journal [115].

Self-supervised robot learning using pretext tasks When no suitable approach for end task label generation can be employed, the next best solution is to limit the amount of expensive labels required to achieve a satisfactory performance level. In this regard, we propose a pretext task in Chapter 7 aimed at drone localization and based on the drone's sound collected by a cheap stereo microphone. The pretext task consists in predicting the sound features from camera frames, exploiting the relation between sound and its source, in this case the drone itself. The robot, by solving this pretext task, learns features that are informative of the drone's sound and, following this relation, of the drone's location itself. Results show that this approach yields models with an average mean absolute error for the localization of the drone in image-space of 4 pixels while using a fraction of the labeled samples. Whereas, the model that has access to labels for the entire training set achieves 2 pixels on the same metric, showing that our approach successfully reduces the need for end task labels without sacrificing the performance. In Chapter 8, we investigate a different pretext task based on LED state prediction to improve the robot localization ability of a perception model. As opposed to the previous approach requiring a microphone, this one is compelling because LEDs are featured on most robot platforms. Collecting labels for the pretext task is straightforward and requires little effort, as the robot to

be localized can simply blink its LEDs and broadcast their state to another robot collecting this information, as well as its camera frames. With the addition of as little as 300 samples labeled with the drone's position, our approach leveraging frames with only LED state known achieves a median localization error of 9.9 pixels, while a supervised model trained on the full labeled dataset, which is 30× larger, scores 6.8 pixels.

The first approach discussed in Chapter 7, exploring a sound-based pretext task for drone localization, resulted in an article published at the IEEE RA-L journal [116]. The second approach discussed in Chapter 8, investigating LED state prediction as a pretext task, is currently under review at the IEEE RA-L journal.

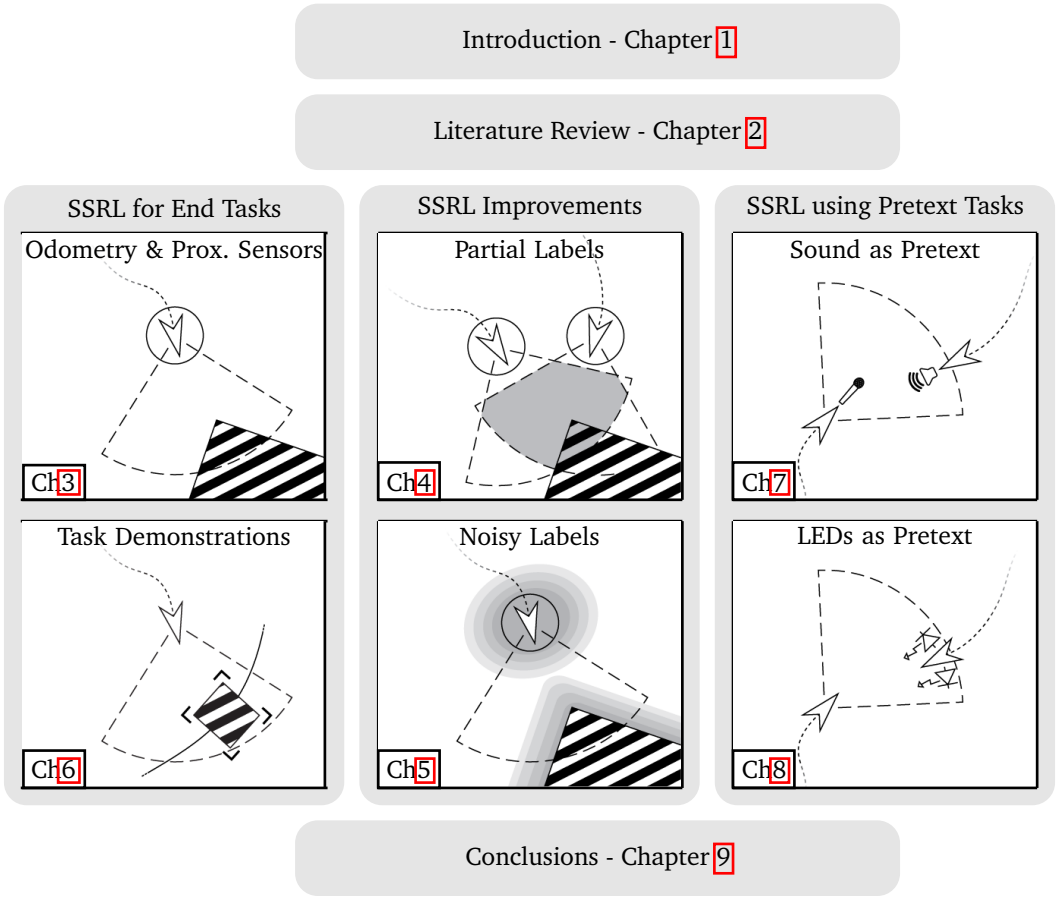


Figure 1.2. Structure of the doctoral dissertation. Contributed self-supervised robot learning approaches are represented by black-outlined boxes and discussed in their respective chapter. Contributions are grouped into three areas by gray rounded-edge boxes, representing approaches for supervising end tasks, additionally using pretext tasks, and overcoming some limitations of self-supervised robot learning.

Chapter 2

Literature Review

Robots are starting to be employed in complex and unstructured environments, presenting them with numerous challenges. These challenges include operating in dynamic environments with varying terrains, changing weather conditions and the presence of other moving entities, requiring robots to adapt their perception based on the setting. Additionally, robots often interact with other agents, such as humans or other robots, adding another layer of complexity. To address these challenges, robots are equipped with deep neural network models capable of handling high-dimensional data and learning complex functions. As such, one of the leading approaches to address these challenges is the use of robot learning (Section 2.1). However, deep learning models require lots of labeled data to train on, and such data is difficult to collect in large quantity, especially considering it should ideally represent all possible scenarios faced by the robot during deployment. In the machine learning literature, the need for additional supervision is addressed by self-supervised pretext tasks, where additional tasks defined on unlabeled training examples are exploited to learn good features and then used to solve a variety of end tasks (Section 2.2). Specifically, the use of pretext tasks can be seen as a representation learning approach [16] with the added constraint that data used to learn a good feature representation is not labeled. Instead, in robotics, self-supervised robot learning tackles the problem by allowing robots to autonomously collect their own training data cheaply and conveniently, leveraging the fitted sensors as additional sources of information (Section 2.3). Despite two different meanings for the same term based on the context, similar ideas can be found in both; in Section 2.4, we propose our own definition of self-supervision within the context of robot learning to reconcile the two meanings. Many methods have been proposed to tackle a range of challenges associated with self-supervised robot learning and have been implemented across various tasks (Section 2.5).

2.1 Robot Learning

Robot learning is a heterogeneous research field at the intersection of machine learning and robotics. It is concerned with solving robotics problems using machine learning techniques, where a parametric model is trained from lots of samples.

Robots operate in a vast amount of environments to solve a variety of different tasks. There are large differences between one scenario and others, motivating the development of custom built robots, which differ in shape, mobility, and fitted sensors. Consequently, robot learning

is mostly done from scratch, without suitable datasets already available and on data collected by the robot itself. Learning from data collected on a different robot introduces a significant domain shift, which has detrimental effects on the learning process. In a scenario in which we control the robot used to collect data and the learning process itself, much effort is put into data collection since data is equally as important as the learning process.

From a machine learning perspective, the goal is to endow robots with the ability to solve the given task via learning, i.e., through a training process based on the collected data. Approaches can be categorized into different paradigms by the amount of target data, also called labels or ground truth, available for training. Supervised learning assumes all training data to have labels and focuses on learning the most from them. Instead, unsupervised learning is used when no labels are available: the learning process focuses on building an internal representation that places similar input patterns close to one another and different patterns apart. Semi-supervised learning is a hybrid of the two, considering data to be only partially labeled, usually in a small percentage. It focuses on learning the most from the labeled data, while learning useful feature representations from the unlabeled data. A recurrent term in all approaches is the word supervision, which tells how much the learning process can watch over or optimize an untrained model utilizing ground truth. It is easy to see how the aforementioned approaches differ in the amount of supervision. Therefore, they can be fitted into an axis of a graph, where supervised and unsupervised learning lie at both ends, and semi-supervised is somewhere in the middle.

2.2 Self-Supervised Learning

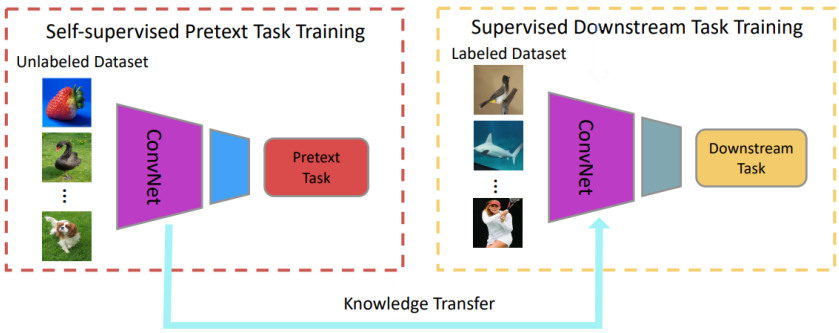


Figure 2.1. Self-supervised visual pretext task pipeline from the survey of Jing and Tian [80]. First, a CNN model learns good visual features by means of solving a pretext task using an unlabeled dataset. Second, knowledge of this features is transferred to a downstream or end task by fine-tuning (part of) the model on a labeled dataset, leading to improvements in performance.

Self-supervised learning – although the name suggests otherwise – is not indicative of a level of supervision, as was the case with supervised, semi-supervised and unsupervised paradigms, but instead of alternative ways to supervise a model with the available data itself [80, 207], hence self-supervised. Self-supervised strategies, in the machine learning sense, learn from unlabeled data a good feature space; in a second moment, this feature space is used to solve the task at hand more easily. To learn an effective feature space, we introduce an additional task based on unlabeled data, named pretext task. The pretext task is different from the task

we are actually interested in solving, called downstream or end task; a conceptual illustration of both tasks and the self-supervised training pipeline is shown in Figure 2.1. The rationale for pretext tasks is that the input patterns the model has to recognize are the same for both pretext and end tasks. In other words, we exploit the link between end and pretext tasks, e.g., that in order to localize an object, we need a high-level representation of the scene, which can be found in the latent space. Typical examples of pretext tasks include image reconstruction [173], image colorization [202], depth estimation from color [78], and inpainting [129]; an example of the latter is depicted in Figure 2.2.

For example, consider an undercomplete autoencoder learning to reconstruct an image through a bottleneck. In this case, the latent space of the bottleneck represents a very informative, high-level and low-dimensional description of the scene. With this high-level description, it is much easier to train a model on an end task, e.g., pose estimation, as part of the latent space would probably contain information about the location of objects.

In the next section, we discuss a different meaning for the "self-supervised" term used by roboticists to indicate the autonomous data collection by a robot without any human required to supervise the process.

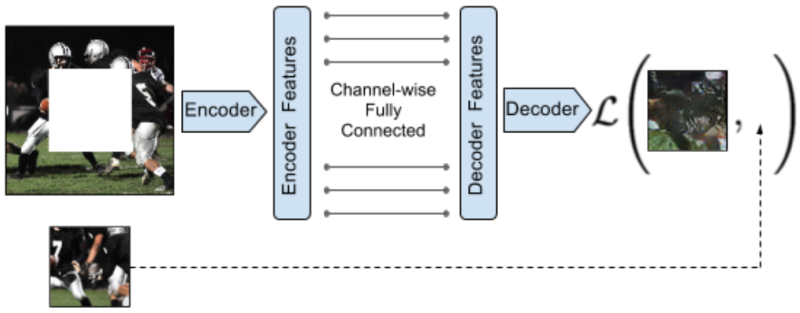


Figure 2.2. Self-supervised pretext task from Pathak et al. [129]. An undercomplete autoencoder is presented an image with the central part filled in with white. Its task is to fill in the pixels of the central part using the context provided by the surrounding area. This pretext task learns in the autoencoder's bottleneck a rich feature space, informative of the image's context, using only unlabeled images.

2.3 Self-Supervised Robot Learning

As previously mentioned, robot learning is often done from scratch, with no data available upfront, using a specific platform for which no previous work exists. In this context, the availability of training data is crucial for the success of a machine learning model. Practitioners often invest much time collecting ad-hoc datasets. Self-supervised learning, in the robotics sense, alleviates this problem by introducing strategies for obtaining supervision that do not involve expensive labeling procedures. In the robotics literature, self-supervised approaches are those in which an autonomous, unattended robot collects data by itself. By carefully choosing which inputs and outputs are involved in the learning problem, labels are produced from information gathered by the fitted sensors; an example of self-supervised robot learning is shown in Figure 2.3. Labeled data is, therefore, less expensive in terms of effort and exploration: an automated robot

is much more efficient than humans at collecting data and only uses its fitted hardware for the collection; in turn, this implies that a robot can collect data in every environment it can explore – even the one in which it will be deployed.

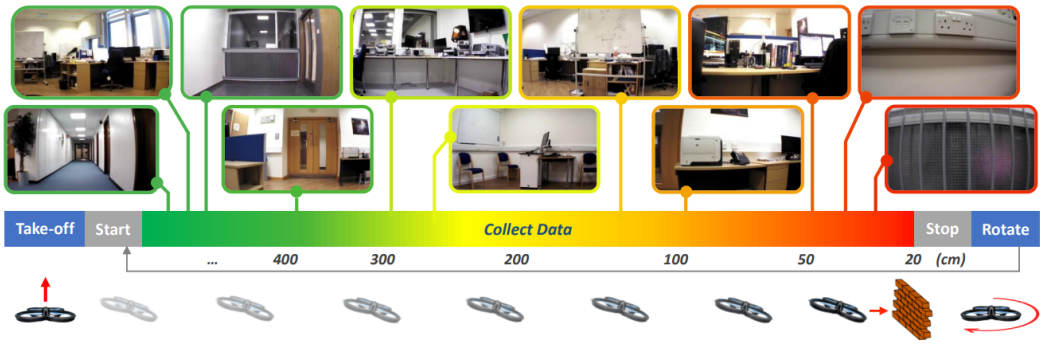


Figure 2.3. Self-supervised robot learning approach from Kouris and Bouganis [87]. The drone flies in an indoor environment by itself, collecting camera frames and infra-red distance sensors. The data collection controller is programmed to fly the drone forward, stopping before any collision is detected by the distance sensors and rotating in a new direction, repeating the process. With this approach, the authors construct a dataset and train a CNN model to predict the distance to obstacles from images; the resulting model is capable of piloting the drone in a reactive manner, without needing additional sensors.

2.4 Different Meanings for Self-Supervision

In Sections 2.2 and 2.3, we have discussed the meaning of the term "self-supervised" in the fields of machine learning and robotics respectively. With the increase in popularity of deep learning-based solutions in robotics, the clash of these two research fields has caused some confusion with regards to the meaning of this term. In this section, we delve into similarity and differences between the two meanings and reconcile them within the scope of robot learning.

Similarities

In machine learning, the term is used to refer to the fact that some tasks can be solved by learning from readily available data, in general considered unlabeled. Specifically, these tasks are defined in such a way that the respective ground truth is trivially defined (e.g., masking an image patch and ask the model recover it), a subset of the available data itself (e.g., predict the next frame from the current one), or is generated from simulators with perfect knowledge of the state. The goal is to improve the performance of a deep learning model by means of learning a good feature space, general enough to support many different tasks. Accordingly, the names pretext and downstream or end tasks are used to denote whether said task is intended for learning general features or is the one we are interested in solving.

In robotics, the term is used to refer to the autonomous and unattended collection of data by a robot, using its own hardware. The accent, here, is put on the fact that human supervision is not needed and, likewise, no human intervention is required to generate ground truth labels used for training. Instead of relying on humans, robots leverage their own sensor apparatus

and process the readings with algorithm to derive the labels (e.g., an increase in weight of the gripper before and after a grasp is indicative of the success of such attempt).

Both meanings entail the idea of taking advantage of cheap data to supervise a model: in one case, with pretext task; in the other, with labeled data collected by the robot. Interestingly, one can consider a robot collecting its own labeled data as yet another approach to obtain ground truth for a task, similarly to pretext learning. Additionally, some works stretch the robotics meaning to include the bootstrapping of a robot with a synthetic labeled training dataset before doing the autonomous data collection [45, 92], similarly to self-supervised approaches in the machine learning sense; however, this approaches fit the description of sim-to-real transfer better than self-supervision.

Differences

In machine learning, pretext tasks are solved first, using widely-adopted datasets from the literature; in a second step, a downstream or end task is learned by fine-tuning (a part of) the pre-trained model. In these works the focus is to show that the pretext task is general and supports many different end tasks successfully. These techniques leverage the availability of large-scale datasets in the literature, and focus on end tasks that all require similar perception skills, such as image classification, object detection, semantic segmentation.

Instead, in robotics, we have very different setups in terms of hardware and robot platforms. The end tasks considered are more varied, e.g., from regressing the pose of an object to predicting the probability of traversing a given terrain. As such, there are no large-scale datasets readily available; practitioners have the robot collect its own data. Furthermore, there is no emphasis on learning general features, but rather that those features are conducive for the chosen end task, which is chosen in advance. Additionally, in machine learning, the self-supervision is used to solve a pretext task, whose utility is limited to the extent in which the learned features improve the performance on the end task; instead, in robotics, the supervision is targeted at the end task itself.

Reconciling the two meanings

Given the similarities and differences of the two meanings, we discuss here a version of self-supervised robot learning that fits with the general idea of self-supervision in machine learning, and is tailored to robotics applications. Specifically, we propose to let the robot use its own hardware to autonomously collect data, be it unlabeled for a pretext task or labeled for an end task. For pretext task learning, the robot is trained simultaneously on the unlabeled data it collected and on a smaller labeled dataset, in which labels are generated from external systems (e.g., a tracking system, overhead depth camera). The dimension of the labeled dataset is considered to be small w.r.t. the unlabeled one since external systems are generally expensive, require precise calibration, and are often non-portable. Due to these issues, external systems are often found in dedicated laboratories, therefore the variety of scenes the robot is exposed to is limited; as such, the pretext task must leverage fitted sensors to allow the robot to explore different environments and, consequently, generalize well to unseen ones. The pretext task can be easily designed: it can be defined solely on input data or be based on data coming from other sensors fitted on the robot. The training is carried out simultaneously for both pretext and end tasks, as there is no need for the learned features to support different end tasks; instead, the goal is to be as specific as possible, choosing the pretext task which best suits the respective end

one. For end task learning, the robot generates ground truth labels for the end task without the use of external sensors; the collected labeled data is used to train the robot in a supervised fashion. This definition allows data collection to be relatively cheap, offer a satisfactory variety of scenarios collected from multiple environments, and ensures that the learned features are applicable to the specific tasks it needs to perform.

2.5 Applications to Perception Problems

Perception tasks consist in processing visual data, such as images, into a high-level representation of the system comprising robot and its surroundings. Due to the high-dimensionality and complexity of images, modern approaches utilize CNN, borrowing ideas from Deep Learning and Computer Vision fields and specializing them to tasks involving robots. Typical Perception tasks include localizing a given Object of Interest (OoI) (Section 2.5.1); learning to manipulate objects (Section 2.5.2); predicting terrain properties such as material and traversability score, or the presence of obstacles in the robot's trajectory (Section 2.5.3) and estimating the robot movements from image sequences (Section 2.5.4). In each of the following sections, we first consider approaches that solve the respective task in a fully-supervised fashion; then, we contrast them with self-supervised ones, highlighting how the latter enable the training of models with less effort and using a small amount of ground truth labels.

2.5.1 Object Pose Estimation

In designing a pick-and-place system for a given OoI, its position in space is central to the solution of the problem. Here, we discuss how to represent an object's position in space, challenges arising from different representations, and the state of the art in object pose estimation with supervised and self-supervised approaches.

Background

In robotics, the position of an object in space is called pose, and consists of the object's position and orientation defined with respect to (w.r.t.) a frame of reference. An object's pose is defined differently based on the dimensionality of the space, be it 2D or 3D.

Modeling the problem in the 2D space is a simplifying yet effective choice for ground robots since they are forced to the ground and cannot directly alter their height, like a flying robot would. A 2D pose consists of the objects position on a plane, denoted by x and y, and its angle of rotation ϕ. Alternatively, one can consider the full 3D pose of an object, having 6 Degrees of Freedom (DoF). A pose in 3D space is represented by 6 variables: x, y and z for the position and roll, pitch and yaw for the rotation. Lastly, some problems assume the camera to be calibrated; thus, its projection matrix is known [40]; as such, the 3D object pose can be recovered from its position on the image plane (u on the horizontal axis, v on the vertical) by back-projecting into the 3D space. This operation creates a ray passing through the camera optical axis and pointing towards the OoI. By adding the distance of the object d as an additional constraint, one recovers the point lying on the ray at that distance as the OoI's position.

In practice, pose estimation approaches can be divided into direct or indirect: direct approaches regress a pose representation with a model that is optimized using a loss designed specifically for the chosen representation [100, 206]; indirect approaches predict the 2D image-space location of known 3D points of interest, also known as 2D-3D correspondences, then solve

a Perspective-n-Point (PnP) problem to obtain the most likely 6-DoF pose of the object [94]; analogously, other approaches predict which 3D points belong to the object and then feed them as well as reference points of a known 3D model into an Iterative Closest Point (ICP) algorithm, in which 3D point correspondences are defined, and the distance between corresponding points is iteratively minimized, to obtain the object's pose. Given the emphasis that we put on collecting data, and the need for approaches that require a small amount of supervision, we focus on direct pose estimation since alternatives assume knowledge of the 3D object model, which requires scanning each object using dedicated equipment and do so for all objects we are interested in. Direct pose estimation simply refers to approaches in which the deep neural networks directly produces the 6-DoF pose.

One problem arises when learning a model for pose estimation: preserving the continuity of rotations around zero [100, 206], where tiny mistakes can result in large errors, e.g., the mean absolute error between 1° and 358° is 357° despite the two angles being only 3° apart. This problem makes it harder for models to learn the regression task, as rotation discontinuity requires specifically designed losses and causes gradients to be noisy, distorted [206]. The most straight-forward pose representation is the pose itself, making sure to limit the values for the Euler angles to be between 0 and 359 [102]. Alternatively, approaches discretize the possible rotations into a fixed number and then cast the problem to classification [136, 81]; utilize unit quaternions $\boldsymbol{q} \in \mathbf{H}$ as the representation, where $\mathbf{H}$ is the non-commutative ring of quaternions, and the quaternion distance as loss [100]; 3×3 matrices of the 3D rotation group $SO(3)$ and the geodesic distance [100]; or a 6D representation consisting of two of the three columns/rows of the 3×3 rotation matrix [206].

State of the art

In the literature, many approaches for object pose estimation are fully supervised, i.e., they assume to have unlimited access to ground truth pose labels. These approaches focus on achieving the best performance on State-of-the-art (SoA) benchmark datasets, failing to address the problem of lack of ground truth in many real world applications. Self-supervised approaches, instead, do not assume access to a large labeled dataset; they relax the assumption to either cheaply collecting their own ground truth or enact strategies to minimize the amount needed to obtain an adequate performance level.

Supervised approaches for indirect pose estimation focus on building 2D-3D correspondences, or 3D correspondences, from which algorithms can infer the full 6-DoF pose of the OoI. Tekin et al. [170] utilize a custom version of the YOLO model [145] to detect an object as well as the 2D locations of the eight corners of a 3D box containing the object; with these 2D-3D correspondences, they use the PnP algorithm to obtain the pose. Similarly, Tremblay et al. [175] predict the 2D-3D correspondences of the 3D bounding box of the object, however, they train using only domain-randomized [171] synthetic data; while Hu et al. [72] derives the pose by feeding the 2D-3D correspondences inside a deep neural network. Xiang et al. [194] predict the pixel-wise object center, orientation, and semantic segmentation, from which a voting algorithm produces the most likely object pose. Kehl et al. [81] use a custom SSD [96] model, with object rotations discretized into bins, to directly predict a coarse pose, then refined with the ICP algorithm. Other approaches refine an initial pose guess by rendering the known 3D object model back into the camera frame and minimize a visual error [93]; meanwhile others additionally use depth to compute geometrical discrepancies to drive the refinement process [182].

So far, we described approaches designed for generic objects; however, there are others specifically designed to estimate the pose of robots, which in general have a more complex shape, possibly changing over time. The solution is to use fiducial markers, which are used in conjunction with computer vision algorithms to predict the pose of the marker and the robot they are attached to. Saska et al. [155] use circles printed on paper, a simple yet effective approach, as the size of the perceived circle is proportional to its distance, and the distortion to an ellipse to its rotation; while Dias et al. [46] use multiple light-emitting markers, as perceived changes in the configuration relate to a change in pose. Other approaches utilize depth images to detect the robot itself, using handcrafted features [181] or by training a CNN model [24, 25].

Self-supervised approaches that assume knowledge of the 3D model of the object work by segmenting 3D points of the object [200], supervise the training by minimizing a visual loss [183], or by pre-training with synthetic data [45]. Zeng et al. [200] combine RGB-D frames taken by an array of cameras to segment the object, project its points into 3D space and remove those associated with the background, thus obtaining the object's point cloud from which they estimate the pose with ICP, as shown in Figure 2.4. Inspired by the iterative refinement approaches, Wang et al. [183] train a model by minimizing the visual differences between the perceived object and its 3D rendering placed at the inferred pose, exploiting differential rendering techniques; further, they improve the approach in [184] by introducing an additional loss based on depth images and the chamfer distance [23]. Lastly, by pre-training on synthetic data, Deng et al. [45] train a pose estimation model that is deployed on a robot arm; using this model, they are able to pick and place the object in different locations, thus producing real labeled data to train on iteratively.

For robot pose estimation, a CNN is trained using labels generated by the robot [92] with a Ultra-Wide Band (UWB) relative localization algorithm [179] to predict its image-space position and depth; combining this information with the camera intrinsics allows the authors to back-project the image-space position and obtain the full 3D position of the robot.

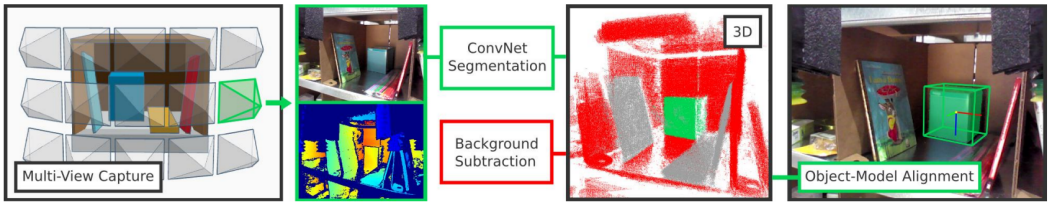


Figure 2.4. Self-supervised object pose estimation approach from Zeng et al. [200]. The robot takes color and depth images from multiple viewpoints of the scene. Each color image is processed by a pre-trained Fully Convolutional Network (FCN) [97] to segment objects in 2D; the segmentations are then combined in 3D by projecting the pixels. The resulting point cloud is further processed to remove the background and then aligned with a pre-scanned 3D model using ICP to obtain the object pose.

2.5.2 Manipulation

Manipulating an object involves understanding its geometrical properties and how those relate to the gripper fitted on a robot arm. Tasks such as grasping, pushing and pick-and-place are

common examples of robot manipulation. Due to the interplay between an object's geometry and the gripper or end-effector, manipulation tasks are complex and too difficult to be modeled analytically; as such, the prominent solution is to use deep learning. Adequately training deep learning models from high-dimensional, visual data requires lots of training samples collected in different environments. This is essential for the model to exhibit satisfactory performance and generalization ability. Despite much effort put towards supervised robot learning solutions for manipulation, few deep learning-based approaches are found in the literature due to the high cost of labeling the collected data [109]. Most approaches, nowadays, learn in a self-supervised manner, deriving the success of a grasp or push attempt using sensor data or casting the problem as reinforcement learning, where sensors are used to derive the reward signal.

Supervised approaches learn to grasp objects with robot arms from camera frames, determining the orientation of the object for a successful grasp [90]. Redmon and Angelova [144] split the image into multiple patches, predicting for each patch the probability of success for the grasp, the gripper rotation and finger distance; during inference, the grip settings with higher probability of success are selected for execution. Seita et al. [158] use depth images to predict which point to grasp on a bed sheet and use the robot for a bed-making task.

Self-supervised approaches learn the object orientation to perform a successful grasp with labels generated by measuring the weight before and after a grasp attempt [136]. Other approaches cast the problem as reinforcement learning, where rewards are generated autonomously. Specifically, Zeng et al. [201] learn to push and grasp with rewards obtained by measuring the finger distance after a grasp attempt and the difference in depth images before and after a push. Other approaches derive the reward signal from a given goal state: Berscheid et al. [18] derive the rewards by comparing the similarity of the embeddings of current and goal views; the embeddings are learned with a contrastive loss, where positive pairs consist of close-in-time images, and negatives of far-in-time or temporally non-ordered images. Jang et al. [77] propose to use representation learning to derive reward signals and the robot actions to supervised the learning of said representation, as depicted in Figure 2.5. Berscheid et al. [17] improve the data efficiency of a self-supervised learning approach by employing a depth-based consistency loss, where different crops of the same scene should result in the same predicted location for a successful grasp [17]. Unlike other approaches, Radosavovic et al. [142] pre-train a visual feature extractor with a masked autoencoding pretext task; they use self-supervised learning in the deep learning sense, where an auxiliary task is solved for the sake of learning a better feature space, in which to solve the task at hand.

2.5.3 Navigation

Navigating an environment requires a major understanding of terrain and obstacles present in the robot's surroundings and how these interact with the robot based on its motion capabilities. Navigation systems rely on a model to estimate traversable versus obstacle areas, classify terrain based on its composition (e.g., gravel, sand, concrete, dirt), and provide useful information such as estimated time of traversal and energy consumption. Based on these estimates, approaches directly compute in a purely reactive fashion which action to take to avoid collisions, while others feed the information inside the optimization problem of planners and/or controllers.

Training said models in a supervised fashion present a major challenge since collecting a large quantity of real-world visual data annotated with all of the aforementioned properties

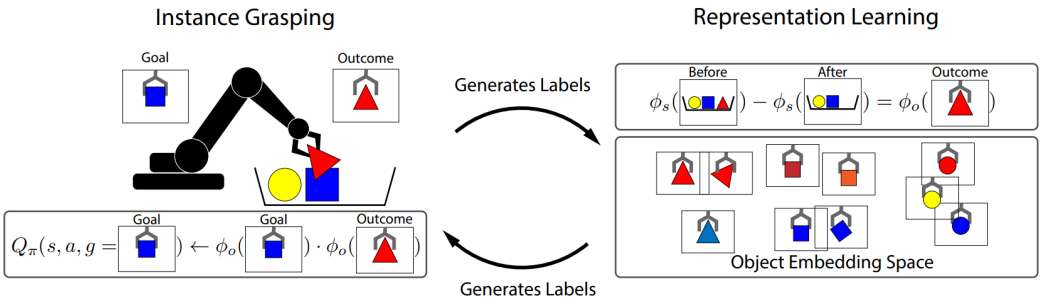


Figure 2.5. Self-supervised grasping approach from Jang et al. [77]. The authors derive the reward signal for a pick-and-place task using representation learning and exploiting object permanence [34]: when the robot successfully grasps an object and raises it, the object was present in the scene prior to the grasp, and is missing after that. They enforce the scene embedding before the grasp minus the one after it to match the object embedding. In the learned representation space, a reward signal is derived by computing the similarity between a given goal scene and the current scene observed by the robot.

is not feasible [137, 47]. Early approaches, therefore, relied on rule-based approaches [48, 49], avoiding the issue altogether at the cost of loss of generality, some casted the problem as imitation of a human pilot [138], while others used simulated data to train purely-visual models [137]. Given this challenge, a limited number of hand-labeled datasets can be found in the literature, some of which being domain specific [149], with supervised approaches relying on domain adaptation or other generalization techniques to avoid overfitting. Instead, most modern approaches use self-supervised learning, where robots autonomously collect their own ground truth labels [159].

Supervised approaches learn to estimate terrain properties and then feed this information into a planner to define a feasible trajectory or directly navigate the environment given an input image, i.e., in an end-to-end fashion. Classic approaches learn to navigate by imitating expert demonstrations in simulated environments [137]; or by imitating humans in real-life scenarios [138]. Tai et al. [166] learn to navigate from depth images, imitating a human pilot; while Pfeiffer et al. [135] use as input 2D laser range finder readings. Giusti et al. [63] teach a drone how to navigate complex forest trails by collecting data from a human expert traversing this environment. The authors train a CNN on the task of predicting the direction of the trail, as given by the trajectory of the human.

Other approaches simply limit the task to terrain property estimation: Rothrock et al. [149] train a CNN model to navigate Mars terrain from a large hand-labeled dataset, specifically by predicting the terrain class out of multiple possible classes from grayscale images. Zhang et al. [203] use an undercomplete convolutional autoencoder to segment RGB images into different terrain classes. Palazzo et al. [120] utilize a regression approach in which vertical bands of set width divide up the camera view. Each band starts from the top side of the image, has a height normalized between 0 and 100% of the image height, and represents the portion of space that is deemed non-traversable. By dividing each image into multiple bands, each parametrized by a scalar, the authors greatly simplify the traversability estimation task by reducing the dimensionality of the output; in the same work, they also propose an unsupervised domain adaptation

technique to generalize the model to different environments. Lastly, Chavez-Garcia et al. [30] train a CNN on synthetic 2.5D heightmap patches of the robot surroundings, labeled by the success of the robot in traversing those patches inside the simulator.

Self-supervised approaches collect their own training data, consisting of the notion of traversability, or the dual, obstacle-ness, and derive labels from readings of the onboard sensors. Dahlkamp et al. [43] learn to predict the terrain traversability in front of the robot using labels produced from the terrain height computed by an onboard stereo module, adopting a shallow machine learning model; instead, Hadsell et al. [68] use CNNs to solve a similar task on a different robot platform. Other approaches rely on spare laser scans [101] as the source of ground truth labels for traversable areas. Zhou et al. [204] use lidar scans to construct a 3D voxel representation of the environment in font of the robot. The authors derive from this representation summary features that are fed into a support vector machine predicting which image patches depict traversable terrain. Similarly, Maturana and Scherer [105] use a 3D voxel representation to represent the terrain below an unmanned aerial vehicle (UAV), training a 3D CNN to classify whether the depicted terrain is a save landing zone or not.

Interestingly, some approaches extract labels from information gathered at a later time than the input frame: in Barnes et al. [11], the authors note that successfully traversing a patch of terrain automatically indicates the label of pixels belonging to that patch; by leveraging this fact, they train a segmentation model to classify pixels into traversable or not. Kouris and Bouganis [87] rely on a similar mechanism but regress three values for the entire image representing the distance measurements obtained by laser sensors, enabling the drone to autonomously fly while avoiding collisions. Gandhi et al. [60] collect camera frames from a drone piloted by a random controller. When, inevitably, the drone crashes with an obstacle, it signals that all the collected camera frames near the crash time depict an obstacle, while all previous ones represent unoccupied space. Leveraging the labels extracted using this method, the authors teach the drone to autonomously fly while avoiding obstacles without the need to hand-label the frames.

Different approaches predict terrain properties that can be used by another module to determine the traversability score and, consequently, which route to take. Stavens and Thrun [163] assume that, at time zero, an autonomous vehicle is placed so to have traversable road in front of it; knowing this, they learn a model of the road visual appearance, steer the vehicle towards areas classified as road, and update the appearance model online. Force and vibrations felt at the wheel or foot level are indicative of the terrain a robot is currently on, enabling one to estimate its degree of traversability [21, 15]. Wellhausen et al. [191] derive labels from the force-torque sensors placed on the feet of a quadruped robot. This sparse information is projected onto the image space and propagated to dense, pixel-wise labels using the Mean teachers approach [169], as represented in Figure 2.6. In a similar fashion, Zürn et al. [208] use the sound produced by the wheels of a ground robot to infer the terrain type.

2.5.4 Visual odometry

Odometry is the ability of a robot to estimate its own motion by integrating sensors' readings over time. Visual odometry achieves the motion estimation by comparing pairs of consecutive images, and relating pixel movement to the robot's movement. Due to being an integrative process, odometry suffers from the drift problem [40] in which the accumulation of tiny instantaneous errors causes the estimate to depart from the real robot's position and doing more

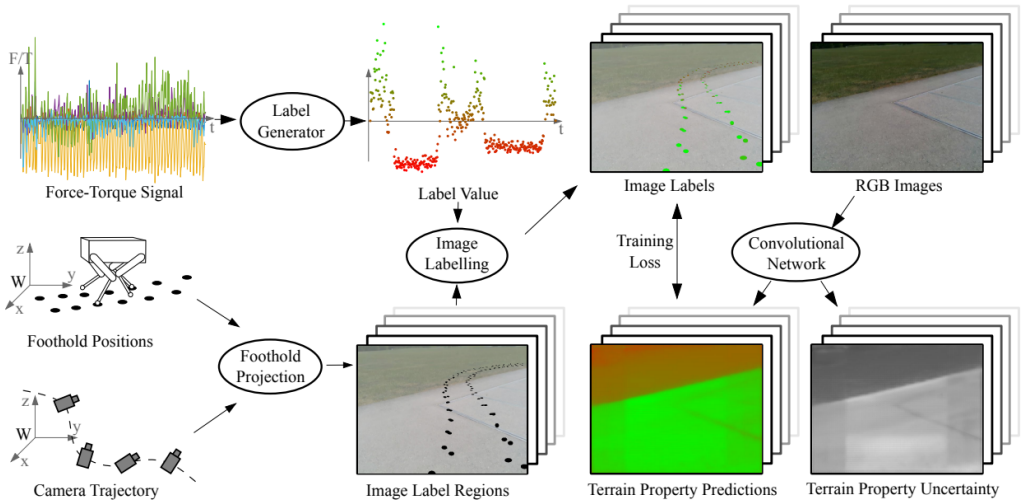


Figure 2.6. Self-supervised terrain properties estimation from Wellhausen et al. [191]. The ANYmal robot [74] traverses multiple terrains while recording images and torque sensors at its footholds. The torque sensors' readings are projected in image space using the camera intrinsics and extrinsics. The authors generate sparse terrain properties from the readings, such as terrain class and a ground reaction score. They propagate the sparse labels during training using different image augmentations with Mean teachers [169].

so as time passes. Notably, supervised approaches solve the issue by relying on external measurements not affected by drift, while self-supervised approaches rely on sensors' readings and optimize a consistency loss. The goal of a consistency loss is to enforce that the sensor reading at one location matches the reading from a different location composed with the model's estimate of the motion occurred between the two locations, i.e., that the estimated motion explains the difference of two real sensors' readings.

Supervised approaches estimate the visual odometry by predicting the relative motion occurred between a pair of images using a CNN [85]. Ummenhofer et al. [177] improve the prediction by refining the model output using a depth-based consistency loss: given the image pair, their model predicts the relative transformation as well as the depth of the two depicted scenes. The consistency loss warps one depth using the relative transformation and forces it to match the other depth, supervising the CNN model. Peretroukhin and Kelly [133] propose a different approach, in which they start from an already existing visual odometry estimator and train an additional model to predict correction terms that, when integrated with the estimates, result in a more precise odometry. Other approaches integrate monocular frames with a Recurrent Convolutional Neural Network (R-CNN), instead of using pairs of images, to estimate the motion [35, 186, 187].

Self-supervised approaches extract ground truth information directly from motor sensors, possibly affected by drift, learning the relative transformation between two images collected by an RGB camera [4]. Self-supervised approaches improve the model's performance by introducing additional consistency losses. Zhou et al. [205] enforce consistency by minimizing the

difference of inverse-warped depth images collected at time $t+1$ and $t-1$ with respect to the current depth image, collected at time t [205], as depicted in Figure 2.7. Iyer et al. [76] use a different consistency loss based on the notion that single consecutive estimates combine to form the estimate over a longer distance between two camera frames. The authors enforce that the concatenation of all relative motion estimates between time t and $t+n$ (e.g., $t+1$ w.r.t. t, $\ldots$, $t+n$ w.r.t. $t+n-1$) to match the single motion estimate of $t+n$ w.r.t. t.

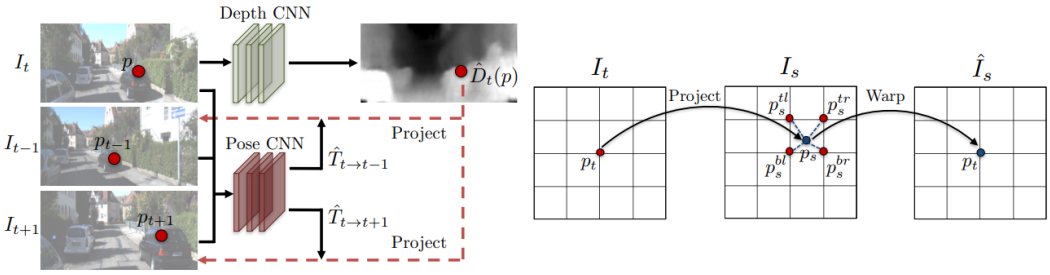


Figure 2.7. Self-supervised visual odometry approach from Zhou et al. [205]. The authors adopt two neural networks: a pre-trained CNN predicting depth from RGB images, and a visual odometry CNN predicting the difference of poses between two camera frames. Their consistency loss enforces the depth predicted from images at different time steps to match each other. Comparing depth from a pair of images taken from different poses requires inverse-warping one depth image using the transformation occurred between the two poses. The similarity between two depth images is used to supervise the visual odometry CNN.

Chapter 3

Supervision From Odometry and Proximity Sensors

In self-supervised learning for robotics, the traditional approach generates a training sample by processing the readings from one or more sensors collected at a single timestep. In this chapter, we take a different approach by exploring the use of readings collected at different time steps and combined with the robot's odometry to generate richer training labels. Although odometry can be imprecise, it provides valuable information that is often overlooked when it comes to automated label generation. Furthermore, by utilizing the readings collected in a static environment at different time steps, the robot learns to establish geometrical relationships between the current view of the environment and properties observed from multiple locations. This approach allows for a more comprehensive understanding of the environment and enhances the robot's perception capabilities.

3.1 Background

We consider a mobile robot capable of odometry and equipped with at least two sensors: a long-range one, such as a camera or laser scanner; and a short-range sensor such as a proximity sensor or a contact sensor (bumper). We then consider a specific perception task, such as detecting obstacles while roaming the environment. Regardless on the specific choice of the task and sensors, it is often the case that the long-range sensors produce a large amount of data, whose interpretation for the task at hand is complex; conversely, the short-range sensor readings directly solve the task, but with limited range. For example, detecting obstacles in the video stream of a forward-pointing camera is difficult but potentially allows us to detect them while they are still far; solving the same task with a proximity sensor or bumper is straightforward as the sensor directly reports the presence of an obstacle, but only works at very close range.

We propose a novel technique for solving a perception task by learning to interpret the long-range sensor data; in particular, we adopt a self-supervised learning approach in which future outputs from the short-range sensor are used as a supervisory signal. We develop the complete pipeline for an obstacle-detection task using camera frames as the long-range sensor and proximity sensor readings as the short-range sensor (see Figure 3.1). In this context, the camera frame acquired at time t (input) is associated to proximity sensor readings obtained at a

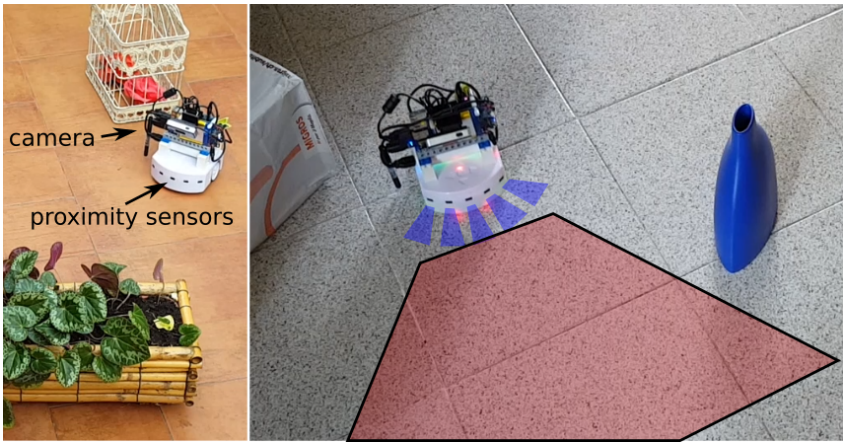


Figure 3.1. The Mighty Thymio robot in two environments; five proximity sensors can easily detect obstacles at very close range (blue areas), whereas the camera has a much longer range (red area) but its outputs are hard to interpret.

different time $t' \neq t$ (labels); for example, if the robot odometry detects it has advanced straight for 10cm between t and t', the proximity sensor outputs at t' correspond to the presence of obstacles 10cm in front of the pose of the robot at t. These outputs at time t' can be associated to the camera frame acquired at time t as a label expressing the presence of an obstacle 10cm ahead. The same reasoning can be applied to other distances, so that we define a multi-label classification problem with a single camera frame as input, and multiple binary labels expressing the presence of obstacles at different distances.

The approach is *self-supervised* because it does not require any explicit effort for dataset acquisition or labeling: the robot acquires labeled datasets unattended and can gather additional labeled data during its normal operation. Long-range sensors do not need to be calibrated: in fact, they could even be mounted at random, unknown poses on the robot. Exploiting a combination of long-range sensors is handled naturally by just using all of them as inputs to the learning model.

Potential instances of the approach include: a vacuuming robot that learns to detect dirty areas, by using a camera as the long-range sensor and an optical detector of dust in the vacuum intake as the short-range sensor; an outdoor rover learning to see challenging terrain by relating camera and/or LIDAR readings to attitude and wheel slippage sensors; a quadrotor learning to detect windows at a distance, using camera/LIDAR as long range sensors and a vision-based door/window detector which works only at close range as the short-range sensor. Note that in this case, the short-range sensor is not a physical one but is the output of an algorithm that operates on camera data but is unable to produce long-range results.

Our main contribution is a novel, general approach for self-supervised robot learning of long-range perception. In Section 3.2 we implement this model on the Mighty Thymio robot [66, 174] for obstacle detection using a forward-looking camera as the long-range sensor and five forward-looking proximity sensors as the short-range sensor. We report extensive experimental results on this task, and quantitatively evaluate the quality of predictions as a function of distance. To test the generality of the approach, we finally instantiate the it on a different task and report results obtained in simulation.

3.1.1 Related Work

A common theme in self-supervised robot learning approaches is that labels are acquired simultaneously to the data they are associated to, and consider only information from a single near-future step, as described in detail in Chapter 2. Our approach crucially differs in that we derive supervisory labels from multiple short-range sensor readings acquired at a *different time* than the long-range one to be classified, when the robot is at a different pose.

In this regard, similar approaches have been used in literature for terrain classification [21, 15]: in these approaches, accelerometer data is collected along with the front-facing camera's feed. Training examples are generated by matching the two streams in such a way that the image collected at a given time, which contains the visual representation of a terrain patch in front of the robot, is associated to the accelerometer readings collected when the robot was traversing that specific terrain patch, from which the label is derived. Note that this implies that the mapping between the image and the future robot poses is known, i.e., that the long-range sensor is calibrated. Our approach does not rely on the knowledge of such mapping; instead, we expect the Machine Learning model, which is fed the raw long-range sensor data without any specific geometric interpretation, to automatically devise it; this also allows us to simultaneously train for multiple labels at different relative poses.

Gandhi et al. [60] trained a model to determine whether the image acquired by the front camera of a drone depicts a nearby obstacle or not. The former class is assigned to all images acquired near the time in which of a drone crash is detected; remaining images are associated to the latter class. This approach can be seen as a specific instance of the one we propose, with the camera as a long-range sensor, a crash detector as a short-range sensor, and a single label corresponding to a generic "nearby" pose. Van Hecke et al. [180] adopt a similar approach to estimate average depth using a monocular image, by using the stereo vision depths from the past as trusted ground truth.

3.1.2 Model

We consider a mobile robot with pose $\boldsymbol{p}(t)$ at time t; the robot is equipped with one long range sensor $\boldsymbol{l}(\cdot)$ and one or more short-range sensors $\boldsymbol{s}_i(\cdot)$, $i = 1, \ldots, m$. For simplicity, we limit the analysis to wheeled mobile robots for which $\boldsymbol{p}(t) \in SE(2)$. We model all sensors as functions that return the sensor readings for a given timestep t.

We assume that short-range sensors return binary values $\boldsymbol{s}_i(\cdot) \in \{0, 1\}$ that provide very local but unambiguous information for the robot (e.g., bumpers). Instead, long-range sensors provide a wider but maybe not directly interpretable information (e.g., a camera).

We define a set $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ of predefined *target poses* relative to the current pose $\boldsymbol{p}(t)$ (see Figure 3.2): our objective is to predict the readings $\boldsymbol{s}_i(j), i = 1, \ldots, m \wedge j = 1, \ldots, n$ of all short-range sensors at the target poses, given the current reading $\boldsymbol{l}(t)$ of the long-range sensor.

3.1.3 Learning-Based Solution

We cast the problem as a supervised learning task. We gather a large dataset of training instances and use it to model the relation between $\boldsymbol{l}(\cdot)$ and $\boldsymbol{s}(\cdot)$. Each training sample consists in a tuple containing the long-range sensor reading and the short-range sensors' readings for all target poses. Each sample is collected in a self-supervised manner as the robot roams in the

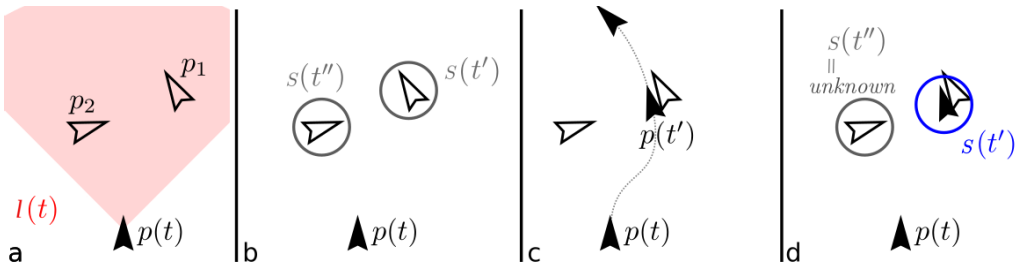


Figure 3.2. (a) A mobile robot at pose $p(t)$ has a long-range sensor $\boldsymbol{l}$ (red) and (b) a short-range sensor $\boldsymbol{s}$. Our objective is to predict the value of $\boldsymbol{s}$ at n target poses $p_1, p_2, \ldots p_n$ from the value of $\boldsymbol{l}(t)$. (c, d) For a given instance, we generate ground truth for a subset of labels by searching the robot's future trajectory for poses close to the target poses.

environment while sensing with all its sensors and recording odometry information; for each time t, we record $(\boldsymbol{p}(t), \boldsymbol{l}(t), \boldsymbol{s}_i(t), \ldots, \boldsymbol{s}_m(t))$.

Self-supervised label generation

After the data is collected, we consider each pose in the dataset as a training sample. Let $\boldsymbol{p}(t)$ be such pose. In order to generate ground truth labels, we consider each of the target poses $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ in turn. For each given target pose $\boldsymbol{p}_j$, we look for a time t' such that $\boldsymbol{p}(t')$ is closest to $\boldsymbol{p}_j$. In this step, we may limit the search to $t' \in [t - \delta_t, t + \delta_t]$, e.g., to limit the impact of odometry drift. If the distance between $\boldsymbol{p}(t')$ and $\boldsymbol{p}_j$ is within a tolerance δ_d, the recorded values of $\boldsymbol{s}_i(t')$, $i = 1, \ldots, n$ are used as the labels for target pose $\boldsymbol{p}_j$. Otherwise, the labels associated to target pose $\boldsymbol{p}_j$ are set to *unknown*. Therefore, it is possible that for a given instance some or even all labels are unknown. While in the former case the instance can still be used for learning, in the latter case it must be discarded.

The amount of training instances for which a given label is known depends on the corresponding target pose and on the trajectory the robot followed during data acquisition. Section 3.2.1 illustrates a robot's behavior designed to efficiently generate a large dataset for a specific set of target poses.

Given the structure of the labels, the machine learning problem we ought to solve is an instance of multi-label binary classification with incomplete training labels, in which the model predicts the value of m sensors at n poses (i.e., $n \times m$ labels) given one reading from $\boldsymbol{l}(\cdot)$. The specific model to solve this problem depends on the type of the data generated by $\boldsymbol{l}(\cdot)$; in Section 3.2.3, we consider a setting in which $\boldsymbol{l}(\cdot)$ outputs images, therefore we adopt a CNN.

3.2 Experiments

The robot platform adopted for the experiments is a Mighty Thymio [66], a differential drive robot equipped with 9 infra-red proximity sensors with a range of approximately 5 to 10cm, depending on the color and size of the object. 5 of these sensors point towards the front of the robot at angles of $-40°$, $-20°$, $0°$, $+20°$, $+40°$ with respect to the robot's longitudinal axis; we use these five sensors as the short-range sensors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_5$, and treat their output as a binary value: 1 for obstacle in range and 0 for no obstacle in range. The robot is also equipped with

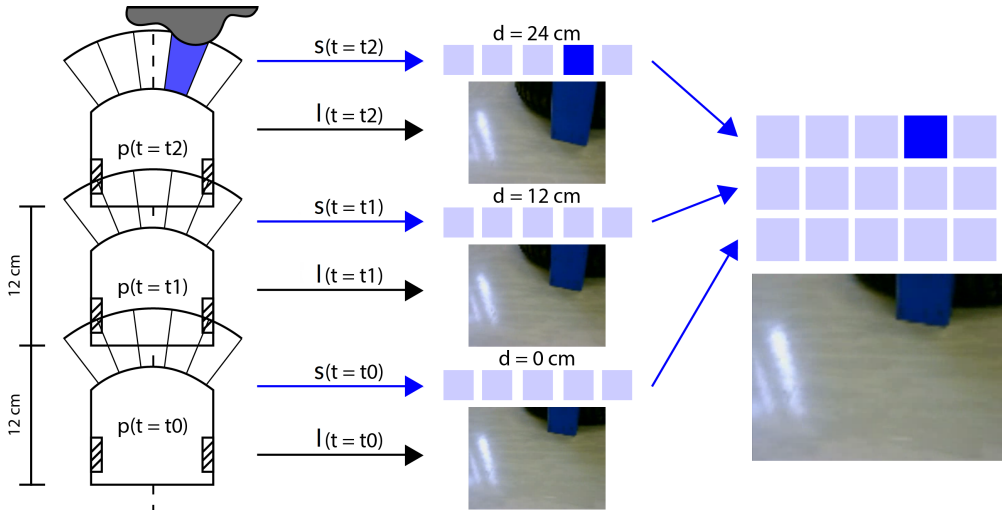


Figure 3.3. Simplified illustration describing how a training instance is built. The camera image from the current pose $\boldsymbol{p}(t)$ (bottom) is associated to the sensor readings (blue squares) from future poses that are close to the target poses aligned with the robot's axis.

a forward-looking 720p webcam with an horizontal Field of View (FOV) of 68°, used as the long-range sensor.

We define a set of 31 target poses $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{30}\}$ which lie in front of the robot, aligned with its longitudinal axis, evenly spaced at a distance of 0 to 30cm. Note that since target pose $\boldsymbol{p}_0$ coincides with the current robot pose $\boldsymbol{p}(t)$, labels for $\boldsymbol{p}_0$ are present in every training instance.

3.2.1 Data Acquisition Controller

We implemented an ad-hoc controller for efficient unattended collection of datasets, consisting of the readings from the five short-range sensors, the robot odometry and the camera feed. The controller behavior is illustrated in Figure 3.4: the robot moves forward (*a*) until an obstacle is detected (*b*) by any of the proximity sensors; at this point, it stops and defines 5 directions which are offset from the current direction by −30°, −15°, 0°, 15° and 30° respectively (*c*, the figure shows three for clarity). For each of these directions in turn, the robot: rotates in place to align with this direction, moves back by a fixed distance of 30cm (*d, f, h*), then moves forward by the same distance, returning to the starting position (*e, g, i*). After the process is completed, the robot rotates away from the obstacle towards a random direction, then starts moving forward again (*j*) and continues the exploration of the environment.

Note that the controller is built in such a way to efficiently populate labels for the target poses (i.e., it proceeds straight when possible); moreover, the controller strives to observe each obstacle from many points of view and distances, in order to mitigate the label imbalance in the data.

However, it is important to note that the general approach we propose is not dependent on any special controller. For any given choice of the target poses, a random-walk trajectory would eventually (albeit inefficiently) collect instances for all labels.

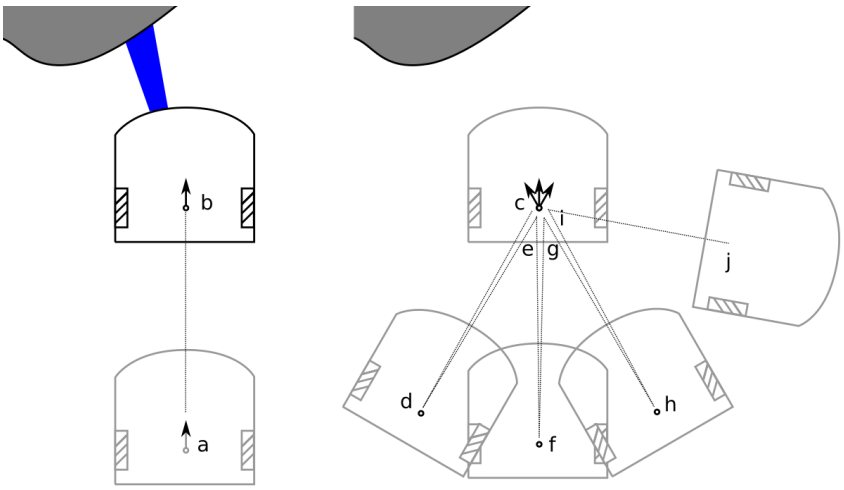


Figure 3.4. Example trajectory generated by the data acquisition controller.

3.2.2 Datasets

We acquired datasets from 10 different scenarios (see Figure 3.5), some indoor and some outdoor, each featuring a different floor type (tiled, wooden, cardboard, linoleum) and a different set of obstacles. For each scenario, we left the robot unattended, acquiring data for about 10 minutes using the controller described above.

The collected data amounts to 90 minutes of recording, which is then processed in order to generate labeled instances as described in Section 3.1.2, resulting in a total of 50K training examples extracted at about 10Hz. Figure 3.6 reports the total number of known labels as a function of the distance of the corresponding target pose. Note that the total of known labels for a distance of 0cm amounts to 250K, i.e., 50K for each of the 5 sensors. We observe that the classification problem is imbalanced in favor of negative labels; a potential countermeasure, which is not necessary in our case, is to implement a cost-sensitive loss [22].

All quantitative experiments reported below split training and testing data by grouping on scenarios, i.e., ensure that the model is always evaluated on scenarios different from those used for training. This allows us to test the model's generalization ability.

Data preprocessing and augmentation Camera frames are resized using bilinear interpolation to 80×64 pixel RGB images, then normalized by subtracting the mean and dividing by the standard deviation. The robot's pose $\boldsymbol{p}$ consists of its position on the ground plane and orientation angle $\langle x, y, \phi \rangle$ since the robot operates in 2D. Proximity sensor readings are expressed as binary values, i.e., 0 if no obstacle is detected and 1 if any obstacle is detected regardless on its range.

Data augmentation has been adopted to synthetically increase the size of the datasets: with probability 0.5, the image is flipped horizontally, and the corresponding target labels are modified by swapping the outputs of the left and right sensors, and the outputs of the center-left and center-right sensors; with probability $1/3$, a Gaussian noise with $\mu = 0$ and $\sigma = 0.02$ is added to the image; also, with probability $1/3$ the image is converted to grayscale; lastly, a smooth grayscale gradient with a random direction is overlayed on the image so as to simulate a soft

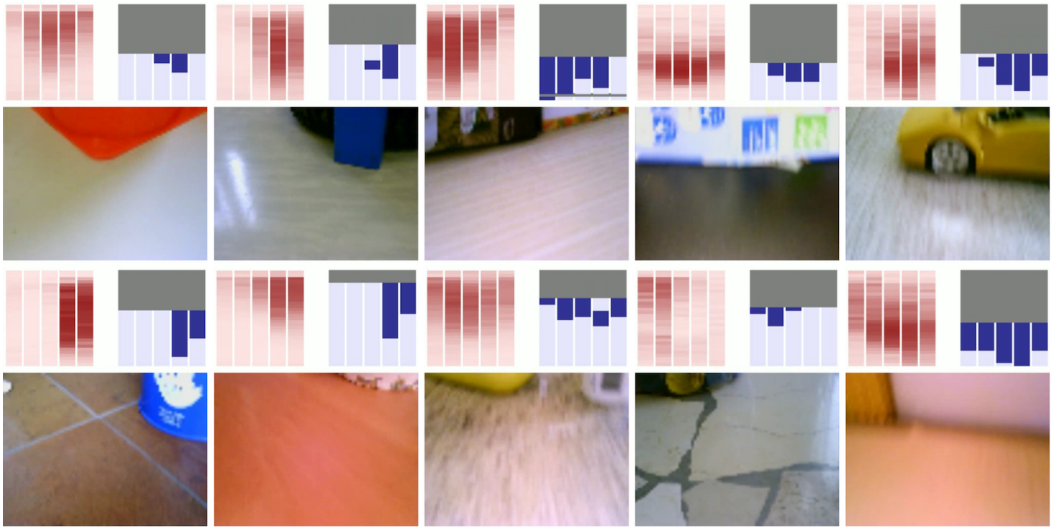


Figure 3.5. 10 instances from the acquired dataset, each coming from a different scenario (top row: scenarios 1 to 5; bottom row: 6 to 10). For each instance, we show the camera image (bottom) and the 31×5 ground truth labels as a blue matrix (top right): one row per distance, one column per sensor; dark = obstacle detected; light = no obstacle detected; distances masked by gray rectangles correspond to unknown labels (due to the robot never reaching the corresponding pose). The red heatmap at the top left shows the predictions of a model trained on the other 9 scenarios.

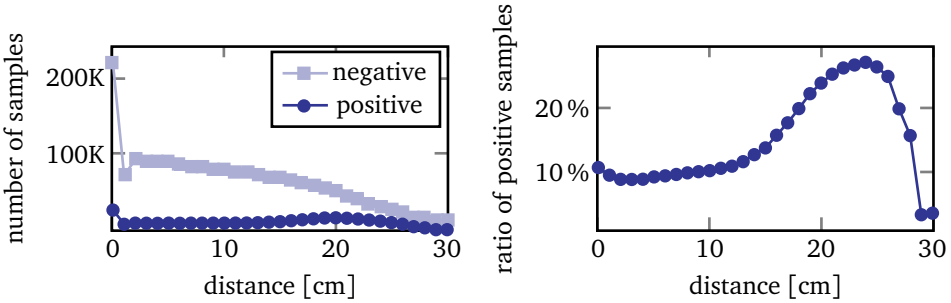


Figure 3.6. Left: number of known positive (obstacle) and known negative (no obstacle) labels as a function of the distance of the corresponding target pose. Right: percentage of positive labels as a function of distance.

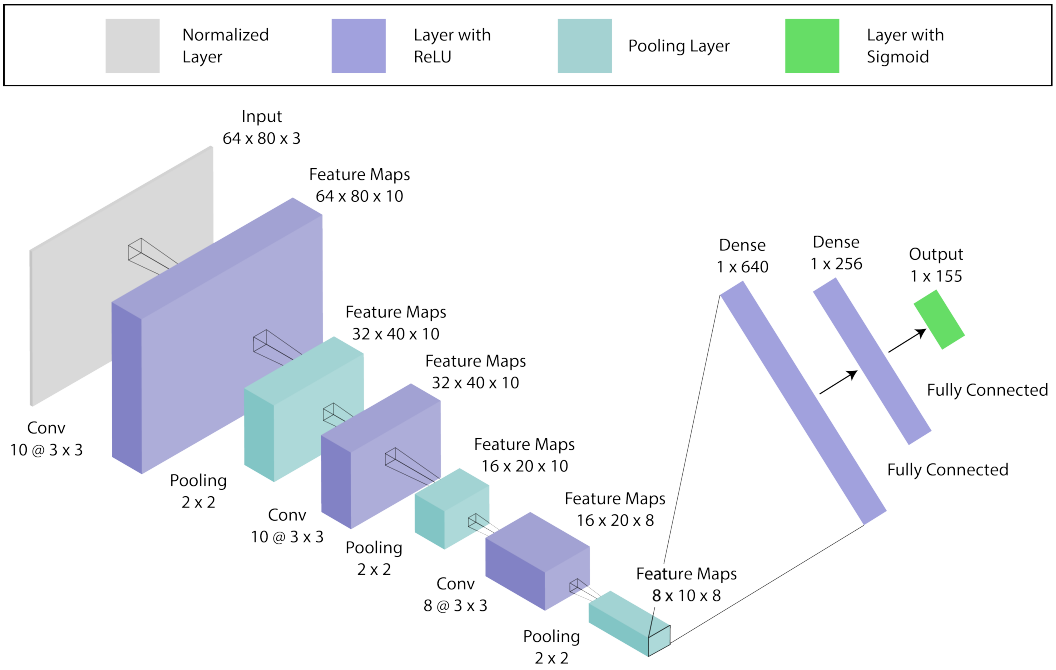


Figure 3.7. Convnet architecture

shadow.

3.2.3 Network Architecture and Training

We use a convolutional neural network, with input shape 64 x 80 x 3 and output shape 1 x 155. Namely, the outputs consist of one binary label for each of the five sensors, for each distance in the set $\{0cm, 1cm, \ldots, 30cm\}$. The architecture is a simple LeNet-like architecture [88] with interleaved convolutional and max-pooling layers, followed by two fully connected layers. The architecture is detailed in Figure 3.7. The model is trained for a total of 15 epochs with 1000 steps per epoch, using gradient descent on mini-batches composed by 64 instances; we use the Adam [83] optimizer with a learning rate $\eta = 0.0002$; the loss function is the mean squared error computed only on the available labels.

Since our dataset has incomplete labels (meaning that for a given instance only a subset of the labels might be known), we adopt a masking approach to prevent the loss function to be influenced by the outputs corresponding to the labels that are missing [51, 193]; in turn, this ensures that the corresponding errors will not be backpropagated, which would compromise the learning process.

To achieve this, for each instance we build a binary mask containing one value per label; each value in the mask is equal to 1 if the corresponding label is known, and equal to 0 if the label is unknown. During the forward propagation step, this mask is multiplied element-wise with the difference between the prediction and the ground truth for each output, i.e., the error signal. This nullifies the error where the mask is 0 (i.e., for the subset of labels which are unknown for the given instance), and lets it propagate where the mask is 1.

Performance metrics We evaluate the performance of the model by computing the area under the receiver operating characteristic curve (AUC) for every output label (corresponding to a distance-sensor pair). This metric is evaluated over 100 rounds of bootstrapping [9] to robustly estimate a mean value and a confidence interval. Because of the heavy class imbalance in the dataset (see Figure 3.6), accuracy is not a meaningful metric in this context; instead, AUC is more robust to class imbalance and does not depend on a choice of threshold. In particular, an AUC value of 0.5 provides a clear baseline: in fact, it corresponds to the performance of a baseline classifier that always returns the most frequent class; conversely, an AUC value of 1.0 corresponds to an ideal classifier. We use these fixed bounds in all our figures.

3.2.4 Self-Supervised Occupancy Map Estimation Results

We report two sets of experiments. We begin by quantifying the prediction quality using the datasets described above, acquired on the Mighty Thymio. After that, we aim to test the robustness of the approach: to this end, we consider a model trained on these datasets and report qualitative results for testing on video streams acquired in different settings.

We consider a model trained on scenarios 1 to 8, and we report the results on testing data from scenarios 9 and 10. Figure 3.8 reports the AUC values obtained for each sensor and distance. Figure 3.9 reports the same data as a function of distance, separately for the central and lateral sensors. Note that "distance" here does not refer to the distance between the obstacle and the front of the robot; instead, it refers to the distance of the corresponding target pose as defined in Section 3.1.2, which corresponds to the distance that the robot would have to travel straight ahead before the proximity sensor is able to perceive the obstacle.

We observe that overall prediction quality decreases with distance. This is expected for two reasons: i. the training dataset contains fewer examples at longer distances, and those examples exhibit more extreme class imbalance (Figure 3.6); ii. obstacles at a long range might be harder to see, especially considering the limited input resolution of the network. We also observe that:

- AUC values at very short distances (0cm, 1cm, 2cm) are slightly but consistently lower than the AUC observed between 4cm and 8cm. This is caused by the fact that when obstacles are very close to the robot, they cover almost the whole camera field of view, and there might be no floor visible at the bottom of the image; then, it is harder for the model to interpret the resulting image.
- AUC values dramatically drop to 0.5 (i.e., no predictive power) for distance values larger than 28cm. This is expected, since this value corresponds to the distance of obstacles when they appear at the top edge of the image; an obstacle that is placed farther than that will not be visible in the camera frame.
- For distances lower than 10cm the central sensor is significantly easier to predict than lateral sensors. This is explained by the fact that objects that are detected by lateral sensors are at the edge of the camera field of view when they are close, but not when they are far away.

In the right plot of Figure 3.9 we report the AUC values obtained for each sensor, separately for each testing environment. These values have been obtained by a leave-one-scenario-out cross validation scheme. We observe that the predictive power of the model is heavily dependent on the specific scenario. In particular, scenarios 9 and 10, which were used as a testing set for the experiments above, are in fact harder than the average.

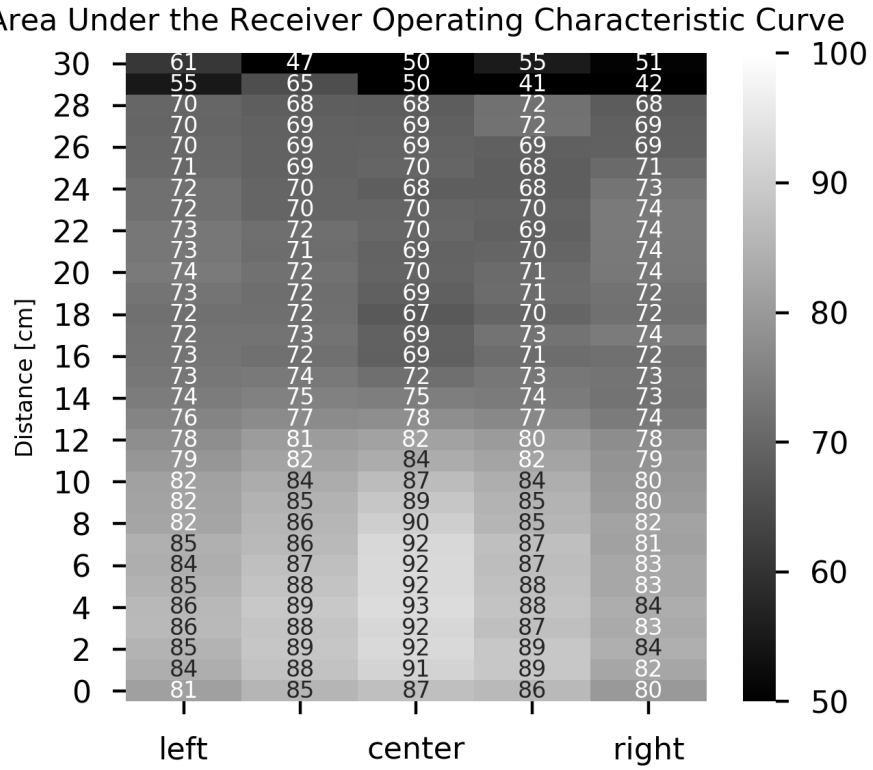


Figure 3.8. AUC value obtained for each sensor (column) and distance (row).

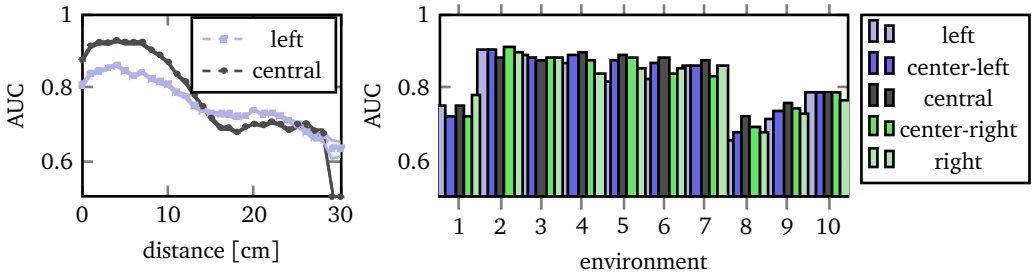


Figure 3.9. Average AUC over 100 bootstrap rounds. Left: AUC for the center (black) and left (cyan) sensors as a function of distance. Shaded areas report 95% confidence intervals on the mean value (dashed line) over all environments. Right: AUC for each sensor for each environment, averaged over all distances between 0 and 30cm.

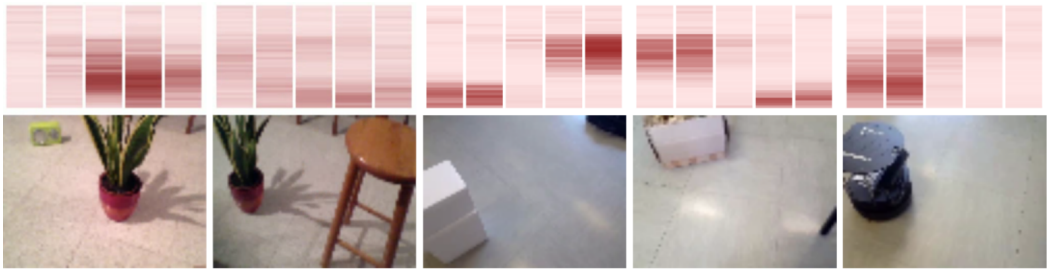


Figure 3.10. Robustness tests: input (bottom images) and outputs (top heatmap) of the model trained on datasets acquired by the Mighty Thymio robot. Leftmost two images are acquired by a TurtleBot 2; the remaining three images are acquired by a belt-mounted camera.

3.2.5 Robustness Test and Control

Figure 3.10 reports qualitative results concerning the performance of the model, trained on the whole dataset described above, when used for inference in two setups which do not match the training data. We run the model on the video stream from a TurtleBot 2 [1] robot, acquired by a laptop webcam mounted about 60cm over the ground (compare with the Mighty Thymio camera, which is 12cm from the ground), and oriented with a similar pitch as the Mighty Thymio camera. Because the robot has no proximity sensors, we don't have ground truth information; still, we qualitatively observe that obstacles are detected reliably. The same figure reports the results we obtain when feeding the model with data coming from a webcam mounted on the belt of an user during walking (height 95cm, variable pitch). Videos are available as supplementary material, and also include a brief experiment showing the effects of extreme camera pitch angles. Supplementary videos show the system used as the sole input of an obstacle avoidance controller, both on the Mighty Thymio robot (with disabled proximity sensors) and on the TurtleBot 2 robot; the robots react to obstacles appropriately.

3.2.6 Simulated Experiment on Terrain Properties Estimation

In order to highlight the generality of the approach, we run an additional experiment using a Pioneer 3-AT platform simulated in Gazebo (see Figure 3.11), equipped with 3 RGB cameras looking at random angles (long-range sensor) and a single short-range sensor observing the floor color just below the robot (which returns binary data: bright or dark). Data is collected while the robot moves at a constant linear speed of 0.5ms^{-}1 and every 3 seconds changes its angular speed to a randomly chosen value between $-15°s^{-}1$ and $+15°s^{-}1$. We use 10 large maps with size $50m \times 50m$ each, featuring a planar floor textured in a random procedurally-generated black and white image obtained by thresholding low-frequency Perlin noise. On these maps, we run the controller for a total of 70 simulated minutes, respawning the robot to the center of the area should it get too close to the edge. This results in 84000 training examples collected at 20Hz; examples for 5 maps are used for self-supervised training, the remaining for evaluation.

We consider a set of $17 \times 17 = 289$ target poses $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{289}\}$ in a square grid with a step of 0.5m; because the short-range sensor is not affected by the robot's orientation, we disregard the orientation of the poses and depict them as small circles; the grid covers an area of $8m \times 8m$ and is horizontally centered on $\boldsymbol{p}(t)$; it extends to 5m in front and 3m behind $\boldsymbol{p}(t)$. The task is

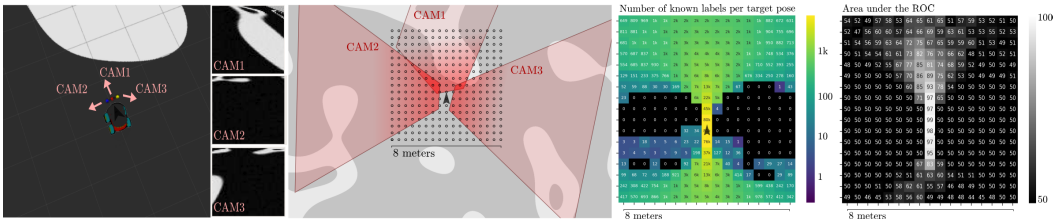


Figure 3.11. From left to right: the simulated Pioneer 3-AT platform on one of the 10 random maps; images acquired by the three cameras; top-down view with the grid of target poses and the exact area of floor seen by each camera depicted in red (note that CAM1 is tilted laterally, so its imaged area is not a trapezoid); log-scale heatmap of the number of known labels per target pose in the training set; heatmap of the AUC on the testing set for each target pose.

to predict the color of the floor (dark or bright) that the robot would measure at $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{289}$ given the three camera images acquired at $\boldsymbol{p}(t)$.

The results on the right of Figure 3.11 show that the approach learns to predict the output of short-range sensors for generic target poses, including those not on the robot's longitudinal axis, as long as the pose is visited often (i.e., its label is known in a sufficient number of training instances). Interestingly, the approach learns to predict even some target poses that are *not directly observed* by any of the three cameras; for example, the poses directly under the robot and up to two meters behind it. Note that this can not be due to the short-range sensor or its history, because the predictions are a function of a single input: the long-range sensor readings at the current timestep. Instead, the model has learned to exploit the fact that bright and dark areas in the floor are smooth and vary with low spatial frequency: this makes it possible to extrapolate the floor color on poses behind the robot, as long as the true labels for these poses are observed frequently in the training set.

3.3 Discussion

We presented a self-supervised approach that learns how to predict future and past outputs of an informative short-range sensor by interpreting the current outputs of a long range sensor, which might be high-dimensional and hard to interpret. We implemented the approach on the Mighty Thymio robot for the specific task of predicting the future outputs of the robot's proximity sensors (i.e., the presence of obstacles at different distances from the robot) from the video stream of the robot's forward-pointing camera. We quantitatively verified that the approach is effective and generalizes well to unseen scenarios: we qualitatively evaluated robustness to different operating conditions and usage as input to an obstacle-avoidance controller. Finally, we successfully instantiated the approach on a different, complementary task in simulation.

The presented approach is effective, however, it lacks in efficiency: much of the training samples have a low percentage of known cells inside the map constituting the label for a given image. Due to this missing information, we adopted a masked loss to train models using only map cells with known values while ignoring remaining cells. In the following chapter we discuss how to take full advantage of the collected data, supervising models also on those cells having missing values using a consistency loss.

Chapter 4

Learning From Partial Labels

In the previous chapter, we investigated a self-supervised approach to automatically label camera frames with occupancy maps, leveraging sensors' readings acquired at different time steps. One crucial detail of the generated labels is that they are dependant on robot exploration: by choosing to head one direction instead of another, the robot explores some areas of the environment, leaving the others untouched. This translates in our approach to cells of the occupancy map with no known label value, since the robot did not sense that specific area. In this chapter, we explore how to exploit unknown map cells to provide additional supervision to a perception model. Specifically, we take advantage of the fact that the environment state changes smoothly and not abruptly to define a consistency loss. Apart from providing additional supervision, this loss additionally introduces a desirable bias by forcing a model to have consistent predictions from different views of the same area of the environment.

4.1 Background

We consider a class of perception problems in which a robot has to interpret data acquired by onboard sensors (e.g., cameras) in order to derive spatial information about its environment state (e.g., the presence of obstacles or the pose of a person in front of the robot). In the ideal case, one learns a model in a supervised setting by using a large dataset which maps inputs to known desired outputs. In many real-world cases, however, one can acquire a training set which only features *partial* ground truth information. Two reasons why this might occur are the following.

- In semi-supervised learning [27], one acquires (usually at a significant cost) a limited amount of labeled training instances, but also wants to exploit a large amount of unlabeled instances; those can be acquired with limited effort, or in environments more representative of the deployment scenario; for example, this is the case when labels are generated by motion-tracking systems, which can only be used in some environments; or when a robot is expected to automatically adapt to a deployment environment, where supervisory labels are not available.
- When learning multi-output models, each training instance might have ground truth information only for a (potentially different) subset of the outputs; in particular, this is relevant in one type of self-supervised robot learning [204, 60, 113, 191], in which a robot

autonomously acquires its own training set by using readings of different sensors, possibly when the robot is at different poses, to determine ground-truth labels. Depending on the robot behavior, every training sample might only have ground truth information for some of the outputs.

The naive solution to the problem is to ignore the missing labels while training. This is trivially implemented when using neural models that minimize a task loss: in the former case by defining a task loss only on the labeled instances; and in the latter by using a masked task loss [51, 193] which avoids backpropagating the error from the outputs with unknown labels.

This paper formalizes, demonstrates and evaluates a simple but general approach to exploit unlabeled information in this context. In particular, we keep the task loss defined above, and associate an auxiliary state-consistency loss: it captures the prior expectation that the environment state does not change over time. In dynamic environments, where the state does change (but in a continuous way), we assume that state-consistency applies within time windows that are short enough, i.e., by sampling at a sufficient frequency, the change in state between sequential samples is negligible.

This expectation can be interpreted as a rough translation to robotics of the *Object-Permanence hyperprior* [34], a well-known concept in developmental psychology [57] that describes our brain's (innate or learned) expectation that objects continue to exist even when they are not directly perceived by the senses. According to the Predictive Coding theory of brain function [33], this hyperprior yields a powerful supervisory signal that helps the process of learning to make sense of sensory information[1].

Our main contribution is a general approach to handle spatial perception problems with partial labels. In Section 4.2, we directly instantiate the approach for two different realistic robotic visual perception tasks, as depicted in Figure 4.1, and report results on both tasks with simulated and real data, quantifying improvements, and exploring how the approach performs in dynamic environments.

4.1.1 Related Work

Multi-view consistency approaches exploit different views of the same scene as a source of supervision, e.g., allowing a model to better predict the overall shape of an object and its pose given only a partial representation, possibly with occluded areas; other applications of multi-view consistency are discussed in greater detail in Chapter 2. Given predictions generated from different views of the same object, model consistency is achieved either by construction, aggregating the predictions [200], or by penalization, using a consistency term [176, 189]. Our approach shares the main idea of the latter class, multi-view consistency, but is not limited to a specific sensor type or to operating on individual objects; instead, it supports a generic representation of the environment state, and can be adopted on any sensing modality. In particular, Zeng et al. [200] solve the 3D pose estimation task for a robot arm using multiple RGB-D cameras. Self-supervised learning is used to collect large quantities of RGB-D images containing single objects, which are easily labeled by a background-removal algorithm. Multi-view consistency is achieved by projecting the segmented objects from different views into a shared 3D space and then combine them together. Next a template matching algorithm tries to fit a model of

[1] Another hyperprior described in developmental psychology is *sensory synchronicity*: the expectation that perceptions from different senses could be caused by the same physical object in the world, and thus can be used to predict each other: this is the core idea behind cross-modal self-supervised learning [80], a very active research field in deep learning.

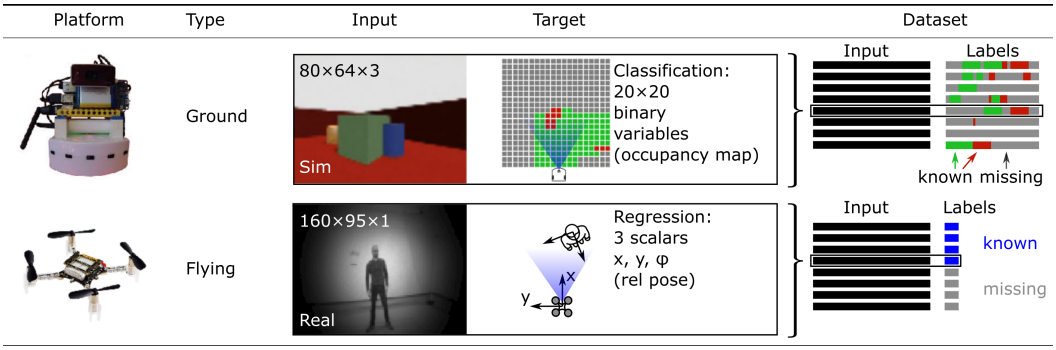


Figure 4.1. We demonstrate our general approach on two different spatial perception tasks. On the first row, a differential drive robot takes RGB images to predict incomplete state observations. On the second row, a tiny drone learns state estimation from grayscale images and a partially labeled dataset. Videos, datasets and code available at `https://github.com/idsia-robotics/state-consistency-loss`.

the object, obtaining the most probable 3D pose. Tulsiani et al. [176] and Wei et al. [189] use single-view RGB images of the same object taken from different positions to predict the 3D shape of the object and it's pose. During training multiple views of the same object are fed into the network and the predicted shapes are forced to be equal by a consistency term: Tulsiani et al. [176] compute the expected cost of each voxel being correctly represented in a depth image taken from the point of view of another input instance, while Wei et al. [189] adopt the chamfer distance computed over pairs of predictions belonging to the same object seen from different points of view. In contrast, our loss function does not focus on shape prediction and enforces consistency over all state observations, where overlapping.

Global consistency One important challenge for Simultaneous Localization and Mapping (SLAM) techniques is to generate globally-consistent maps: this is tackled by optimizing the global trajectory at each timestep and consequently updating the map [44], by limiting the optimization step to consider only a handful of keyframes [69] or by maintaining a collection of overlapping submaps [146]. While focusing on a different task (model-based perception), our approach shares with those mentioned above the common goal of a consistent state representation, enforced by means of constraints placed upon the optimization problem.

4.1.2 Model

We consider a mobile robot capable of odometry, in an environment whose state $E(t)$ evolves over time. Let $\boldsymbol{p}_t$ be the pose of the robot at time t and $\boldsymbol{x}(t)$ the outputs of one or more onboard sensors: our goal is to train a model to interpret $\boldsymbol{x}(t)$ in order to predict $\boldsymbol{y}(t) \subseteq E(t)$, which is multi-dimensional (i.e., the model predicts multiple target variables). As an example, consider a wheeled robot tasked to detect and avoid obstacles. Here, a sensible choice would be to represent $E(\cdot)$ as an occupancy map and $\boldsymbol{y}(\cdot)$ as the subset of the map local to the robot.

In a standard supervised learning setup, we would collect a training set of instances, each of which is composed by a pair: inputs $\boldsymbol{x}(t)$ and ground truth outputs $\boldsymbol{y}(t)$. We consider a more general scenario, often occurring in self-supervised robotics and real-world robot deployments,

in which supervisory information is derived by an additional (onboard or offboard) sensor $\boldsymbol{s}(t)$ from which we can compute ground truth for only *a subset* of the target variables of the training set; in particular, we might know ground truth information for a subset of target variables for every instance, and/or for a subset of the training instances (semi-supervised learning).

We learn the model by minimizing a loss function $\mathcal{L}$ defined as the weighted sum of two terms.

- A task loss $\mathcal{L}_{\text{task}}$, which penalizes the distance between the model's prediction $\hat{\boldsymbol{y}}(\cdot)$ and the ground truth $\boldsymbol{y}(\cdot)$; if ground truth is known for only part of the target variables, one can use a masked loss [193], such that unknown target variables for a given instance will not contribute to the loss

- A state-consistency loss $\mathcal{L}_{\text{sc}}$ which penalizes inconsistent predictions at different times. If we assume that the environment is dynamic, $\mathcal{L}_{\text{sc}}$ can be enforced only between instances acquired within a (possibly very short) time window δ_t within which we expect the evolution of the environment to be negligible

Let B be the batch of training instances and P be the set of all pairs of different state observations that lie within such a time window, $P = \{(\hat{y}(i), \hat{y}(j)) | \text{abs}(i-j) < \delta_t\}$; the loss can be written as the mean squared difference on P:

$$\mathcal{L}_{\text{sc}}(P) = \frac{1}{\|P\|} \sum_{\hat{\boldsymbol{y}}(i),\hat{\boldsymbol{y}}(j)\in P} (\hat{\boldsymbol{y}}(i) - \hat{\boldsymbol{y}}(j))^2 \tag{4.1}$$

The overall loss function can be written as $\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{sc}}\mathcal{L}_{\text{sc}}$; the scaling factor λ_{sc} is set such that the two losses are balanced, optimizing both task and state-consistency terms at the same time. In particular, if $\mathcal{L}_{\text{task}}$ is negligible with respect to $\lambda_{\text{sc}}\mathcal{L}_{\text{sc}}$, the model would trivially learn to predict an arbitrary, constant output independent on the input.

4.2 Experiments

We instantiate the general approach described above on two very different tasks and platforms (Figure 4.1): one estimates a robot-centered occupancy map given the camera feed of a differential-drive robot, similarly to the approach described in Chapter 3, and the latter predicts the user pose w.r.t. a drone given the camera feed.

4.2.1 Self-Supervised Occupancy Map Estimation Setup

We consider a small differential-drive robot [111, 66] equipped with front-facing camera $\boldsymbol{x}(\cdot)$ and five short-range infrared proximity sensors $\boldsymbol{s}(\cdot)$, moving in a static environment $E(\cdot)$ consisting of an enclosed floor with some obstacles. The task consists in predicting a robot-centered obstacle occupancy map $\boldsymbol{y}(t) \subset E(t)$, given a camera frame $\boldsymbol{x}(t)$ as input.

The cells of the map lie on a regular 20×20 grid defined by a set of 2D coordinates $\mathcal{P}$: the model solves a 400-variable binary classification problem and predicts in $\hat{y}(\cdot)$ the probability of an object being present in each cell of the map.

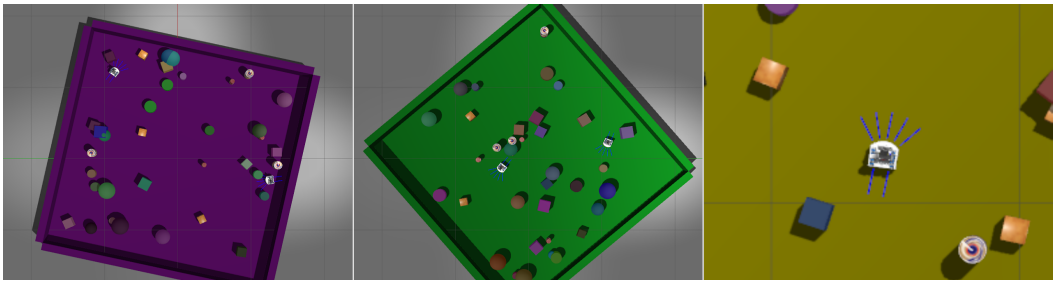


Figure 4.3. Self-supervised occupancy map estimation: top-view of two randomly generated environments and a close-up view of the robot.

The model is trained in a self-supervised way by autonomously generating a training set while the robot randomly explores the environment: for a given image $\boldsymbol{x}(t)$ acquired at time t, we compute ground truth for a subset of the cells in $\boldsymbol{y}(t)$ by considering the infrared proximity sensors' readings $\boldsymbol{s}(k)$ acquired from different poses at times $k \neq t$. In particular, given a pose $\boldsymbol{p}_k$ relative to $\boldsymbol{p}_t$, one can derive constraints on $\boldsymbol{y}(t)$ from the proximity sensor readings $\boldsymbol{s}(k)$ as shown in Figure 4.2: for example, if $\boldsymbol{p}_2$ is 20 cm in front of $\boldsymbol{p}_1$, and the center proximity sensor does not detect any obstacle within 1 cm, we can enforce that there is no obstacle in a cell 21 cm in front of $\boldsymbol{p}_1$. By considering all k in a time window centered on t, we acquire ground truth information for a small subset of the cells of $\boldsymbol{y}(t)$: free cells are labeled with 0, occupied cells with 1 and unknown cells with -1, depicted respectively as green, red and gray in Figure 4.1[2]. A training example is generated from each timestep t and represented by the pair $(\boldsymbol{x}(t), \boldsymbol{y}(t))$.

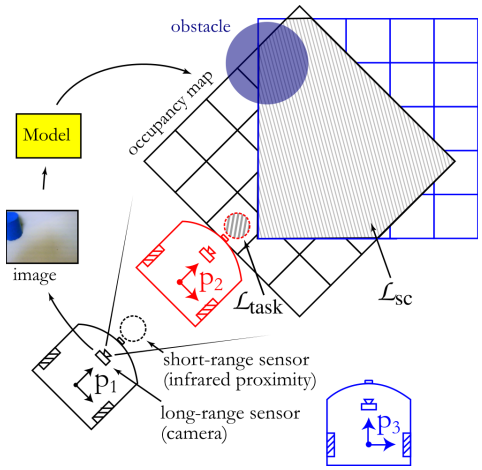


Figure 4.2. Self-supervised occupancy map estimation: given an image acquired at $\boldsymbol{p}_1$, partial obstacle presence ground truth is computed by considering short-range infra-red sensor readings acquired when the robot is at different poses (e.g. $\boldsymbol{p}_2$). Additionally, the state-consistency loss enforces that predictions from pairs of poses (e.g. $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$) are consistent where they spatially overlap.

Simulated environments

The setup consists in randomly generated environments, similar to Tobin et al. [171, 172], with varying textured floors, object placement, size and texture properties such as ambient and specular color, light direction and intensity. Objects are boxes, cylinders, spheres and additionally models of various real-world objects (see Figure 4.3). The whole environment is 3 x 3m large and is enclosed by walls having same texture as the floor.

We use Gazebo [84] to simulate 20 randomly-generated environments. In each environ-

[2] Cells with contradicting ground truth are assigned the label (0 or 1) observed most frequently.

ment, we simulate 64 episodes with an average duration of 14 seconds. In each episode, the thymio is spawned at a random pose within the enclosure. A controller initially moves the robot forward until it gets close to an object; then it rotates in one of the two directions for a random time and starts moving forward again; the process is iterated four times; then, the robot is spawned at a new pose so as to yield a good coverage of the environment. This results in a total of 5 hours of simulated time, from which we sample 50k instances. Each instance consists of an RGB image with size 80 x 64 pixels and the corresponding partial occupancy map covering an area of 80 x 80cm with a step size of 4cm (400 variables). 30k instances from 10 environments are used as the training set $\mathcal{T}$, and 20k instances from the 10 remaining environments are used for the testing set $\mathcal{Q}$.

Loss function

The formulation of the proposed loss function is reported in Equation (4.1). For the task term we adopt a masked MSE loss that ignores missing labels: the cells of the occupancy map for which the ground truth is unknown are skipped when computing the mean squared error. For the state-consistency loss we used a value of $\delta_t = 10$s and $\lambda_{\text{sc}} = 1$. In a perfectly static environment with perfect sensing, one could use a $\delta_t = \infty$; however in this case, a value of 10s has been chosen in order to limit errors caused by odometry drift. The value of λ_{sc} has been chosen among the set of candidates $\{10, 5, 2, 1, 0.5, 0.2, 0.1\}$ as the one resulting in the lowest validation loss.

4.2.2 Semi-Supervised Estimation of User Pose in a Nanodrone Setup

We consider a Crazyflie 2.1, a nano-drone measuring 10cm in diameter and weighing only 27g, fitted with a forward-looking monocular camera $\boldsymbol{x}(\cdot)$ and a Parallel Ultra-Low Power (PULP) deep learning expansion board [123]. The camera acquires gray-scale images that are cropped to a size of 160 x 96 pixels. The task is a 3-variable regression problem: predicting the relative pose $\boldsymbol{y}(\cdot)$ consisting of frontal displacement x, lateral displacement y, and relative head rotation ϕ, of a user moving in front of the drone [102].

Datasets

We consider a training set set $\mathcal{T}$ of 60k instances, collected on 5 subjects, composed of two distinct subsets: one subset with 30k instances, where an external motion tracking system produced ground truth information $\boldsymbol{s}(\cdot)$ for $\boldsymbol{y}(\cdot)$ and a second subset of 30k instances which we pretend has no ground truth. This identifies a semi-supervised problem, where only $\mathcal{L}_{\text{task}}$ is used for the former dataset, and only $\mathcal{L}_{\text{sc}}$ is used for the latter. 10k additional labeled instances are used for the validation set set $\mathcal{V}$.

The testing set set $\mathcal{Q}$ is composed by 30k additional labeled samples collected from 5 other subjects disjoint from those in the training and validation sets.

Loss function

The end task loss $\mathcal{L}_{\text{task}}$ is defined as the mean absolute error between $\hat{\boldsymbol{y}}(\cdot)$ and $\boldsymbol{y}(\cdot)$. In contrast, $\mathcal{L}_{\text{sc}}$, as defined in (4.1), penalizes the motion of the user in a *fixed* reference frame, i.e., discounting the motion of the drone itself, which is known from the drone's odometry. Therefore, the $\mathcal{L}_{\text{sc}}$ loss captures the intuitive prior that, in an unlabeled dataset, within a short time frame

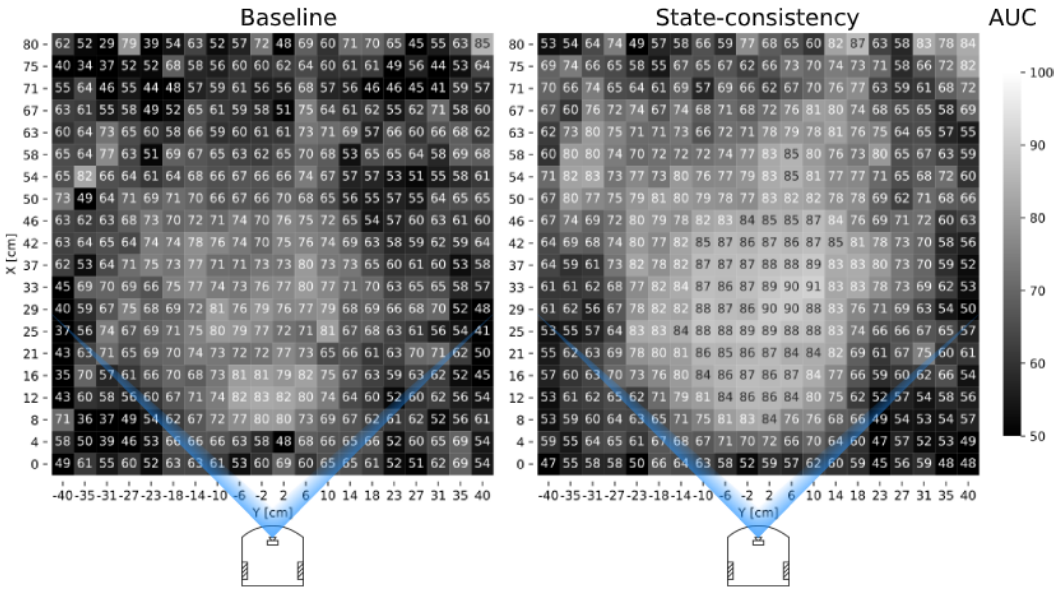


Figure 4.4. Self-supervised occupancy map estimation: AUC of the occupancy maps predicted from held-out testing data. The lighter the color the better the performance. Left, baseline model trained using only the task loss (masked MSE) [113]. Right, model trained using task and state-consistency losses.

the pose of the user should not change dramatically, i.e., that the user is subject to the laws of physics.

4.2.3 Network Architectures and Training

Both models are small resnet-like convolutional architectures [70], with approximately 300k parameters. We adopt early stopping in order to select the best performing model on the validation set. Both networks are trained with mini-batch gradient descent, using the combination of task and state-consistency losses with a weight $\lambda_{sc} = 1$ (kept fixed throughout all experiments), Adam [83] as the optimizer and a learning rate $\eta = 5 * 10^{-5}$. Each batch is composed of temporally-close training instances in order to maximize the pairs contained in the set P, used to compute the state-consistency loss.

4.2.4 Self-Supervised Occupancy Map Estimation Results

On the held-out testing set, we compare in Figure 4.4 the performance of a model trained as described above with an identical model trained on the same data with $\lambda_{sc} = 0$, which disables the proposed state-consistency loss and serves as a baseline. Because this is a multi-label binary classification problem, we evaluate each model by computing the Area Under the ROC Curve (AUC) for each output variable. The metric is independent on the choice of threshold: a value of 50% implies that the model does not have any predictive power, whereas a value of 100% denotes an ideal model.

The performance of both models is best in the central part of the occupancy map, which

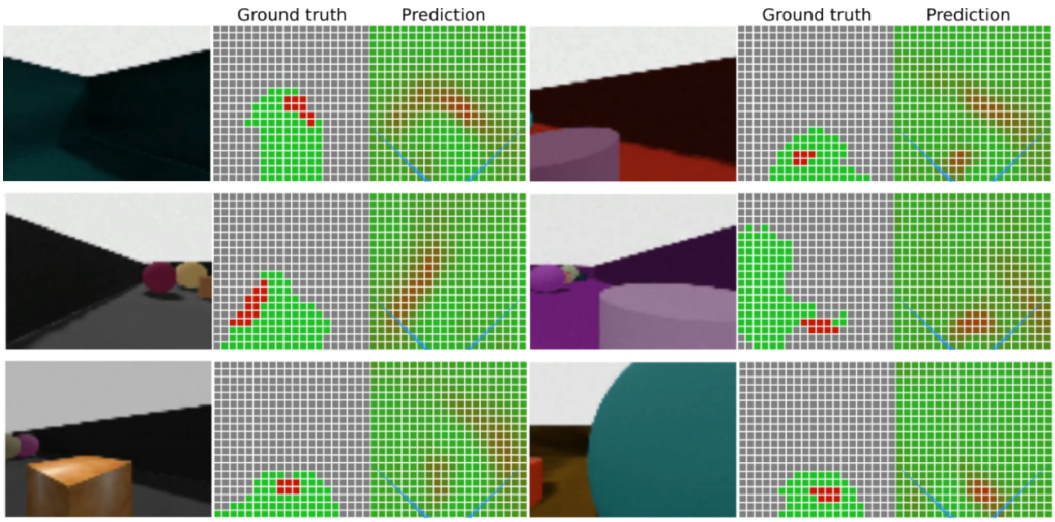


Figure 4.5. Self-supervised occupancy map estimation: on six testing instances. Left: input before down-scaling. Center: self-supervised labels (not used for prediction). Right: model prediction and FOV (blue). Red represents occupied cells, green for empty cells, and gray for missing information.

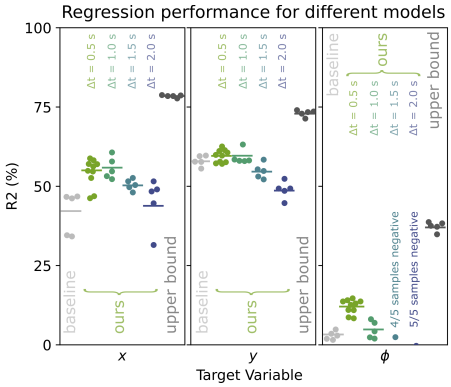
is well visible in the camera image; it decreases towards the sides due to the limited field of view of the camera – with objects being invisible in the frame, especially at short distances. Interestingly, the models yield (weakly) informative estimates even for cells at the bottom left and bottom right of the map, which are well outside the camera field of view; this occurs because the model captures regularities in the environment – such as the common size of some objects – and is therefore able to sometimes extrapolate to unseen parts of the map, e.g., if just a part of an object is visible on one side. The score is lower the further we get because of the limited resolution available to represent distant objects, and for the much smaller amount of self-supervised labels available for far cells of the map.

For our model, the average AUC computed over all outputs is 69%. This is significantly higher than the model trained solely with the task loss [113], which only scores 63%; the difference is particularly striking if one considers that in most frames, a large number of cells can not be estimated; therefore, the upper-bound for the AUC is likely to be well below 100%.

Qualitative analysis

In Figure 4.5 six different testing-set instances are displayed along with the corresponding network's prediction. The model has successfully learned to identify walls as narrow and long structures, even when only a small part of the wall is visible, and corners as two walls intercepting at a 90° angle. It correctly detects multiple solid-coloured and textured objects, regardless of size and point of view, especially when those are located near the robot. The model has learned that structures like walls not only exist in the part of the map for which labels are provided, but extend beyond those, correctly predicting that they continue in a straight line; furthermore, when faced with corners, the wall extend in the direction opposite from such corner and not in the other way.

Figure 4.6. Semi-supervised estimation of user pose in a nano-drone: consecutive frames with the user standing up, changing direction and walking. In such a dynamic environment, our approach is applied using small time windows.



Model	R^2 (%) x	y	ϕ
Baseline	41.7 ± 7.0	58.0 ± 1.0	3.0 ± 1.4
Ours ($\delta_t = 0.5$s)	54.2 ± 5.0	**59.1** ± 1.9	**13.1** ± 2.0
Ours ($\delta_t = 1.0$s)	**55.7** ± 3.5	**59.1** ± 2.2	4.7 ± 2.7
Ours ($\delta_t = 1.5$s)	50.5 ± 1.7	54.8 ± 2.4	-3.2 ± 3.6
Ours ($\delta_t = 2.0$s)	45.0 ± 7.9	48.8 ± 2.7	-4.3 ± 3.5
Upper-bound	78.4 ± 0.6	73.2 ± 1.0	37.3 ± 1.3
Abs. Improv.	+12.4	+1.1	+10.1
Rel. Improv. [%]	+33.8	+7.2	+29.4

Figure 4.7. Semi-supervised estimation of user pose in a nano-drone: R^2 computed for a model trained using only the task loss [102] (baseline), our model trained using task and state-consistency losses using different values of δ_t, and the upper-bound; we report absolute improvements of our approach ($\delta_t = 0.5$) vs the baseline, and improvements rescaled such that baseline performance is 0% and upper-bound performance is 100%.

4.2.5 Semi-Supervised Estimation of User Pose in a Nanodrone Results

We train the proposed approach using $\delta_t \in \{0.5, 1.0, 1.5, 2.0\}$ seconds; we compare the results of these models on the held-out testing set with two alternatives: one baseline model trained with $\lambda_{sc} = 0$ (equivalent to using only labeled data as in [102]); and a model acting as an upper-bound of the achievable performance, obtained by training on all 60k instances, exploiting also the labels of the unlabeled dataset.

Figure 4.7 reports the average coefficient of determination (R^2) and its standard deviation computed over 5 replicas of the experiment (10 for $\delta_t = 0.5$), for each of the three output variables. The R^2 metric quantifies the fraction of the variance of the target variable that is explained by the model, and ranges between 0% (for a model that trivially predicts the average of the testing set) and 100% (for an ideal model); this metric does not depend on the unit of measure and is more interpretable (but related to) the MSE. We quantify the improvement of

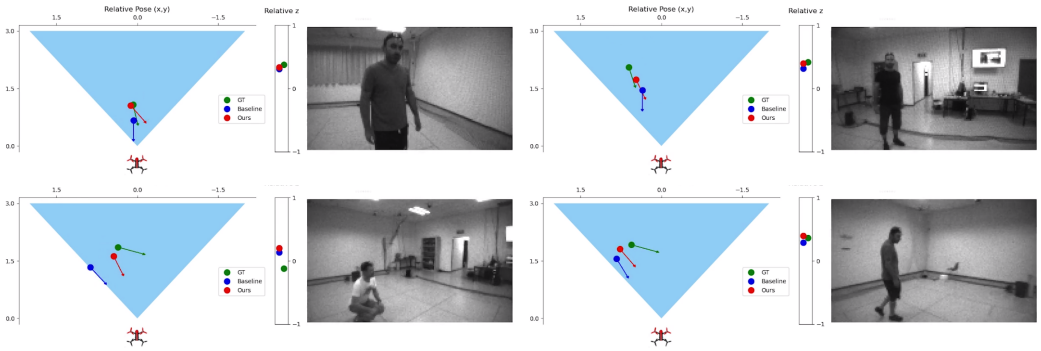


Figure 4.8. Semi-supervised estimation of user pose in a nano-drone: user's head location (x, y, z) and heading (ϕ) predictions. Compared to the model trained using only the task loss [102] (blue), our approach with $\delta_t = 0.5$s (red) is usually closer to the ground-truth (green).

our model over the baseline model, trained using only the task loss [102], both in absolute terms (R^2 points) and, more importantly, as percentage relative to the maximum achievable improvement; this is computed as (model − baseline)/(upper − baseline) and yields 0% for a model that performs as well as the baseline, and 100% for a model that performs as well as the upper-bound.

On this metric, the model trained with the state-consistency loss and $\delta_t = 0.5$s yields statistically-significant improvements over the baseline (evaluated with the one-sided Welch's t-test [190]) for x (33% relative improvement, $p = 0.007$), and ϕ (29% rel. improvement, $p = 0.00002$), but not for y (7% rel. improvement, $p = 0.15$). This is explained as x and ϕ represent harder pattern recognition problems compared to the estimation of the lateral displacement y.

Models trained on larger δ_t values assume that the environment is approximately static over longer time periods; if the assumption is met, this should improve the model as stronger state consistency priors are exploited during training; however, if the assumption is violated the state consistency loss introduces noise and hinders training. In our dynamic environment, we observe this trade-off over different target variables. In particular, x (the distance of the subject to the camera) changes most slowly: in this case, intermediate values for δ_t maximize performance. y tends to change faster than x, because it is heavily affected by rotational motion of the camera; ϕ, corresponding to head rotation, changes most abruptly. On y and ϕ, larger values of δ_t hinder performance and make the model even worse than the baseline.

Implementation and qualitative performance

Figure 4.8 shows a comparison of the model's prediction against a baseline model trained using only the task loss [102] and ground-truth. The model is in general more accurate than the baseline, especially for x and y, while has some difficulties in estimating ϕ and the user's height z, especially when the user is crouching. Figure 4.9 shows the Crazyflie nano-drone driven by a controller which tracks never-seen-before users at close range; the controller is fed with the outputs of the model trained using the proposed approach, and controls the robot position and yaw in order to stay 1.5 m in front of the user's head, pointing towards it. Height is fixed to the average head height (1.5 m), which is the same height used for training set acquisition. The

Figure 4.9. The Crazyflie drone tracking the user's movements with a perception model learned using the state-consistency loss.

whole pipeline runs onboard the drone using the commercially available AI-deck [121], based on the PULP heterogeneous model [38]. The improvements in R^2 performance on the testing set maps to a noticeable subjective improvement in tracking quality. Similarly, the upper-bound model trained using ground-truth labels for the whole dataset performs noticeably better than our model.

4.3 Discussion

We proposed a general state-consistency loss enforcing a prior that the robot environment changes smoothly; the loss is used in addition to a task loss for self-supervised or semi-supervised spatial perception problems, and allows to learn from portions of the dataset for which labels are not available. We instantiate the approach in two very different visual perception tasks, and measure statistically-significant improvements which have noticeable effects in observed robot behavior. The success of the approach depends on the proper choice for the size of the time window δ_t and on the quality of the robot's odometry. The time window can be easily tuned through trial and error, grid search, or using an optimization strategy such as bayesian optimization [132].

The reliance of the approach on odometry causes the model's performance to decrease proportionally to the noise affecting the odometry itself. In the next chapter, we address this issue by considering odometry and sensors as being affected by noise, thus modeling them as random variables. Then, task and state-consistency losses are updated into what we call an uncertainty-aware formulation, using Monte Carlo to approximate the distribution of the losses minimized during training.

Chapter 5

Learning From Sparse Noisy Labels

In the previous two chapters, we studied self-supervised robot learning approaches that leverage the combination of sensors' readings and the robot's odometry to generate labels. Thus far, the approaches assumed the sensors and odometry to be ideal, i.e., not affected by noise or that its contribution is negligible. However, this assumption falls short when looking at realistic sensors that, due to fabrication errors and operational conditions, are almost always affected by noise. In this chapter, we take noise into account and explore a simple approach to handle it. We assume to know the model of the noise affecting sensors, and represent it with normal distributions. This approach does not require any alteration to the chosen deep neural network and can be customized to facilitate any loss function. The approach is applied to three increasingly challenging scenarios and different tasks to test its effectiveness.

5.1 Background

Many robot perception tasks consist in interpreting sensor readings to extract high-level spatial information [80], such as the pose of an OoI with respect to the robot, or the pose of the robot itself in the environment. When sensors produce noisy, high-dimensional data that is difficult to interpret (e.g., cameras or lidars), a common solution is to rely on supervised learning [162]. In many real-world scenarios, collecting the necessary training sets is a fundamental problem; Self-Supervised Learning (SSL) in robotics aims at equipping robots with the ability to acquire their own training data, e.g., by using additional sensors as source of supervision, without any external assistance. In some cases, this allows robots to acquire training data directly in the deployment environment [163]; a detailed description of these approaches can be found in Chapter 2.

In this context, a state estimator, such as robot's odometry, can be a rich source of information. Odometry allows a robot to estimate its own motion in the environment according to its kinematics, e.g., by integrating over time the motion of its wheels as measured by wheel encoders, often with some uncertainty and accumulating errors. Consider for example a robot capable of odometry, equipped with a camera and a collision detector [60]; after bumping into an object, the robot could reconsider the camera observations from the timesteps preceding the collision; assuming a static obstacle, the camera image acquired when the robot was (according to its own odometry) 1m behind the place of collision, would depict an obstacle at a distance of 1m. This piece of information was acquired by the robot without any explicit external su-

pervision, and can be used for training machine learning models that map acquired images to the spatial position of obstacles. The role of odometry is to leverage sparse information from a simple detector – which provides relevant information only for a few timesteps in a training sequence – to generate an informative labeled training set.

In this paper we generalize and extend this basic idea: we consider a generic robot that has a spatial perception task, such as estimating the pose of an OoI in the environment, is capable of state estimation, possibly affected by accumulating uncertainty due to errors, and is equipped with one or more sensors, whose outputs we want to use to estimate the target pose. Furthermore, the robot is equipped with a detector that, for at least a small fraction of timesteps, produces ground-truth information about the target pose (possibly uncertain). Given training sequences, we want to learn a stateless model that, given the sensor readings, estimates the target pose.

Our main contribution is a formalization of the problem of learning spatial perception tasks in the presence of noisy labels and a general solution based on deep learning, that: i) learns from sporadic supervision given by a detector and a possibly uncertain state estimator; ii) can explicitly account for uncertainty in the state estimates and in the supervision using a Monte Carlo approach; iii) integrates the consistency loss described in Chapter 4 to further improve results, even with the hurdle of uncertainty. In Section 5.2 we investigate the generality of our contribution by instantiating it in three different applications as shown in Table 5.1:

- Estimating the relative 3D pose of an OoI from a camera mounted on a robotic arm manipulator.
- Estimating the heading of a differential drive robot in the vicinity of a straight wall, using data from 7 sensors that measure the amount of infrared light reflected from the environment.
- Estimating the 2D pose of a docking station, using images from a camera mounted on a mecanum robot.

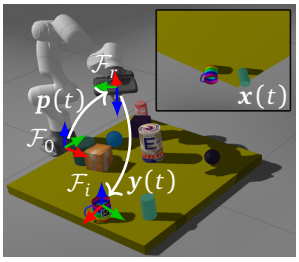
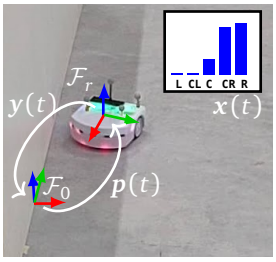
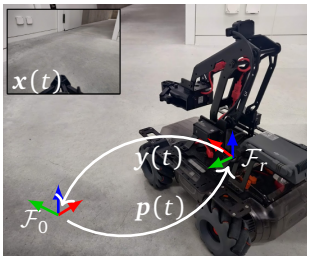
We experimentally evaluate the approach on the three applications and quantify the improvements due to explicitly modeling uncertainty and enforcing state-consistency.

5.1.1 Related Work

In real world scenarios, especially when data is collected directly by robots, noise present in the labels can severely degrade the performance of learned models [162]. Most approaches that do regression from noisy data focus on estimating the uncertainty associated to predictions. These approaches fall under the category of Bayesian learning: some apply an uncertainty estimation model on an existing regressor [99], while others design ad-hoc architectures, casting the problem as learning the distribution of the model weights [19, 59, 192].

Loquercio et al. [99] estimate the uncertainty associated to model predictions by fitting a Bayesian belief network. Dropout is used during multiple forward passes of the network to approximate the uncertainty of the prediction in a Monte Carlo sampling fashion. Similarly, Gal and Ghahramani [59] propose to approximate the true uncertainty by utilizing dropout Monte Carlo samples. In contrast to Loquercio et al. [99], they require to alter the network architecture by placing dropout layers after each non-linearity. By keeping the dropout functionality active during inference, they compute the uncertainty as the variance of the produced samples.

Table 5.1. Platforms and sensory equipment used in the perception tasks

Robot	Simulated Robot Manipulator	Real Diff. Drive Robot	Real Mecanum Robot
			
Sensor	RGB Camera	Infrared Sensors	RGB Camera
Target	Object of Interest 3D Pose	Robot Heading	Docking Station 2D Pose
Detector	Fiducial Marker	Initial Robot Pose	Initial Robot Pose
State Estimator	Forward Kinematics	Odometry (inaccurate)	Odometry (inaccurate)

Blundell et al. [19] propose to model Neural Network (NN) weights with a zero-centered Gaussian distribution. Learning is then casted as a variational inference problem, where the true target distribution is approximated by the learned model conditioned on input data. By noting that commonly used weight distributions are symmetric and independent, Wen et al. [192] improve the ideas of Blundell et al. [19] by decomposing the forward pass into multiplication of input and weight means by independently sampled sign matrices [192, eq. (4)], resulting in faster computation and less variance in the gradients.

In contrast, our approach tackles the different problem of learning from noisy data, without delving into the estimation of the uncertainty associated to predictions.

5.1.2 Model

We aim to train a model that, given readings $\boldsymbol{x}(t)$ collected by onboard sensors, predicts a target variable $\boldsymbol{y}(t)$, consisting in a pose $\boldsymbol{p}$ in SE(3) of the frame $\mathcal{F}_i$ of an OoI, relative to the moving robot frame $\mathcal{F}_r$. The sensor readings $\boldsymbol{x}(t)$ do not need to have an explicit geometric interpretation, might be high-dimensional (e.g., an uncalibrated image), and could potentially represent the concatenated outputs from multiple heterogeneous sensors.

To train the model, we use data collected in one or more training episodes. During each episode, the OoI is static (i.e., $\boldsymbol{y}$ does not change when expressed in a fixed reference frame) while the robot moves in the environment. Let T be the set of all timesteps in a given training episode.

We assume that a black-box *detector* provides temporally sparse estimates of $\boldsymbol{y}(t)$:

$$\boldsymbol{d}(t) = \begin{cases} \tilde{\boldsymbol{y}}(t) & \text{if } t \in T_d, \\ \text{undefined} & \text{otherwise} \end{cases} \tag{5.1}$$

where $T_d \subseteq T$ denotes the set of timesteps in which the detector module provides an output, and the tilde over $\boldsymbol{y}$ denotes the fact that this is a potentially inaccurate estimate of the true value of the target variable.

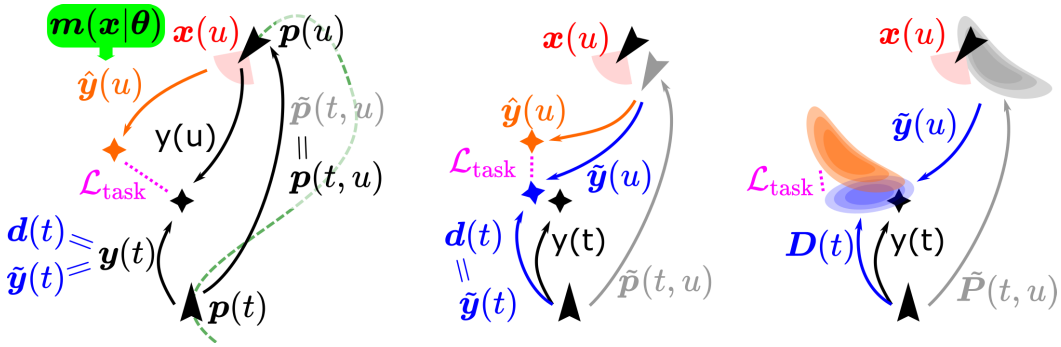


Figure 5.1. Illustration of the task loss in case the detector and odometry are ideal (left), inaccurate (center), or with a known uncertainty model (right). Black color denotes the true poses for the robot (arrowhead) and OoI (diamond); blue color denotes the OoI pose as returned by the detector; orange denotes model predictions, obtained from data (red) sensed at time u. Gray denotes robot odometry. See text for details.

Finally, we assume that an odometry module outputs for every $t \in T$ a (potentially inaccurate) estimate $\tilde{\boldsymbol{p}}(t)$ of the robot pose $\boldsymbol{p}(t)$ in a fixed inertial frame $\mathcal{F}_0$. Given two timesteps $t, u \in T$, we denote with $\tilde{\boldsymbol{p}}(t,u)$ the odometry's estimate of the pose of the robot at u with respect to its pose at t. In particular, $\tilde{\boldsymbol{p}}(t,u) = \ominus\tilde{\boldsymbol{p}}(t) \oplus \tilde{\boldsymbol{p}}(u)$, where $\oplus$ denotes the pose composition operator, and the unary operator $\ominus$ denotes pose inversion [40].

For each training episode, we collect the set of samples

$$\{\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{d}(t) | t \in T\}; \tag{5.2}$$

and use data from all training episodes to learn a mapping from $\boldsymbol{x}$ to $\boldsymbol{y}$. The mapping is implemented by a NN model $\boldsymbol{m}(\boldsymbol{x}|\boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta}$. Training is performed by minimizing a loss function

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{sc}}\mathcal{L}_{\text{sc}}, \tag{5.3}$$

composed by a task loss $\mathcal{L}_{\text{task}}$ and a state-consistency loss $\mathcal{L}_{\text{sc}}$; the latter is scaled with a factor λ_{sc}.

We first describe these two terms in case the detector and odometry return point-wise estimates; then, we extend the discussion to the case in which their uncertainty can be modeled with a probability distribution, see Figure 5.1 and Figure 5.2.

Task loss

Consider two timesteps t, u in the same episode, such that $t \notin T_d$ and $u \in T_d$. The task loss enforces that the model, when fed with $\boldsymbol{x}(t)$ returns a pose that is consistent with $\boldsymbol{d}(u)$, accounting for the pose transform measured between t and u by the robot odometry $\tilde{\boldsymbol{p}}(t,u)$.

More specifically, $\boldsymbol{d}(u)$ is an estimate of the target variable with respect to the robot frame at time u; it follows that $\tilde{\boldsymbol{y}}(t) = \tilde{\boldsymbol{p}}(t,u) \oplus \boldsymbol{d}(u)$ is an estimate of the target variable at t. Thus, the task loss is defined as

$$\mathcal{L}_{\text{task}} = \sum_{t \in T_d, u \in T} \Delta_{\text{SE3}}(\tilde{\boldsymbol{p}}(t,u) \oplus \boldsymbol{d}(u), \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})), \tag{5.4}$$

where the function $\Delta_{\text{SE3}}(\cdot,\cdot)$ is a measure of the distance between two poses in SE(3) and defined as

$$\Delta_{\text{SE3}}(\boldsymbol{p}_a, \boldsymbol{p}_b) := \lambda_{\text{o}} \|\boldsymbol{o}_a - \boldsymbol{o}_b\| + \frac{1}{\pi} \Delta_{\text{quat}}(\boldsymbol{q}_a, \boldsymbol{q}_b), \tag{5.5}$$

where $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ are two generic poses, composed of the position components $\boldsymbol{o}_a, \boldsymbol{o}_b \in \mathbb{R}^3$, and the rotation components represented as quaternions $\boldsymbol{q}_a, \boldsymbol{q}_b \in \mathbb{H}$, being $\mathbb{H}$ the non-commutative ring of the quaternions. In (5.5), $\Delta_{\text{quat}}(\cdot,\cdot)$ denotes the quaternionic distance [100, eq. (4)]. Note that the rotational term of the distance is bound in $[0,1]$ while the positional term has no upper bound. The parameter λ_{o} is introduced as scaling factor to weigh the two terms.[1]

This definition of the task loss uses odometry to propagate the estimate of $\boldsymbol{y}$ (produced by the detector in a timestep $u \in T_d$), to any other timestep t in the same episode. If both the detector and the odometry are ideal (error-free), using any $u \in T_d$ yields the same value of $\tilde{\boldsymbol{y}}(t) = \tilde{\boldsymbol{p}}(t,u) \oplus \boldsymbol{d}(u)$. Otherwise, if the detector and/or the odometry are not ideal, every different choice of u yields a different estimate of $\tilde{\boldsymbol{y}}(t)$. In practice, this is expected to mitigate inaccuracies as errors are averaged out during training.

State-consistency loss

Consider two timesteps t, u in the same sequence, and assume that $t, u \notin T_d$. The state-consistency loss enforces that the predictions of the model at t and u are consistent with each other, accounting for the robot's odometry between t and u. More specifically,

$$\mathcal{L}_{\text{sc}} = \sum_{t,u \in T} \Delta_{\text{SE3}}(\tilde{\boldsymbol{p}}(t,u) \oplus \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta}), \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})). \tag{5.6}$$

Consider the following example; given $\boldsymbol{x}(t)$, the model returns an estimated pose for an OoI 1.5m in front of the robot; after the robot advances 1m, at time u, the model given $\boldsymbol{x}(u)$ should return a pose that is 0.5m in front of the robot. The state-consistency loss ensures that predictions $\hat{\boldsymbol{y}}(t) = \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})$ and $\hat{\boldsymbol{y}}(u) = \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta})$ match this expectation.

5.1.3 Dealing With Uncertainty

Our approach relies on two sources of information, namely the detector $\tilde{\boldsymbol{d}}(t)$ and the odometry $\tilde{\boldsymbol{p}}(t,u)$, both of which are possibly affected by measurement errors (instantaneous for the detector, accumulated over time for the odometry). If such errors can be modeled, we can explicitly account for them in our approach.

In particular, we represent the uncertainty of $\boldsymbol{d}(t)$ by considering that the detector's output is, instead of a pointwise estimate of the target pose, a probability distribution over poses, defined in SE(3). We denote such probability distribution as $\boldsymbol{D}(t)$.

Similarly, for odometry we define $\boldsymbol{P}(t,u)$ as the probability distribution of the relative pose of $\boldsymbol{p}(t,u)$; this also accounts for the fact that odometry errors accumulate over time, and therefore are not independent for different times.

This representation allows to reformulate the task loss as

$$\mathcal{L}_{\text{task}} = \sum_{\substack{t \in T_d \\ u \in T}} \mathbb{E}\Big[\Delta_{\text{SE3}}\Big(\boldsymbol{p}(t,u) \oplus \boldsymbol{d}(t), \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta})\Big)\Big], \tag{5.7}$$

[1] In principle, other options for representing poses and their distance [206, 134] might be adopted, as long as the distance function is continuous and differentiable

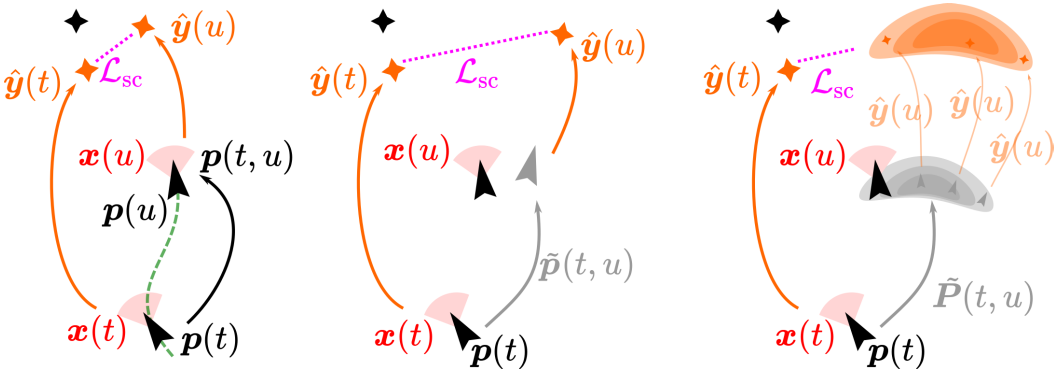


Figure 5.2. Illustration of the state-consistency loss in case the odometry is ideal (left), inaccurate (center), or with a known uncertainty model (right); black color denotes the true poses for the robot (arrowhead) and OoI (diamond). Orange denotes the outputs of the model at times t and u, which depends on sensed data (red). The state consistency loss (violet) forces the model to output consistent estimates, accounting also for odometry (gray) and its uncertainty, if known.

and similarly, the state-consistency loss can be rewritten as

$$\mathcal{L}_{\text{sc}} = \sum_{t,u \in T} \mathbb{E}\Big[\Delta_{\text{SE3}}\big(\boldsymbol{p}(t,u) \oplus \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta}), \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})\big)\Big] \tag{5.8}$$

where $\boldsymbol{p}(t,u) \sim \boldsymbol{P}(t,u)$ and $\boldsymbol{d}(t) \sim \boldsymbol{D}(t)$.

In practice, when implementing the losses we approximate the expectation as the average over a finite number of realizations, in which $\boldsymbol{d}(t)$ and $\boldsymbol{p}(t,u)$ are Monte Carlo samples of the respective distributions.[2]

5.2 Experiments

This section validates our approach with three applications that differ in complexity and input dimensionality: (i) pose estimation of an OoI with a robotic arm; (ii) robot heading estimation using infrared sensors; and (iii) indoor localization of a ground robot.

5.2.1 Object of Interest Pose Estimation With a Robotic Arm Setup

We consider the robotic manipulator Panda by Franka Emika, equipped with an Intel RealSense D435 sensor [82] at its end-effector, simulated with Gazebo [84], see Table 5.1. The task is to estimate the 3D pose of an OoI, i.e., a colored mug, which is equipped with a small visual fiducial marker that is visible only when the cup is observed from a specific viewpoint. The frames $\mathcal{F}_0$ and $\mathcal{F}_r$ are placed at the base of the robot and at its end-effector respectively; while the frame $\mathcal{F}_i$ of the OoI is in the center of the mug. The end-effector pose $\boldsymbol{p}(t)$ is given by the robot forward kinematics. The input $\boldsymbol{x}(t)$ is a 160 × 120 pixel RGB image acquired from the

[2] In a straightforward implementation, this multiplies the number of training samples by a factor equal to N_{mc}, and yields a correspondingly longer training time; in contrast, no additional computation is needed during inference.

RealSense camera. The estimates $\boldsymbol{d}(t)$ are generated by the AprilTag [185] off-the-shelf fiducial marker detector operating on $\boldsymbol{x}(t)$; only when the marker is clearly visible in the frame, the detector returns an estimate of the 3D pose of the marker w.r.t. $\mathcal{F}_r$[3].

As shown in Table 5.1, the environment consists of a flat table of 90×90 cm with different objects, some textured and some others having a solid color, besides the mug. In each training episode, the table and objects color, the position of the objects, and the direction of the light illuminating the scene are randomized to generate different environments. For each environment, the robot moves the end-effector in order to reach a total of 32 goal poses using the ROS MoveIt [36] implementation of the RRT planner. Each goal position is sampled from a semi-sphere having a radius of 55cm placed at an height of 35cm from the table; the goal orientation is set to make the camera look towards a random point lying 5cm above the table. For each environment, the end-effector pose is initialized at the center of the semi-sphere.

The collected data amounts to 237k tuples (of which only 78k have the mug visible), corresponding to 157 environments and 5 simulated hours. The data is finally split into a training set (119 environments), a validation set (18 environments) and a testing set (20 environments). Training is performed with a $\lambda_o = 10$, striking a balance between the positional and rotational errors.

5.2.2 Robot Heading Estimation Using Infrared Sensors Setup

For this application, we use a Thymio [111, 66], a small differential drive robot equipped with 9 infrared sensors: 5 mounted at the front, 2 at the rear of the robot body, and 2 at the bottom that are not used in this experiment. Each sensor measures the amount of infrared light reflected from the environment, which is related in some unknown way to the distance and orientation of the sensor with respect to an obstacle. The input of the model $\boldsymbol{x}(t)$ consists in the uncalibrated readings of the 7 sensors at time t, while $\boldsymbol{y}(t)$ is the angle of the wall w.r.t the robot. Note that we can still adopt Δ_{SE3}, setting $\lambda_o = 0$, thus considering only the heading. The robot odometry $\tilde{\boldsymbol{p}}(t)$, derived from the wheel encoders, provides the 2D transformation of the robot frame $\mathcal{F}_r$ w.r.t. the inertial frame $\mathcal{F}_0$. The scenario, along with an illustrative schematic of the sensor readings, is shown in Table 5.1.

Episodes are collected by teleoperating the robot along trajectories in the proximity of the wall. During each episode, the robot true pose is tracked by a fixed tracking infrastructure (12 Optitrack cameras), which is used as a comparison for experiments. At the beginning of each episode, the robot touches the wall with its rear side, thus the inertial frame $\mathcal{F}_0$ coincides with the robot frame $\mathcal{F}_r$ at this instant in time. This piece of information also acts as a virtual detector, whose output $\tilde{\boldsymbol{d}}(t)$ is available only in the first timestep of each episode.

Information about the rotation of each wheel is computed by the robot's firmware by measuring the current flowing through each motor – an inaccurate approach whose errors we model to compute the robot's uncertain odometry.

A total of 16 distinct episodes are recorded, each lasting on average 34 seconds, during which samples are collected at 10 Hz (a total of 5453 samples). Episodes are split into training and validation sets for experiments using a leave-one-episode-out cross-validation scheme.

[3] The rigid transformation between the end-effector and the camera frame is assumed to be known, through a calibration procedure. Similarly, the offset between the origin of $\mathcal{F}_i$ and the center of the marker is taken into account in our computations.

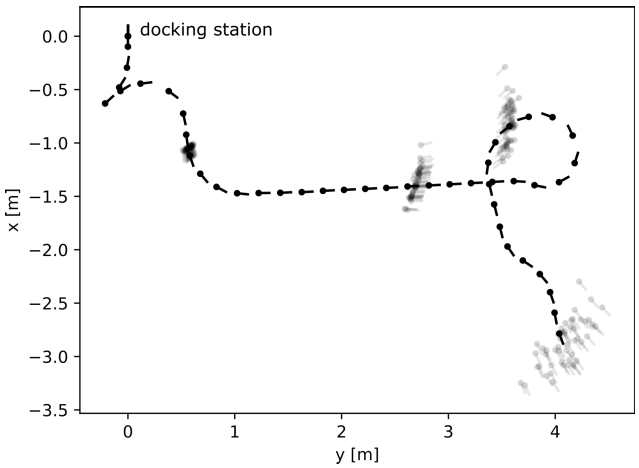


Figure 5.3. Visualization of measured inaccurate odometry (black), and 50 realizations of the uncertain odometry, for the first minute of a training episode of the RoboMaster robot.

5.2.3 Indoor Localization of a Ground Robot Setup

We consider a wheeled ground robot, the DJI RoboMaster EP, equipped with an onboard camera and omnidirectional motion capabilities using Mecanum or Swedish wheels, see Table 5.1. We consider a situation in which the robot navigates a given indoor environment and, when required, needs to come back to a fixed docking station, e.g., to recharge the batteries. In our setting, the docking station is represented by a mark on the floor. The input $\boldsymbol{x}(t)$ is the camera stream, downsampled to a resolution of 160×120 pixels; the perception task consists in predicting the pose of the docking station relative to the robot frame $\mathcal{F}_r$, given an image $\boldsymbol{x}(t)$ acquired at a generic pose. Note that, since the docking station is fixed in the environment, this problem is equivalent to robot localization; it differs from the object localization task presented in Section 5.2.1 because the docking station does not need to be visible in the input image for successful estimation.

The robot onboard odometry module provides $\tilde{\boldsymbol{p}}(t)$ w.r.t. the inertial frame $\mathcal{F}_0$, which coincides with $\mathcal{F}_r$ at the beginning of the acquisition. Figure 5.3 depicts the inaccurate robot odometry as measured, and 50 realizations accounting for uncertainty, for the first minute of a training episode.

Furthermore, at time $t = 0$ the robot undocks from the docking station; similarly to the previous case, we use this information as a detector. In this case study, the OoI is the docking station and its frame $\mathcal{F}_i$ coincides with frame $\mathcal{F}_0$, as in Section 5.2.2.

We collect 20 episodes recording data at 15 Hz, for a total of 70k samples in 80 minutes (4 minutes per episode). Each episode begins with the robot attached to the docking station; the robot is then teleoperated to explore the environment. Data from different episodes are split into a training set (40k samples, 10 episodes), a validation set (15k samples, 5 episodes) and a testing set (15k samples, 5 episodes). Training is performed with $\lambda_o = 10$, similarly to Section 5.2.1.

5.2.4 Network Architectures and Training

In all case studies, a NN is trained using Adam [83] as optimizer with a learning rate of $1e^{-3}$; early stopping is used to determine when to conclude the training process. An architecture based on MobileNet-V2 [154] with a total of 1 million parameters is used for the case studies in Section 5.2.1 and Section 5.2.3; it maps a 160×120 RGB image to an output vector representing a 3D pose (composed of 7 elements, 3 for the position and 4 for the quaternion). In these two use cases, we artificially increase the amount of data used for the training, by adopting the following data augmentation techniques on the input images: blurring, multiplicative Gaussian noise, random brightness and contrast, random resized cropping.

In the case study detailed in Sec. 5.2.2, we use a simpler NN architecture composed of four linear layers with a total of a 1000 parameters, mapping the 7 infrared sensors' readings to a 3D pose.

5.2.5 Object of Interest Pose Estimation With a Robotic Arm Results

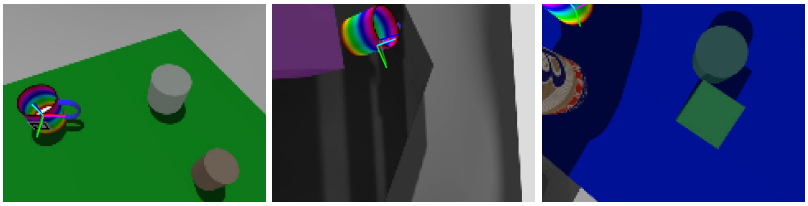
For the scenario described in Section 5.2.1, we first evaluate the trained model on the testing set, by comparing the predicted poses $\hat{\boldsymbol{y}}$ to the corresponding ground-truth $\boldsymbol{y}$. Predictions occur independently for every frame; no information from state estimation or the detector is used in the process.

For the position component, the model achieves a Root Mean Squared Error of 39.9mm, and a coefficient of determination R^2 of 0.962, 0.960 and 0.866 on the x, y and z components, respectively. The coefficient of determination is an adimensional measure of the quality of a regressor, which quantifies the amount of variance in the target variable that is explained by the model; an ideal regressor yields $R^2 = 1$, whereas a dummy regressor estimating the mean of the target variable yields $R^2 = 0$; we observe that, while all components are estimated well, the z coordinate, i.e., the distance of the OoI w.r.t. the camera, is estimated with lower accuracy; this is expected, since estimating distances is hard from low-resolution monocular images. The rotational component is estimated with an average rotational error $\Delta_{\text{quat}} = 0.1417$, corresponding to an angle of about 25°.

As a comparison, we also trained a supervised model using ground truth poses of the OoI instead of the detector outputs: the resulting prediction performance is the same as with the self-supervised approach: in fact, in the considered simulated environment the detector is very accurate and state estimation is ideal, which makes the self-supervised labels almost identical to ground truth labels. This is not the case for the experiment in Section 5.2.6.

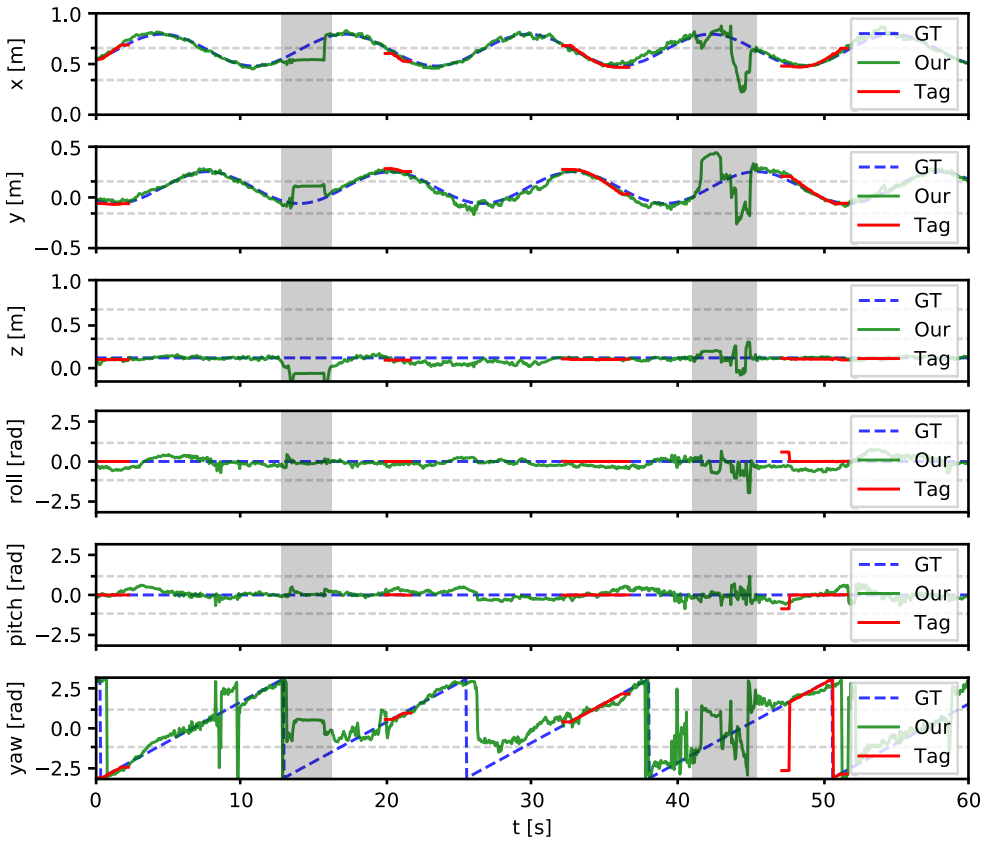
Figure 5.4(a) shows a sequence of camera frames from the testing set with an overlay of the model's prediction and the ground-truth. We observe that the model correctly identifies the object and accurately estimates its pose, even when the OoI is only partially visible or occluded, and independently on the visibility of the fiducial marker used during training. When the object is not visible at all on the image, the model tends to confuse it with other objects that are visually similar; this highlights a limitation of our approach: we do not explicitly handle aliasing, i.e., the case in which $\boldsymbol{x}$ does not contain enough information to estimate $\boldsymbol{y}$.

To demonstrate the fact that the model, once trained, can be used in dynamic environments, we consider an additional testing scenario: the mug is placed on a rotating disk support and observed while the robot also moves. Figure 5.4(b) shows a sequence of camera frames captured during the experiment. Figure 5.4(c) compares the model prediction to AprilTag detections, and to ground-truth. For the sake of clarity, the target has been transformed in the frame $\mathcal{F}_0$,



(a) Predictions (magenta-lime-cyan colored frame) and reconstructed ground-truth (red-green-blue frame) on testing set data.

(b) Input images from the additional testing scenario.



(c) Pose of the OOI in the inertial frame on the additional testing scenario.

Figure 5.4. Prediction of the OoI pose with a robotic arm on the testing set (a); input images (b), ground-truth and predictions (c) relative to the additional testing scenario. In (b-c) a mug is placed on top of a rotating disk support. To more easily evaluate the results, we report both target variables and predictions in the inertial frame $\mathcal{F}_0$. Grayed out areas indicate time intervals in which the mug is not visible.

along with the model predictions. The model (green in the plots) manages to predict a consistent pose (it well overlaps the blue dashed line of the ground-truth), outperforming the AprilTag detection (red line), which provides a measure for a small fraction of frames in which the tag is visible, while our model runs at 25Hz on a GPU Nvidia Quadro P2000. The model produces good estimates of the mug pose from most points of view, even when it appears upside down. Occasional failures occur when the mug is seen from a point of view that does not provide any reference to infer its actual rotation around the vertical axis, i.e., when both the marker and the handle of the mug are not visible. While it is robust to partial occlusions of the mug, the model fails to estimate the mug's pose when a very small portion of it is visible, or when it is totally invisible; time intervals in which the mug is not visible are depicted with a gray shadow in Figure 5.4(c).

5.2.6 Robot Heading Estimation Using Infrared Sensors Results

For the scenario described in Section 5.2.2 we consider three possible sources of odometry: *exact odometry*, where $\tilde{\boldsymbol{p}}(t,u) = \boldsymbol{p}(t,u)$, as measured by the optitrack system, acting as an upper bound of the achievable performance; *pointwise odometry*, where the relative pose $\tilde{\boldsymbol{p}}(t,u)$ is computed according to the known robot kinematics and the readings of the wheel rotation sensors between timesteps t and u; *uncertain odometry*, where the probability distribution $\tilde{\boldsymbol{P}}(t,u)$ is approximated by $N_{\text{mc}} = 50$ realizations of the odometry between timesteps t and u, obtained by corrupting the readings of the wheel rotation sensors with white Gaussian noise, whose variance matches the known uncertainty of the sensor.

Figure 5.5 reports the Mean Absolute Error between the predicted angle of the robot pose with respect to the wall, against the ground-truth angle as measured by the optitrack system; this metric is reported for four models trained with different odometry sources, with ($\lambda_{\text{sc}} = 1$) or without ($\lambda_{\text{sc}} = 0$) the state-consistency loss. Each of the four models is trained and evaluated 16 times according to the leave-one-episode-out cross validation scheme. We observe that: i) using uncertain odometry instead of pointwise odometry improves the prediction performance of the resulting model,[4] ii) additionally enforcing the state-consistency loss ($\lambda_{\text{sc}} = 1$) further improves performance, even though the incremental improvement over the uncertain model with $\lambda_{\text{sc}} = 0$ is not statistically significant; iii) compared to the model trained with exact odometry, serving as a supervised learning upperbound, the performance gap of our best model is less than half of the performance gap of the baseline model (pointwise $\lambda_{\text{sc}} = 0$), specifically of 0.57° vs 1.20°.

5.2.7 Indoor Localization of a Ground Robot Results

For the scenario described in Section 5.2.3, we report one qualitative and one quantitative experiment. In the qualitative experiment, we train a model using uncertain odometry ($N_{\text{mc}} = 50$) and $\lambda_{\text{sc}} = 1$, then apply the trained model independently on each frame of a testing episode. Figure 5.6 shows the position component of the predicted poses (green), compared to the poses returned by the robot's odometry (blue) for the same trajectory; despite the short duration of the testing episode (about 4 minutes), the drift of the odometry trajectory is apparent: the blue trajectory passes through a wall on the left; in comparison, the poses predicted by our approach are locally noisy but not affected by accumulating error. The figure also depicts the predicted pose at the end of the trajectory for each of the two methods; our model correctly predicts that

[4] we use the non-parametric Wilcoxon signed-rank test between matched samples, i.e., the performance of two models on the same cross-validation fold ($p = 0.0003$)

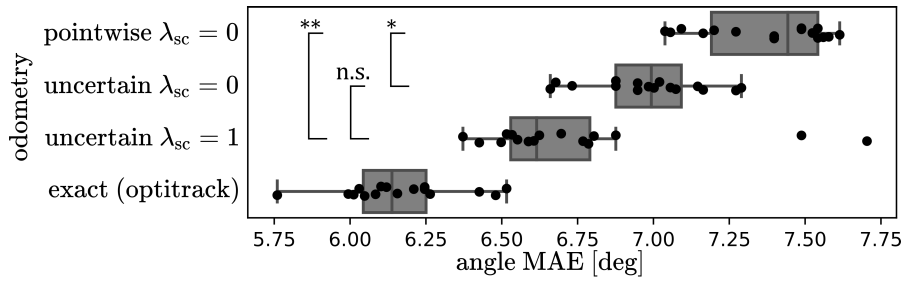


Figure 5.5. Mean absolute angle error (lower is better) in the heading estimation task. * $p = 0.0003$; ** $p = 0.0008$; n.s. means not significant ($p = 0.029$).

Table 5.2. Quantitative performance on the indoor localization task

Method	Position Error [mm]	Rotation Error [deg]
Odometry	84.9	11.9
Pointwise, $\lambda_{sc} = 0$	62.8	23.8
Pointwise, $\lambda_{sc} = 1$	49.1	7.5
Uncertain, $\lambda_{sc} = 0$	73.5	9.1
Uncertain, $\lambda_{sc} = 1$	**35.2**	**3.8**

the robot is back at the docking station, whereas the odometry has drifted by about 50cm and 20°.

In the quantitative experiment, we consider four timesteps $t \in t_1, t_2, t_3, t_4$ of the testing episode, whose ground-truth poses $\boldsymbol{p}(t)$ have been manually measured with respect to the docking station. For each of the four timesteps, we measure the positional and rotation components of the error against such ground-truth, for: i) the pose $\tilde{\boldsymbol{p}}(t)$ estimated by the robot odometry; ii) the poses estimated by each of four models. In particular, we consider models trained with pointwise or uncertain odometry, with or without using the state-consistency loss. Table 5.2 shows that: using the state-consistency loss improves both positional and rotational errors; using uncertain odometry during training consistently outperforms pointwise odometry.

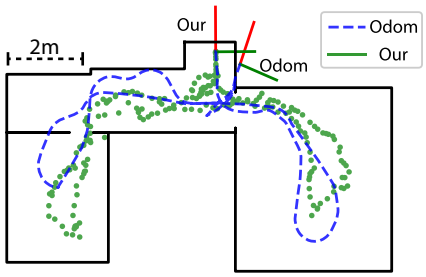


Figure 5.6. Robot localization task on testing data.

5.3 Discussion

We presented a general self-supervised learning approach for spatial perception tasks, and instantiated it on three different case studies. The approach is general enough to be applicable to different robots and sensor apparatus, requiring only a possibly uncertain odometry and a detector that sparsely produces a ground-truth estimate. A novel loss allows us to evaluate the model

outputs also for timesteps in which no supervision is available, by propagating such supervision from different timesteps using uncertain state estimates. Furthermore, the loss formulation enforces consistency among predictions at different timesteps, which further improves performance. Results show consistent and statistically significant improvements of models learned with the uncertainty-aware version of the loss compared to the respective baseline.

So far, we examined self-supervised robot learning approaches that extract supervision from sensors and odometry. Among various options being explored for proposing new supervision sources, task demonstrations are a solid candidate. In the next chapter, we study the use of task demonstrations to supervise a perception model to predict visual features of an OoI. By learning to imitate the demonstrated task, the visual features produced by the model become consistent w.r.t. the OoI, resulting in the desired behaviour.

Chapter 6

Supervision From Task Demonstrations

Classic self-supervised robot learning approaches rely on sensors and other information produced by the robot to provide supervision to perception models. Given this data, robots generate labels to train a perception model, understanding the environment without relying on externally labeled data. In this chapter, we investigate a different way of deriving supervision for a perception approach using task demonstrations. Specifically, we consider the task of predicting visual features of an OoI such that they can be used inside the control law for visual servoing. Overall, this approach explores the potential of learning a perception task by supervising the model with robot trajectories. By leveraging a differential control law, the model learns to produce visual features for the OoI without needing ground truth features. The goal is to train a model to produce visual features that imitate the demonstrated behavior when combined with the control law.

6.1 Background

Recent advancements in the fields of robotics and AI are leading machines to higher levels of reliability and autonomy. Specific application domains, like domotics and Industry 4.0, are showing the big potential of these technologies. Indeed, the ubiquity of robots chased for decades is quickly becoming reality, although it brings novel needs and challenges. In particular, an increasing number of nonspecialist users ask for easy-to-use robots and no programming duties. Even in technical domains, like industry, practitioners with little or no expertise in robotics wish for plug-and-play platforms.

Today's progress of Deep Learning (DL) permits a relatively straightforward application of NNs, which are known to handle well high dimensional data and complex perception tasks [64]. In this work, we focus on how to combine DL and control techniques to solve the perception problem in the specific context of Visual Servoing (VS). The traditional VS paradigm relies on the image processing to extract a suitable control feedback. It has to be properly designed, implemented, and normally tailored to the desired task. To increase the easiness of use, also for nonspecialists, a possible way is the complete removal of the explicit image processing block from the VS scheme. Furthermore, it is desirable to have a modular perception block, possibly

transferable to other visual controllers. To this end, we propose to train a NN that derives supervision from the knowledge of the control structure and the robot motion to provide neural feedback from monocular images. The proposed framework keeps the classic perception-and-control scheme where the feedback extraction algorithm is left outside the control block. Nevertheless, our perception model is made aware of the VS structure and the visual features motion model, leveraging this information to give a geometric interpretation to the neural feedback. This approach differs from an end-to-end one, where sensing and action are, instead, coupled. We claim that keeping the controller analytical structure enables higher flexibility of the entire framework, besides customization of the downstream control.

The rationale of relying on DL, or other sorts of Machine Learning (ML), to avoid the explicit image processing in the VS has already been proposed in different forms (see Section 6.1.1). However, we speculate that neural perception, control, and modeling aspects can bring a beneficial synergy to provide the VS scheme with effective neural feedback (Section 6.1.2). Our approach is to train a neural perception model oriented to the VS task so that the prediction provides a robust and tailored control feedback. Using the setup detailed in Section 6.2, we validate our approach with simulations and real experiments.

6.1.1 Related Work

Classic VS uses visual features extracted from camera images to control robots [28, 29]. In Image-Based VS (IBVS), they are directly measured on the image, whereas Position-Based VS (PBVS) uses visually reconstructed camera poses. The basic formulation has been expanded to planning [106, 31], optimization-based control [3, 126, 108], model predictive control [5, 156, 124] and integrated with ML-based concepts [124, 125]. In any case, the images have to be explicitly processed to properly detect, track and match the visual features.

More advanced approaches avoid the explicit image processing considering the whole image as feedback. This newer class of VS, known as direct VS, proposes to use, e.g., photometric moments [10], luminance of the pixels [37], histograms [13] or Gaussian mixtures [41] as visual features.

Recently, ML techniques have been considered to solve the image processing problem in VS. In [26] a Gaussian Mixture Model is used to imitate the classic VS behavior using a reduced image resolution. In [79], task-relevant features are extracted by a NN to build a more complex feedback and the visual error. The idea of considering the whole image to avoid specific image processing is well suited to the application of DL, which efficiently handle dense visual information [64]. Indeed, nowadays many works propose to learn the relative camera-scene pose from monocular images [93, 45, 115], which in principle can be used to realize PBVS; more relative pose estimation approaches are described in Section 2.5.1. Specifically to the VS case, several works rely on DL to infer from images the relative camera displacement [157, 198, 14]. In [54] visual features are extracted from an autoencoder latent space and their motion model computed, to be considered in the analytical VS law. Deep visual features and their dynamics are learned by an agent in [89].

In end-to-end approaches, a DL architecture computes control commands directly from images, avoiding the explicit visual feature extraction by coupling the perception and control problem [91]. End-to-end architectures use a NN to first learn a representation of the image and another NN to regress velocity commands [139, 53]. In [102] an end-to-end is compared to a cascade of two NNs computing the target pose and the command for an aerial drone.

Our approach uses DL to extract from monocular images geometrically-interpretable visual

features needed to close the VS loop. In principle, it can learn any feedback described by the VS law considered during the training phase, being it a Cartesian pose or an image feature. It differs from an end-to-end approach, since we maintain the classic perception-control paradigm, keeping the controller structure untouched downstream the perception block. Our neural perception derives supervision from the control knowledge so that it can provide a feedback tailored to the VS. Furthermore, keeping the control operation outside the perception block allows major flexibility of the whole framework. Being formally decoupled from the control, the trained perception model can be deployed within different control alternatives.

6.1.2 Visual Servoing

In VS, visual features $\boldsymbol{s} \in \mathbb{R}^f$ serve as feedback, extracted or reconstructed from images, required to close the control loop realizing the desired robot behavior [103, 28, 40]. In general, $\boldsymbol{s}$ depends on the camera calibration and/or Cartesian information [28, 40]; examples are points or lines on the image (in IBVS) or reconstructed camera poses (in PBVS).

Considering a robot arm with n DoF in the eye-in-hand configuration, where the camera is mounted on the robot arm's end effector, the time derivative of the visual features $\dot{\boldsymbol{s}}$ is directly related to the robot joints velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ through the relationship $\dot{\boldsymbol{s}} = \boldsymbol{J}\dot{\boldsymbol{q}}$, where $\boldsymbol{J} \in \mathbb{R}^{f \times n}$ is the Jacobian matrix. Assuming a non-moving target and a constant reference, the classic VS law used to control the robot is $\dot{\boldsymbol{q}} = -\lambda \hat{\boldsymbol{J}}^+(\boldsymbol{s} - \boldsymbol{s}^*)$ [29, 103] where λ is a positive control gain tuning the convergence rate; $\boldsymbol{s}^* \in \mathbb{R}^f$ are the desired visual features; $\hat{\boldsymbol{J}}^+ \in \mathbb{R}^{n \times f}$ is the pseudo-inverse of an approximation of the Jacobian. At convergence, under specific assumptions [28], the VS law makes sure that the measured features match with their desired counterparts. As a consequence, the camera is driven to the desired pose.

The following considerations about the Jacobian matrix shall be mentioned. It is composed as $\boldsymbol{J} = \boldsymbol{L}(\boldsymbol{s})\boldsymbol{V}\boldsymbol{J}_r(\boldsymbol{q})$ [29, 104], where: $\boldsymbol{L} \in \mathbb{R}^{f \times 6}$ is the interaction matrix that depends on the features and relates their motion to the camera velocity; $\boldsymbol{V} \in \mathbb{R}^{6 \times 6}$ is the constant matrix transforming the velocities from the end-effector to the camera frame; $\boldsymbol{J}_r \in \mathbb{R}^{6 \times n}$ is the robot Jacobian in the end-effector frame and depends on the robot configuration $\boldsymbol{q}$.[1] The Jacobian approximation is mainly due to the interaction matrix that depends on generally unknown spatial information, e.g., the visual features depth of the visual features; an estimation or approximation is normally used [28, 5, 104]. Furthermore, the Jacobian can become singular for particular robot or features configurations, with disruptive consequences on the correct control convergence. A damping term can be added to the control law, paid at the cost of lower tracking performances [161].

Neural feedback for visual servoing

We aim at extracting the VS feedback using a NN, tackling a pure perception task. We use the term *model* to denote the function $\boldsymbol{m}$ implemented by the NN and predicting the visual features from an image, i.e., $\boldsymbol{s} \approx \boldsymbol{m}(\boldsymbol{i}|\boldsymbol{\theta})$, where $\boldsymbol{i} \in \mathbb{R}^{whc}$ is an image with a resolution of $w \times h$ pixels and c channels; $\boldsymbol{\theta} \in \mathbb{R}^m$ is the set of weights[2] obtained by optimizing a loss function $\mathcal{L}$ in the training phase.

[1] If not explicitly mentioned, the dependency of the matrices on the robot configuration and the visual features is omitted for brevity.

[2] In what follows, we omit the dependency of the models on the weights.

The majority of NN-based methods rely on labels, i.e., the known value of the target variable contained in the data, to train models in a supervised fashion. Nevertheless, especially in robotics, the labeling procedure is costly and demanding. Alternative approaches use automated procedures to label the data, so that a bigger amount of information crucial for the performance of DL techniques, are made available. In this work, we propose to learn proper visual features for the VS, avoiding the processing required to explicitly label the images with visual features. Instead, we extract supervision from the knowledge of the VS and the motion model.

The ultimate aim of the work is to derive a VS law with the visual feedback provided by a NN:

$$\dot{\boldsymbol{q}} = -\lambda \hat{\boldsymbol{J}}^{+}\Big(\boldsymbol{m}(\boldsymbol{i}) - \boldsymbol{m}(\boldsymbol{i}^{*})\Big) \tag{6.1}$$

where $\boldsymbol{i}^{*}$ indicates the desired image that the camera would see at the completion of the task.

Our goal is to learn a neural perception model $\boldsymbol{m}$, such that the estimated visual features can be geometrically interpreted and thus used by a classic VS law. The task is defined as follows: find proper visual features for the VS law without any explicit image processing, directly using the raw uncalibrated image captured by the robot on-board camera. To this end, our model derives supervision directly from demonstrations of the VS, avoiding the need for explicit labels for the visual features. Training data, collected from the robot itself while it realizes several executions of the visual task, is a sequence of the following tuple:

$$\langle \dot{\boldsymbol{q}}_k, \boldsymbol{J}_{r,k}, \boldsymbol{i}_k \rangle, \; k = 1, \ldots, N \tag{6.2}$$

where N is the number of samples. Note that in the data there is no knowledge about the visual features. All the required information is given by the robot standard sensory equipment, composed of joint encoders and a monocular camera, with no further processing and readily accessible.

6.1.3 Model

Proper visual features can be reconstructed by imitating the desired VS control behavior. Thus, to train our model, we shall consider the following *control imitation* loss function:

$$\mathcal{L}_{\text{CI}} = \frac{1}{N} \sum_{k=1}^{N} \left\| \dot{\boldsymbol{q}}_k + \lambda \hat{\boldsymbol{J}}_k^{+}\Big(\boldsymbol{m}(\boldsymbol{i}_k) - \boldsymbol{m}(\boldsymbol{i}_k^{*})\Big) \right\|_1 \tag{6.3}$$

that forces the model to learn visual features such that their use in the VS law (6.1) imitates the demonstrated commands. We are not interested in learning the controller, as in end-to-end approaches. The knowledge of the control structure is instead leveraged to build a perception model tailored to the VS. As a result, the visual features are learned without the need for their explicit knowledge in the training data. In practice, the supervision of the learning process is provided by the imitation of the control behavior. Note that in (6.3) the reference image $\boldsymbol{i}_k^{*}$ is taken as the image captured at completion of each demonstration.

Recalling Section 6.1, the Jacobian in (6.3) has this shape:

$$\hat{\boldsymbol{J}}_k = \hat{\boldsymbol{L}}_k\Big(\boldsymbol{m}(\boldsymbol{i}_k)\Big)\, \boldsymbol{V}\, \boldsymbol{J}_{r,k}\big(\boldsymbol{q}_k\big) \tag{6.4}$$

where $\boldsymbol{J}_{r,k}$ depends on the robot configuration and its numerical value is taken from the data (6.2), while $\boldsymbol{V}$ is constant and known in advance. Instead, the interaction matrix $\boldsymbol{L}_k$ depends on the

visual features that are predicted by the model. It is worth recalling that the interaction matrix also depends on the features' depth; their known value at the target is used as an approximation. Furthermore, the analytical structure of the interaction matrix depends on the number and the kind of visual features, which are decided in advance in the training procedure (see Section 6.2). This design choice allows us to format the structure of the interaction matrix, which helps the learning process to find a particular geometrical interpretation of the prediction. Indeed, by imitating the control law, which has a precise structure grounded on the geometry of the visual features, it is possible to find in the data the correspondence between raw images and the geometrical interpretation requested by the VS feedback.

During the training of the model, especially at the first epochs, the model might provide naive features configurations. This issue possibly leads to a singularity of the Jacobian and a resultant ill-posed inversion problem. Thus, in (6.3), in place of $\hat{\boldsymbol{J}}_k^+$, it is considered:

$$\left(\hat{\boldsymbol{J}}_k^\top \hat{\boldsymbol{J}}_k + \sigma^2 \boldsymbol{I}_n\right)^{-1} \hat{\boldsymbol{J}}_k^\top \tag{6.5}$$

where σ is a damping term used to better handle the inversion of the Jacobian [161] and $\boldsymbol{I}_n$ is the $n \times n$ identity matrix.

State consistency

The temporal sequence of images and joint velocities is a rich source of supervision that can be exploited during the learning process. In fact, assuming that the object of interest is static during the data collection, the evolution of the visual features is well described by its motion model: given the visual features' estimate $\boldsymbol{m}(\boldsymbol{i}_k)$, the joint velocities $\dot{\boldsymbol{q}}_k$ and the Jacobian $\boldsymbol{J}_k$ at timestep k, we expect the neural output to be geometrically consistent with the motion model, i.e., producing at timestep $k+1$ the estimate $\boldsymbol{m}(\boldsymbol{i}_k) + \delta_t \boldsymbol{J}_k \dot{\boldsymbol{q}}_k$. We take advantage of this information and instantiate the state-consistency loss, as described in Chapter 4, which penalizes models that have an eradic prediction, failing to be consistent with the scene:

$$\mathcal{L}_{\text{SC}} = \frac{1}{N} \sum_{k=1}^{N-1} \| \boldsymbol{m}(\boldsymbol{i}_{k+1}) - \boldsymbol{m}(\boldsymbol{i}_k) - \delta_t \boldsymbol{J}_k \dot{\boldsymbol{q}}_k \|_1 \tag{6.6}$$

being δ_t the time difference between two consecutive timesteps. The effect of this loss is to further enforce the desired geometrical interpretation of the neural perception.

Image reconstruction as a pretext task

A particular class of NNs, called auto-encoder, uses an encoder to reduce high-dimensional images into a small-size latent variable, and a decoder to reconstruct the input image from the latent variable. The latent space is thus meant to be a highly informative domain where the most important notions of the image are condensed. Therefore, it is reasonable to extract the feedback of our VS law (6.1) from the latent variable of an auto-encoder. For this reason, we consider the *auto-encoding* loss function:

$$\mathcal{L}_{\text{AE}} = \frac{1}{N} \sum_{k=1}^{N} \left\| \boldsymbol{i}_k - \boldsymbol{\delta}\Big(\boldsymbol{\varepsilon}(\boldsymbol{i}_k)\Big) \right\|_1 \tag{6.7}$$

where the function $\boldsymbol{\delta}$ and $\boldsymbol{\varepsilon}$ indicate the decoding and encoding parts, respectively. Even though the latent variable is an effective representation of the image and conceptually close to the

visual feature, its semantic interpretation is entrusted to the NN. To be correctly interpreted and actually used in the controller (6.1), the visual features need to be extracted from the latent variable with an additional piece of network that we call *head* and denote with $\boldsymbol{h}(\cdot)$. Therefore, the neural visual features have this form:

$$\boldsymbol{m}(\boldsymbol{i}_k) = \boldsymbol{h}\Big(\boldsymbol{\varepsilon}(\boldsymbol{i}_k)\Big). \tag{6.8}$$

The head, trained by considering the loss functions defined in Section 6.1, gives the correct interpretation to the output of the encoder. Furthermore, it allows setting a desired dimension f of the neural visual features.

Note that the auto-encoding can be seen as a *pretext task* [80] , i.e., an auxiliary task that is of no direct interest, but whose solution stimulates the better performance of the *end task*, as discussed in Chapter 2. Practically speaking, the pretext task helps to find patterns in the data that are useful for the solution of the end task. Most importantly, as for (6.3), this strategy is particularly convenient because we can exploit unlabeled data and thus count on a bigger amount of information to solve our problem.

Regularization term The loss functions described in the previous sections are enough to solve our perception task. However, it has to be considered that the neural model is not enforced to produce an output easy to interpret for a human. In fact, the NN can find infinite solutions to our problem, all equally valid. For instance, it might produce point features outside the image plane borders and randomly displaced around the image of the object to track. To overcome this issue, we add an heuristic-based regularization term $\mathcal{L}_{\text{R}}$ that solves the ambiguity of the NN solutions by constraining particular geometrical properties. For example, it can be used to force the desired prediction $\boldsymbol{m}(\boldsymbol{i}^*)$ to be within the bounds of the image.

6.2 Experiments

To test our framework, we considered the 7 DoF robotic manipulator by Franka Emika [52]. The robot is equipped at the end-effector with an Intel RealSense depth camera D435, used as a monocular camera, streaming images at the nominal frame-rate of 30 Hz with a resolution of 640×480 pixels. The robot also provides a measurement of the Jacobian and the received joint velocity commands at a frequency of 200 Hz. Thus, we had easy access to the required information to construct the dataset (6.2). The framework and the communication with the robot are implemented in Python within the Robot Operating System (ROS) [58] infrastructure; the NN implementation is built upon the PyTorch library [128].

6.2.1 Data Collection

Data is collected within the robotic simulator Gazebo [84], where the exact knowledge of the robot and its working environment is available. In Gazebo, the robot carries out multiple executions of a VS task, consisting in framing an OoI at the center of the image captured by the onboard camera. In principle, to collect proper joint velocity commands, the VS task could be demonstrated in different forms, such as teleoperation, with a classic VS law, or using other control structures. In our setup, the VS task is demonstrated by an *ideal* classic IBVS using 4 point visual features. Thus, we consider the interaction matrix of point features in our learning procedure, and in the computation of the loss functions (6.3) and (6.6). As a consequence, the

model will be forced to give the geometrical interpretation of points to its estimate. In principle, different numbers and kinds of features may be considered, by selecting the corresponding interaction matrix in the training phase. The visual features are geometrically reconstructed from the 4 vertexes of the bounding box containing the OoI and projecting them on the image using the camera projection model. Thanks to the perfect knowledge of the camera model and the pose of the OoI w.r.t. the camera frame, it is possible to realize the VS task with high accuracy. Indeed, this demonstrator serves as a sort of *oracle*, providing the learning procedure with ideal data. However, the only collection of ideal situations might be not enough to guarantee the variability of data needed to reach generality performances during the deployment of the model. In practice, with an ideal IBVS, the smooth convergence produces very similar images, most of which have the OoI at the center of the image. To increase data variability, we adopt a two-steps procedure: first, the robot camera is driven to a random pose, and then converges driven by the ideal IBVS introduced above. In both the steps, we collect the commands as computed by the IBVS. Furthermore, we adopt a domain randomization [171] strategy: color and texture of the background, lighting conditions, the planar pose of the OoI, and the initial configuration of the robot are all randomized. Overall, we collected in the simulator the equivalent of 2 hours of data, totaling to approximately 100k examples, generated from 500 task demonstrations, out of which 83.5k examples were from the training set, and 16.5k from the validation set. Once collected, data is shuffled by pairs of two (to allow the implementation of the state-consistency loss, which needs at least two consecutive time samples) and organized in batches.

6.2.2 Network Architecture and Training

We consider a convolutional NN architecture composed of three main blocks: encoder, decoder, and head. The encoder consists of a series of 4 convolutions with stride 2 followed by ReLU non-linearity, halving each time the size of the image, followed by the bottleneck of size 64. The decoder consists of the bottleneck, followed by the same number of convolutions present in the encoder part, but with stride 1 and an up-sampling of the previous activation map, doubling each time the image size. The head takes the latent representation from the bottleneck and through a series of 3 feed-forward layers with ReLU non-linearities, and the final layer with TanH non-linearity produces the visual features. The model output is finally multiplied by a scaling factor, allowing the estimation to vary in a wider range and handle situations where the OoI is partially outside the camera field of view.

Our model is trained for 100 epochs with gradient descent, using the combination of loss functions described in Section 6.1.2:

$$\mathcal{L} = \lambda_{\text{CI}}\mathcal{L}_{\text{CI}} + \lambda_{\text{AE}}\mathcal{L}_{\text{AE}} + \lambda_{\text{SC}}\mathcal{L}_{\text{SC}} + \lambda_{\text{R}}\mathcal{L}_{\text{R}} \tag{6.9}$$

where the scalars λ_{AE}, λ_{CA}, λ_{SC} and λ_{R} are introduced to weigh the loss terms, allowing the tuning of the different components. In our setup, we heuristically set these parameters to one. The optimizer of choice is Adam [83] with a fixed learning rate of $1e^{-4}$. To synthetically increase the amount of training data, we apply data augmentation in the form of additive gaussian noise, random brightness, and contrast.

6.2.3 Simulated Experiment Results

Our perception model is evaluated in closed-loop simulations, in which the control law (6.1) computes the commands to fulfill the desired behavior. The evaluation considers 50 VS task

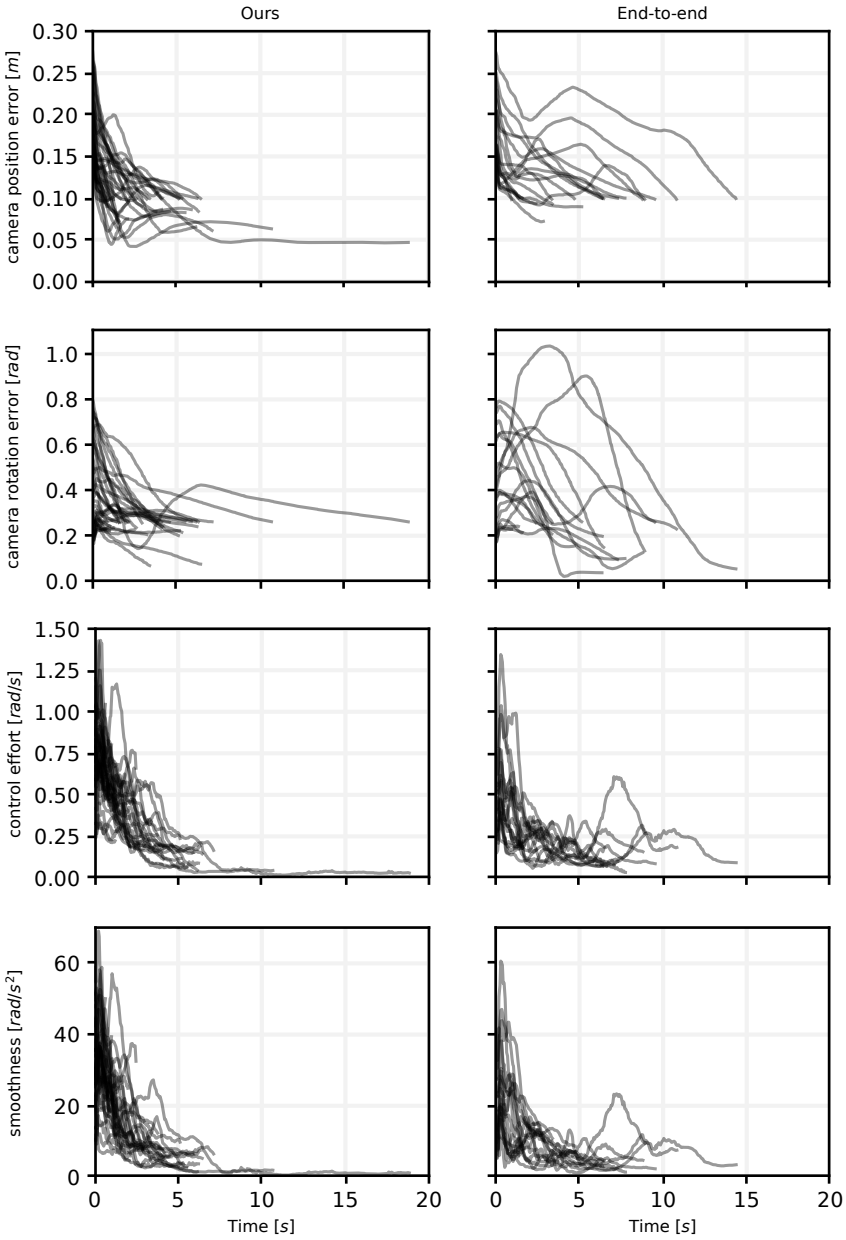


Figure 6.1. Control metrics evolution of the successful executions carried out by the analysis comparing our model with an end-to-end approach.

Table 6.1. Median of the control metrics over the successful executions of the task in simulation

Model	SR [%]	CE [rad/s]	CS [rad/s^2]	PE [cm]	OE [rad]
Ours	73	0.37	15.2	10.8	0.39
End-to-end	38	0.25	10.0	13.9	0.32

executions performed under different initial conditions. Each VS task consists in driving the robot camera at a desired pose w.r.t. a box. For each execution, the simulation environment is set up with the OoI placed at a random pose in front of the robot, a working surface with randomized texture, and an environmental light with random orientation. Then, the robot camera is moved to the desired pose using the knowledge of the simulator; at this time, the reference image $\boldsymbol{i}^*$ is saved, together with the object depth to be used in the interaction matrix computation. Thus, the robot is driven to a pre-defined initial posture and the VS experiment starts: at each time step, the current camera image $\boldsymbol{i}$ is acquired to infer the features using the model $\boldsymbol{m}$; the commands are then computed using the controller (6.1) and sent to the robot.

To evaluate the effectiveness of our perception model in providing a reliable feedback to the VS law, the following "control metrics" are considered: (i) control effort (CE), i.e., the norm of the commands; (ii) control smoothness (CS), i.e., the norm of the commands time derivative; (iii) position error (PE), i.e., the Euclidean distance of the camera position from its desired value; (iv) orientation error (OE), i.e., the quaternion distance [100, Eq. (4)] of the camera orientation from its desired value. These metrics are computed only for the successful, converging, executions. Each execution is considered successful if, in the given time of 20s, PE and OE fall below given threshold set to 10cm and 15° (0.26rad), respectively. CE, and CS are used to evaluate the quality of the control action produced by the neural feedback, whereas PE, OE serves as an index to establish the distance of the executions from the convergence.

The performance of our approach is compared against an end-to-end approach, that is trained to directly infer the joint velocities commands using as input camera feed and joint positions. We consider also the joint position as input of the end-to-end to realize a fair comparison against our approach, where the joint position is indirectly considered through the Jacobian. The end-to-end architecture shares the same layer structure as our model, with the exception of the head part, which receives the latent representation of the image concatenated with the 7 joint positions value.

Figure 6.1 compares the values of the control metrics obtained by our model and the end-to-end approach on the successful executions of the VS task. Table 6.1 show the median of the metrics over the successful execution, along with the success rate (SR). Overall we can observe a faster convergence and higher success rate of our model; we cannot see a remarkable difference in the control effort and smoothness.

It shall be mentioned that both the NNs presented problems in correctly estimating the box orientation, accumulating a drift in the visual features measurement, in the case of our model, and in the commands in the case of the end-to-end. This leads to undesired effects, even after convergence, for which the controlled motion drifted. We overcome this issue by stopping the execution right after the satisfaction of the convergence criterion previously described.

We also carried out an ablation study of our model, and we could not observe a significant difference between the versions of the model. This analysis suggests that there is room for improvements. In particular, we believe that the autoencoder could be more beneficial for the

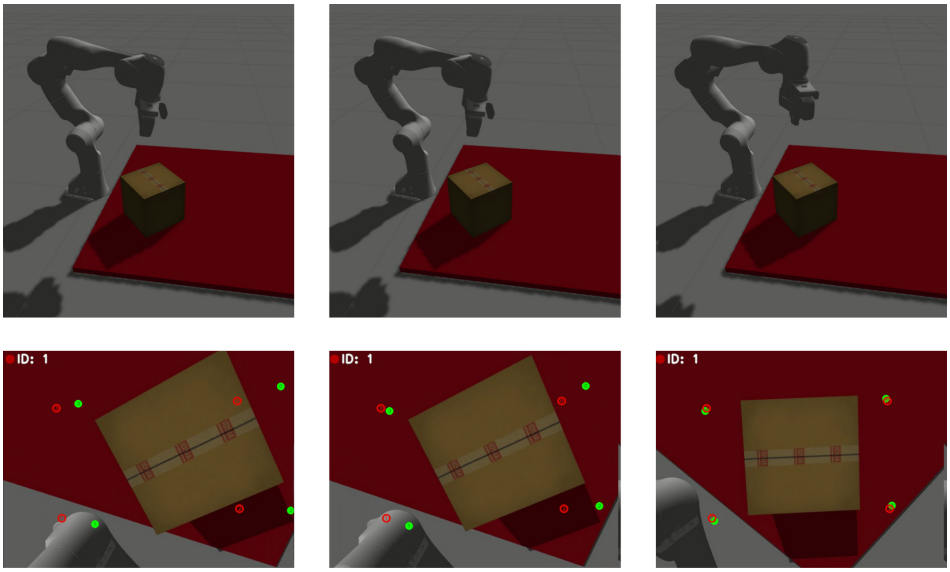


Figure 6.2. Execution of the VS task with the simulated robot manipulator.

overall approach when paired with a more advanced domain adaptation strategy [75], e.g., to cover the reality gap; instead, the state consistency loss could have a greater impact if applied to sequences longer than two samples.

The execution of one single VS task carried out using our model is presented in Figure 6.2. At the beginning of the execution, the robot camera is driven to the desired pose, the reference image taken and the corresponding desired visual features computed, by applying our model. The desired visual features are depicted with red circles in the camera views of Figure 6.2. During the execution, the model provides the VS with current visual features, depicted with green dots, till converge, when the camera pose value goes under a certain threshold. Figure 6.3 shows the computed commands and the norm of the visual error for this simulation execution.

6.2.4 Real Experiment Results

We consider our model also for closed-loop experiments carried out with the real robot manipulator. To this end, we considered the model trained only on randomized simulated data, and trained with additional data augmentations designed to replicate noise and interference present in the real camera sensor. Given the low performance of the model in estimating an accurate orientation of the OoI, we decided to change the structure of the control considering only the position. In practice, we projected the VS task into the null space of another primary task, consisting in keeping a constant camera orientation. Crucially, this design change in the control law was possible thanks to the flexibility of our approach, where the perception model is decoupled from the controller. Indeed, even if our model uses the knowledge of the classical VS scheme during training, its output can be used for deployment in other contexts; whereas the end-to-end approach does not offer such flexibility.

In Figure 6.4 we show three snapshots of the robot performing the control task, along with the corresponding camera views. The experiment starts with the OoI partially out of the camera's field of view – a challenging situation that could result in a failed attempt when using

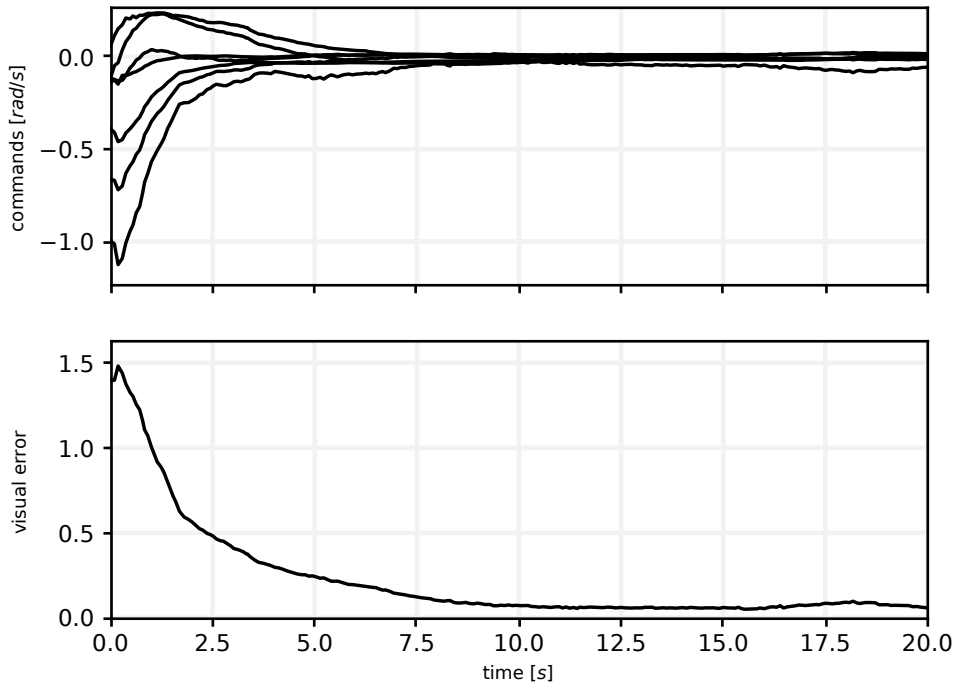


Figure 6.3. Control commands (top) and visual error (bottom) of the VS execution carried out with the simulated robot manipulator.

explicit methods for the image processing. Our neural perception model can provide the required feedback for the correct convergence of the VS task. A video of this experiment, as well as other executions and the simulations, are shown in the multimedia attachment.

6.3 Discussion

We have presented an approach for learning a deep perception model, providing feedback to visual controllers. The neural perception model is trained leveraging the knowledge of the visual servoing and the features' model as supervision. As a result, we estimate geometrically interpretable neural visual features to be used in the analytical form of the visual servoing. By keeping perception algorithm and controller decoupled, we could preserve flexibility and modularity of the framework. In fact, our neural perception could be used in different control structures.

We have shown promising results, and future work will be devoted to achieving a higher performance. To this end, it might be necessary to sophisticate and redesign the architecture of our NN. Further development will focus on the contribution brought by the different loss terms. By modifying the state consistency loss to consider time horizons longer than one timestep, we render the consistency task harder to solve, thus increasing the supervision provided and resulting in a higher model performance. In the future, we will also address the sim-to-real gap by exploiting domain adaptation techniques, some of which can be easily implemented thanks to the auto-encoding part of our model [62]. These aspects will help on improving the estimation of the tracked object orientation. Bayesian optimization techniques [132, 151] can

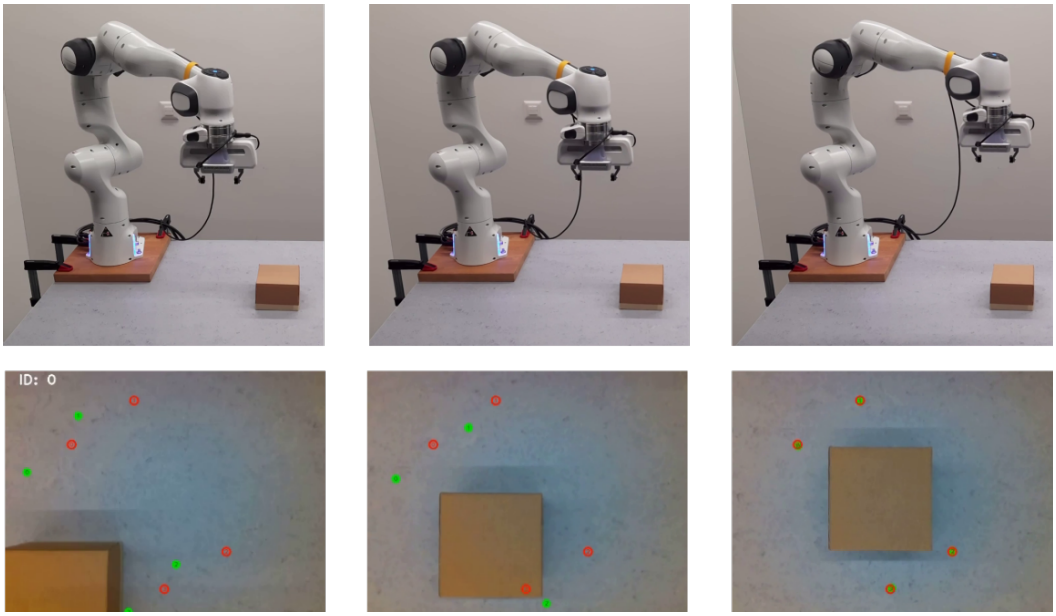


Figure 6.4. Execution of the VS task with the real robot manipulator.

be used to optimally weigh the different loss functions during the training. Also, the perception model could be trained to optimize the task performance so that the estimated feedback could be even more reliable for the downstream visual controller.

The current approach has been developed by considering RGB images as the source of information. However, the approach can be generalized to different sources of sensory information, such as depth, point clouds, or even sound; the network design can be changed to handle these sources without much effort. Finally, we plan to carry out more experiments, training our perception model with different type and number of features and test it in different control contexts and with more complex target objects.

So far, we have explored self-supervised robot learning approaches to collect data autonomously, and improve the performance of deep learning models trained in this paradigm with state-consistency and uncertainty awareness. In the next chapter, we look at self-supervised robot learning from the machine learning perspective, where an auxiliary task, named pretext, is used to further improve models' performance, and how to apply these approaches to spatial perception tasks where only few labelled examples are available.

Chapter 7

Supervision From Sound

So far, we have explored self-supervised robot learning to autonomously collect data for the task of interest. In the machine learning literature, instead, a common self-supervised strategy is the use of pretext tasks (see Chapter 2). These tasks are auxiliary, meaning that their solution is not the ultimate objective; instead, they are introduced with the goal of improving the performance on the task of interest, called end task. In this chapter, we propose to use self-supervised robot learning to autonomously collect labels for the pretext task. In turn, this enhances the learned model, which can solve its perception task more accurately and reduces the need for a large number of labeled samples for the end task. Specifically, we study the use of microphones as inexpensive sensors for providing labels to the pretext task. Furthermore, this approach leverages the complementary information available in auditory and visual modalities to improve the performance of a perception model that relies on visual information.

7.1 Background

Robot perception tasks often involve estimating spatial information, such as the pose of an OoI, from high-dimensional data, e.g., images acquired by onboard cameras; deep learning models such as CNN are a standard tool to solve this kind of problems. If no pre-trained model is available for a task of interest, one standard approach is to acquire large labeled training datasets, as representative as possible of the environment in which the robot will be deployed. Each image in the dataset is labeled with the corresponding relative pose of the OoI, obtained for example through an external tracking system. Then, one trains a deep learning model in a supervised way. However, the acquisition of such labeled datasets is expensive and not always feasible. Therefore, recent research leverages semi-supervised and self-supervised learning approaches, which combine a (typically small) labeled dataset with a large unlabeled dataset; in robotics applications, the latter can be acquired efficiently, by the robot itself, even during deployment.

A common strategy consists in using the unlabeled dataset for training an autoencoder; the encoder part is then used as a feature extractor for learning the perception task of interest in a supervised way using the labeled dataset [64]. Recent advances in the field of self-supervised learning bring this idea further. To solve a task of interest (*end task*), for which a limited labeled dataset is available, it is advantageous to simultaneously learn auxiliary *pretext tasks*, that are defined on large unlabeled datasets. In this context, pretext tasks should: i) require similar perception skills as the end task; ii) not require the explicit acquisition of ground truth

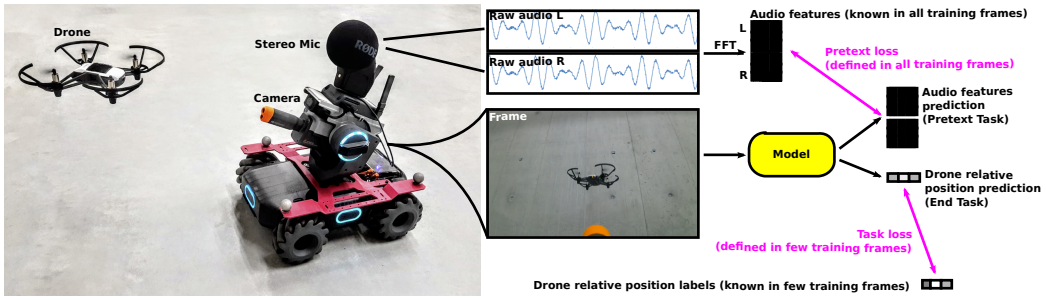


Figure 7.1. Given an image from the ground robot's camera, our model estimates the relative position of the quadrotor; this is the *end task*, learned by minimizing a regression *end loss* on few training frames for which the true relative position is known. We show that simultaneously learning to predict audio features (*pretext task*), which are known in all training frames, yields significant performance improvements on the end task.

labels, but instead use information in the data itself for supervision (hence, "self-supervised"). The intuition is that auxiliary tasks force the model to recognize patterns in the input that are similar to those that must be recognized to solve the task of interest, and thus aid in learning meaningful intermediate representations, as detailed in Chapter 2.

In this work, we apply this approach to a ground robot for learning a visual estimator of the position of a flying quadrotor, given the following training data: a small number of labeled samples for which the relative position of the drone is known; and a large number of unlabeled samples. The ground robot is also equipped with an uncalibrated stereo microphone, mounted at an arbitrary but fixed pose with respect to the camera, which picks up the noise of the quadrotor (see Figure 7.1).

In this context, one could learn to estimate the drone position using the audio signal as an input to the model – a topic covered by the literature concerning sound source localization (see Section 7.1.1). Our work does *not* aim at this goal. Instead, we use auditory perceptions to define a pretext task that aids in learning a purely-visual estimator; our model has one input (the camera frame) and predicts two outputs: the relative position of the drone (end task) and the intensity at different frequency bands of the corresponding sound (pretext task); the model is trained by minimizing the sum of the two respective regression losses: one (end loss) defined only on labeled instances; the other (pretext loss) defined on all instances. This is a novel application to robotics of self-supervised learning techniques, and represents our **main contribution**. After training is completed, one can ignore the pretext task: the model only relies on visual data and can operate in arbitrarily noisy environments, or on robots without a microphone. The approach is validated with experiments on extensive real-world datasets described in Section 7.2, which we release alongside the paper as a **secondary contribution**. Results show that learning with the sound-prediction pretext task yields models that perform significantly better than two baselines: one using no pretext task, and one using a standard image-reconstruction pretext task with an undercomplete autoencoder.

7.1.1 Related Work

Audio is a rich source of supervision utilized in many recent works to learn useful visual feature representations from data. Learned features are then used to solve classification problems [119,

[6], to learn an audio-conditioned forward model of how an object moves based on a robot action [61], or to identify the region of interest responsible for the sound [118, 7, 86, 130, 8]. Video classification is done by training a CNN to predict audio features from images [119], or by using one network per modality [6]: features extracted from images are clustered into classes and used as labels for the audio network, while audio features are used as labels for the image network. Then, both approaches classify the video by clustering together video frames based on the predicted audio features. Similarly to Owens et al. [119], our approach solves the task of predicting audio features from images; however, we do so in a robotics context to improve the performance on the end task of visually localizing an object from images. Gandhi et al. [61] learn to relate audio features to actions taken by a robot arm: they use the sound collected by microphones mounted on a box, as a robot tilts the box while having an object inside. The audio features learned with this approach are shown to better predict the action taken by the robot than using visual features extracted before and after executing the aciton.

Coarse-grained object localization can be achieved by identifying the region of interest responsible for the sound, obtained by applying class activation mapping to a trained network: Owens and Efros [118] train the network by extracting audio samples and frames from a video and predicting the probability that the two streams are temporally aligned. Instead, Arandjelovic and Zisserman [7] use a triplet loss, in which positive examples are image-sound pairs coming from the same video, while negative pairs are taken from different videos; whereas Korbar et al. [86] train on hard-negatives, generated by sampling sound from the same video but at a different moment. Patrick et al. [130] generalize the approaches above, generating different tasks by choosing which transformation to apply to the data. Training is done by sampling random transformations pairs, and then predicting the probability for the pair to correspond to the same instant of a video. Arandjelovic and Zisserman [8] train a model on image-sound correspondence: the feature map, computed as the scalar product of image and sound features, is used for a coarse-grained localization of the sound source.

Sound source localization

A large amount of literature investigates the problem of sound source localization [143]. Classic approaches compute hand-crafted audio features and algorithms based on sound propagation models to solve the problem; several recent approaches learn features directly from data with NN [167, 197, 168, 55, 71]. Takeda and Komatani [167] propose to use a NN on a directional activator, while Yalta et al. [197] utilize a more modern ResNet architecture on the power spectrogram extracted from the short-time Fourier transform (STFT). When localizing from audio information, approaches suffer from noise present in the environment or produced by the robot itself, and from reverberations coming from different surfaces [143]. Takeda and Komatani [168] tackle these challenges by minimizing the cross-entropy loss computed on data collected in the deployment environment. Ferguson et al. [55] propose to jointly minimize two loss functions: a polar loss based on the direction of the source, and a mean squared error loss based on the distance of the source. He et al. [71] focus on the localization of multiple sources, by estimating a spatial spectrum that can be decoded into the locations of the sound sources. To aid the learning process, the network output is constrained to predictions that are coherent with the number of known sources present in a given sample.

In contrast to this body of work, our goal is not to localize the sound source from audio; instead, we take advantage of the (unknown) correlation between audio features and the position of the OoI – which is expected to emit sound at least while collecting training data – to

learn a better visual feature representation. During deployment, the trained model utilizes the camera feed to estimate the location of the OoI using the corresponding head, while the sound prediction head is left idle.

7.1.2 Model

We consider the problem of estimating from a monocular image the position relative to the camera reference frame of an OoI that, at least during training-data acquisition, produces sound. In addition to the camera video stream, we record audio from a microphone placed at a fixed pose with respect to the camera reference frame. In this context, we learn a NN model from a set of instances

$$\{\langle \boldsymbol{i}_k, \boldsymbol{f}_k, \boldsymbol{p}_k \rangle\}_{k=1}^{N} \tag{7.1}$$

where $\boldsymbol{i}$ denotes a camera frame, $\boldsymbol{f}$ audio features computed from the corresponding audio signals, $\boldsymbol{p}$ the position of the OoI relative the robot's camera. More specifically, the (possibly very small) subset of instances for which $\boldsymbol{p}$ is available, is denoted as the labeled training set $\mathcal{T}_\ell$; remaining instances compose the unlabeled training set $\mathcal{T}_u$, which is assumed to be much larger, as it can be acquired by a robot without external supervision.

We learn a Sound as Pretext (SaP) model that, given the image $\boldsymbol{i}$, estimates: $\hat{\boldsymbol{p}}$, the relative position of the OoI (*end task*); this task is trained using data in $\mathcal{T}_\ell$, and is the task whose performance we are interested in optimizing. $\hat{\boldsymbol{f}}$, the corresponding audio features (*pretext task*); this task is trained using data in $\mathcal{T}_\ell \cup \mathcal{T}_u$, and is of no direct use for model deployment. We represent the relative OoI's position by its image-space position $\boldsymbol{o}$ and its distance d from the camera. The model implements a function parametrized by $\boldsymbol{\theta}$

$$\left(\hat{\boldsymbol{p}}, \hat{\boldsymbol{f}}\right) = \boldsymbol{m}_{\text{SaP}}(\boldsymbol{i}|\boldsymbol{\theta}). \tag{7.2}$$

Training consists in obtaining optimal values of $\boldsymbol{\theta}$ by minimizing through gradient descent the following loss function

$$\frac{\lambda}{|\mathcal{T}_\ell|} \sum_{i=1}^{|\mathcal{T}_\ell|} \mathcal{L}_{\text{end}}(\boldsymbol{p}_i, \hat{\boldsymbol{p}}_i) + \frac{1}{|\mathcal{T}_u \cup \mathcal{T}_\ell|} \sum_{i=1}^{|\mathcal{T}_u \cup \mathcal{T}_\ell|} \mathcal{L}_{\text{pretext}}(\boldsymbol{f}_i, \hat{\boldsymbol{f}}_i) \tag{7.3}$$

where $\mathcal{L}_{\text{end}}$ and $\mathcal{L}_{\text{pretext}}$ both compute the absolute error between their two arguments, $|\cdot|$ denotes the cardinality of a set, and λ acts as a tradeoff on the optimization of the two losses. Note that training instances in $\mathcal{T}_u$ have no label $\boldsymbol{p}_i$.

Application to quadrotor position estimation

In the rest of the paper, we instantiate the general approach presented above to a concrete problem: learn from scratch a visual estimator of the position of a flying quadrotor from images captured by a ground robot, using audio data acquired by an onboard stereo microphone to aid learning. We exploit a small amount of training data $\mathcal{T}_\ell$, in which the position of the drone relative to the camera is known from a motion tracking system; and a large unlabeled training set $\mathcal{T}_u$, acquired by the robot without supervision.

The scenario is attractive because it matches many similar robot perception tasks of wide interest, in which deep learning models must be trained from scratch and large pre-existing labeled datasets are not available. Acquiring those labeled datasets ad-hoc is potentially very expensive and time-consuming; exploiting self-supervision from cross-modal cues is in this context an attractive alternative.

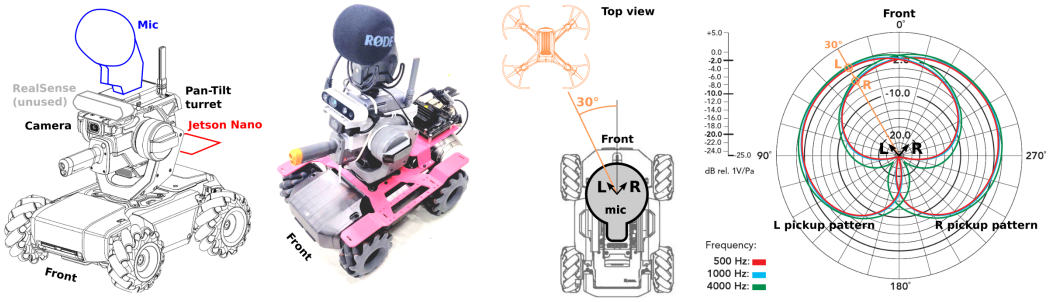


Figure 7.2. From left to right. Technical drawing of the Robomaster S1 platform (black) and some of our addons (gray, blue, red). Photograph of the real platform. Top view of the robot with placement and orientation of the two condenser capsules (L, R) in the stereo mic. Polar pickup pattern for the two cardioid capsules for three different frequencies. Orange elements refer to the example of direct sound coming from a drone at a 30° angle, see text.

7.2 Experiments

We use a modified DJI Robomaster S1 ground robot (see Figure 7.1 and Figure 7.2); the robot features a controllable pan-tilt turret and four Swedish wheels, which provide omnidirectional motion capabilities. On the turret, the robot is equipped with: an integrated RGB camera acquiring images at a resolution of 1280×720 pixels; a stereo microphone (RØDE Stereo VideoMic Pro), mounted in such a way that the frontal direction is aligned with the camera optical axis; an NVIDIA Jetson Nano single-board computer mounted on the back; and an Intel RealSense camera that is not used in our experiments.

The microphone uses a matched pair of high sensitivity 0.5 inch cardioid condenser capsules mounted in a coincident XY stereo configuration [164] (Figure 7.2); therefore, the two capsules point 45° to the left (L) and right (R) of the camera optical axis. Each capsule has a cardioid pickup pattern that attenuates sounds depending on the angle of the sound source with respect to the axis of the capsule, as depicted in Figure 7.2 (right): sounds coming from a direction aligned with the capsule axis are not attenuated (0dB); sounds coming at an angle of 90° are attenuated by approximately 6dB.

As an example, Figure 7.2 shows the case of a drone (orange) hovering at the same height of the microphone, in a position that lies at an angle of 30° to the left of the camera optical axis; we further assume that the microphone is pointing horizontally (turret tilt equal to 0°). Interpreting the polar plot, we expect that the *direct* noise from the drone will be attenuated by approximately -0.5dB in the left channel (15° with respect to the capsule axis), and by approximately -4dB in the right channel (75° with respect to the capsule axis); note that in most indoor environments including ours, a significant part of the drone noise will not originate directly from the drone direction, but instead be reflected by the surrounding environment as reverberations. Note also that low frequencies (red) are attenuated more than high frequencies (green). Noise and rumble induced on the microphone from robot movements and vibrations can be attenuated with an optional anti-shock acoustic suspension (RØDE Rycote Lyre). It is important to remark that the configuration of the microphone setup is not assumed known in our system; our only assumption is that there exists some unknown, potentially weak, and potentially nonlinear correlation between the drone relative position (target variables for the end task) and the corresponding audio features (target variables for the pretext task). If this

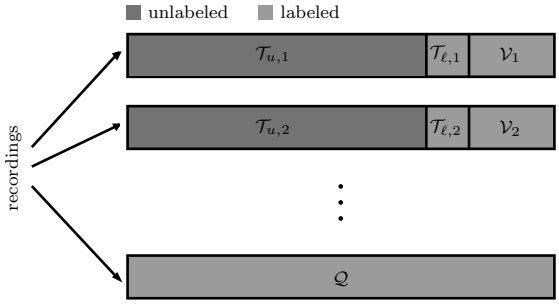


Figure 7.3. Data is collected in 22 recording sessions, and then split in unlabeled ($\mathcal{T}_u$) and labeled ($\mathcal{T}_\ell$) training sets; validation set ($\mathcal{V}$); and testing set ($\mathcal{Q}$).

is the case, solving the pretext task indirectly favors learning visual features that encode the drone position.

The drone we use in our experiments is a Ryze Tello quadrotor; the Robomaster turret and the Tello body are outfitted with infrared reflective markers and tracked during data collection by a motion tracking system (12 Optitrack Prime-13 cameras), which provides the precise Tello position w.r.t. the RobotMaster's camera frame[1] used as labels in the training procedures.

7.2.1 Data Collection

We collect data in 22 different recording sessions taking place in a laboratory environment, comprising a total of approximately 50 minutes of data. During each session, the Tello is teleoperated by a user with a joypad, flying 3D trajectories that attempt to stay within a maximum distance of 3m from the Robomaster, such that the drone remains visible in the scaled-down camera feed used for learning (128×80 pixels); at the same time, a different user teleoperates the Robomaster to move in the environment and pan/tilt the turret to keep the Tello visible in the camera feed – this ensures that the image background is very variable in the entire dataset. For each session, we record camera frames at 30Hz, raw stereo audio from the microphone, and absolute pose information of the Robomaster turret and Tello body. A set of instances is then generated associating to each camera frame $\boldsymbol{i}$ the corresponding audio features $\boldsymbol{f}$ and relative drone position $\boldsymbol{p}$ with respect to the camera. Following [194], the position is represented as the normalized coordinates u and v of the drone as it appears on the image plane, and its Cartesian distance d from the camera. Instances in which the drone is not visible (defined as having the drone position projected in the image plane out of the bounds of the camera frame) are discarded.

All instances from one recording are used in the following as a testing set $\mathcal{Q}$, on which we compute performance metrics. Instances from the remaining 21 recordings are partitioned into three sets: the first 70% of the frames as the unlabeled training set $\mathcal{T}_u$, the following 10% for the labeled training set $\mathcal{T}_\ell$, and the remaining 20% for the labeled validation set $\mathcal{V}$, which is only used to monitor models during training (see Figure 7.3). In total, we collected 89k instances, of which 60k are in $\mathcal{T}_u$, 8k in $\mathcal{T}_\ell$, 17k in $\mathcal{V}$, and 4k in $\mathcal{Q}$.

[1] The offset between the OptiTrack markers placed on RobotMaster turret and its camera frame is taken into account.

7.2.2 Audio Features

Audio features corresponding to a given frame acquired at time t are extracted from raw audio data acquired in a time window of 0.15s centered on t (6615 audio samples at 44.1 kHz); within this short time window, we assume that the quadrotor noise profile is approximately stationary. For each of the two channels, we use the discrete Fourier transform to compute the logarithm of the average magnitude of the frequency spectrum within each of three frequency bands: 1 to 2kHz; 2 to 5kHz; and 5 to 15kHz. This yields 3 features for each channel, which can be interpreted as the values of a log-spectrogram of the audio signal, sampled at a single timestamp, and average-pooled along the frequency axis. We further compute the difference between the features in the two channels, for a total of 9 features, because we expect intensity differences between the two channels to be correlated to the drone position relative to the robot camera and mic. We consider multiple frequency bands since, depending on the angle of the sound source, each band is attenuated by a different amount, and that environmental reflections are more prevalent in some frequencies than in others. In contrast, because our microphone uses virtually-coincident positions for the two capsules, our approach can not rely on inter-channel time difference cues [148].

7.2.3 Alternative Strategies

To validate the rationale of our approach, we analyze and compare its performance with several alternatives. First, we train the Baseline (B) model only on the end task using data in $\mathcal{T}_\ell$, i.e., $\mathcal{L}_{\text{pretext}}$ is not considered in (7.3).

To estimate the maximum achievable performance, we train the Upper Bound (UB) model, using only the end task as in the previous case, but considering the whole training set as if it was all labeled. The model is trained with labels for $\mathcal{T}_u$ that we collected only for this purpose and ignored for all other models.

A common approach in the robotics literature to exploit unlabeled image data is to train on such data an undercomplete autoencoder [64], and use the resulting compressed representation as features for subsequently learning the end task; the expectation is that these features encode high-level information in the original image. In our context, this corresponds to using autoencoding as a pretext task. Therefore, we train the Autoencoding as Pretext (AaP) model, by simultaneously considering the end task (on $\mathcal{T}_\ell$), as well as the autoencoding pretext task (on $\mathcal{T}_u \cup \mathcal{T}_\ell$).

Taking inspiration from sound source localization approaches, we consider the Audio Only (AO) model which is trained solely on regressing the position from audio features $\boldsymbol{f}$, using $\mathcal{T}_\ell$. Similar to other audio-only approaches [167], it is negatively affected by noise generated by the robot itself, environment reverberations, and other external sound sources, which makes the approach less desirable for deployment in unstructured scenarios.

When provided with data from multiple sensors, a common approach in the literature is to fuse readings coming from different sensors together [195, 178]. To explore this option, we train the Sensor Fusion (SF) on the task of regressing the drone position from both image and audio features, using only instances in $\mathcal{T}_\ell$.

We also compare SaP, trained using our 9-dimensional audio features, with two models trained using different audio features. The first, SaP-Mono, uses only 3 features ($\boldsymbol{f}_{\text{Mono}}$) obtained as the average of the features for left and right channels, actually emulating a mono-channel microphone. The latter, named SaP-Mel, is trained using a richer audio representation

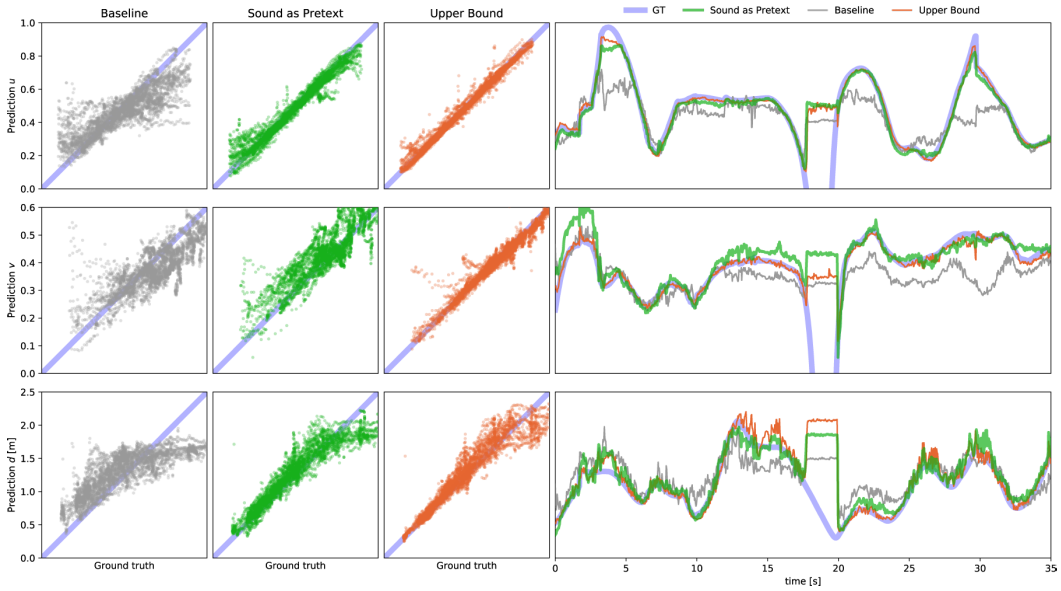


Figure 7.4. End Task Regression Performance on the testing set $\mathcal{Q}$. On the left side we compare ground truth (x-axis) and predictions (y-axis) for different models (columns) and variables (rows). On the right, predictions on 35s of the testing set. Between seconds 17 and 20 the drone exits of the camera FOV, causing all models to temporarily fail.

($\boldsymbol{f}_{\text{Mel}}$) based on Mel-spectrograms [140]. More specifically, we compute for each channel a 64-band Mel-spectrogram in the range 1 to 15kHz, using 0.15s windows; for each frame, we associate the 64×2 values sampled from the two spectrograms at the corresponding time, plus their difference, for a total of 64×3 audio features.

7.2.4 Network Architectures and Training

Most of the strategies considered in this work employ a CNN architecture based on MobileNet-V2 [154], with a total of 1 million parameters, and a variable number of output neurons dependant on the chosen strategy (3 for strictly supervised approaches, 12 for the SaP model). The SF model utilizes the same convolutional architecture for the image branch, while a series of 4 feed-forward layers with ReLU non-linearities processes the audio information and 3 more feed-forward layers fuse the two streams, similarly to [178]. The AaP model implements an encoder-decoder CNN architecture, with a bottleneck of size 128 [64]. A separate head takes the latent representation and through a series of 3 feed-forward layers with ReLU non-linearities produces a prediction of the position. The AO model is composed of a series of 5 feed-forward layers with ReLU non-linearities, for a total of 60k parameters. For all models, training uses the Adam [83] optimizer with an initial learning rate of 10^{-3}, which is reduced by a factor of 10 halfway through the training process, which lasts a total of 60 epochs. In designing our loss, we choose a tradeoff factor $\lambda = 1$. Batches of size 64 (or possibly less, for the last one) are drawn from either of the two training datasets, and the corresponding loss is used for the optimization.

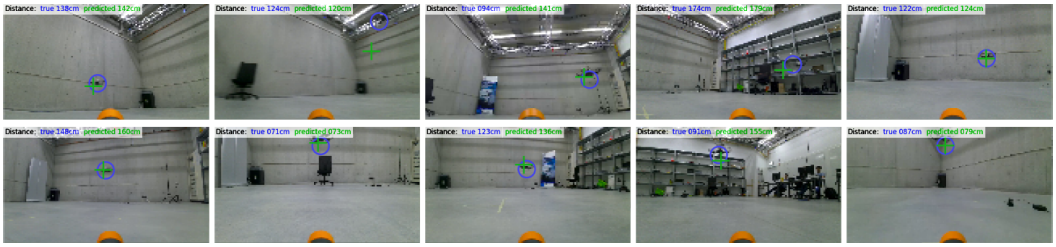


Figure 7.5. Predictions of the Sound as Pretext model (green cross) compared to ground truth (blue circle) on ten frames taken from the testing set.

Model	Input	Output	Training set for End Task	Training set for Pretext Task	MAE_{uv} [px]	MAE_d [cm]	R^2_u [%]	R^2_v [%]	R^2_d [%]
Dummy	–	–	–	–	14	50	0	−4	−23
B	$\boldsymbol{I}$	$\boldsymbol{o}$	$\mathcal{T}_\ell$	–	7	28	66	56	58
AaP	$\boldsymbol{I}$	$\boldsymbol{I}, \boldsymbol{o}$	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	14	47	−2	−9	−12
SaP-Mono	$\boldsymbol{I}$	$\boldsymbol{f}_{\text{Mono}}, \boldsymbol{o}$	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	6	15	83	52	86
SaP-Mel	$\boldsymbol{I}$	$\boldsymbol{f}_{\text{Mel}}, \boldsymbol{o}$	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	6	22	82	66	71
SaP *(ours)*	$\boldsymbol{I}$	$\boldsymbol{f}, \boldsymbol{o}$	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	**4**	**14**	**94**	**68**	**87**
SaP-60	$\boldsymbol{I}$	$\boldsymbol{f}, \boldsymbol{o}$	60% $\mathcal{T}_\ell$	60% $\mathcal{T}_\ell \cup \mathcal{T}_u$	7	24	72	53	63
B-60	$\boldsymbol{I}$	$\boldsymbol{o}$	60% $\mathcal{T}_\ell$	–	13	40	10	−28	14
SaP-30	$\boldsymbol{I}$	$\boldsymbol{f}, \boldsymbol{o}$	30% $\mathcal{T}_\ell$	30% $\mathcal{T}_\ell \cup \mathcal{T}_u$	8	31	73	13	45
B-30	$\boldsymbol{I}$	$\boldsymbol{o}$	30% $\mathcal{T}_\ell$	–	14	43	−3	−12	−4
SaP-10	$\boldsymbol{I}$	$\boldsymbol{f}, \boldsymbol{o}$	10% $\mathcal{T}_\ell$	10% $\mathcal{T}_\ell \cup \mathcal{T}_u$	9	32	68	−55	47
B-10	$\boldsymbol{I}$	$\boldsymbol{o}$	10% $\mathcal{T}_\ell$	–	15	49	−2	−88	−21
AO	$\boldsymbol{f}$	$\boldsymbol{o}$	$\mathcal{T}_\ell$	–	7	18	87	−23	81
SF	$\boldsymbol{I}, \boldsymbol{f}$	$\boldsymbol{o}$	$\mathcal{T}_\ell$	–	6	27	83	68	60
UB	$\boldsymbol{I}$	$\boldsymbol{o}$	$\mathcal{T}_\ell \cup \mathcal{T}_u^\star$	–	2	11	98	93	91

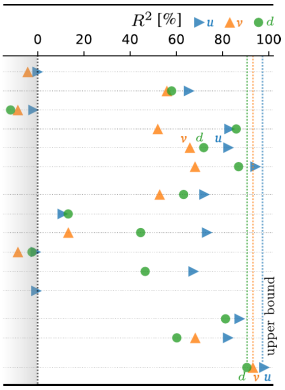


Figure 7.6. Summary of results; each model reports the position MAE (lower is better) and the R^2 metric for each output variable (higher is better).

7.2.5 Sound Prediction as a Pretext Task Results

This section explores the effectiveness of our approach against alternatives (Section 7.2.5), different choices for the sound features (Section 7.2.5), the impact on the amount of labeled data (Section 7.2.5), and a comparison with approaches that use audio features as inputs (Section 7.2.5). For our evaluation, we consider the following metrics computed on the entire testing set $\mathcal{Q}$: the mean absolute error (MAE) of the prediction w.r.t. the ground truth, computed separately for uv (MAE_{uv}, expressed in pixels), and distance (MAE_d, expressed in cm); and the coefficient of determination R^2 for each of the three components of the relative position, denoted with R^2_u (u image coordinate), R^2_v (v image coordinate) and R^2_d (distance from the camera). The coefficient of determination is a standard adimensional metric for regression performance. It represents the fraction of the variance of the target variable that is correctly explained by the model (higher values are better). A trivial model that predicts the average of the target variable in the whole testing dataset yields $R^2 = 0$; an ideal model yields $R^2 = 1.0$; a model might have a negative R^2 in case its MAE exceeds the variance of the data, which frequently occurs with weak models operating on high-dimensional inputs. The R^2 metric enables meaningful comparison of regression performance on different variables, since, unlike the MAE, it does not depend on the variance of the target variable.

Sound prediction as a pretext task improves performance

In the top section of Figure 7.6, we compare the performance of our SaP approach with baselines, alternatives, and a Dummy model that simply returns the average of each target variable in the training set. We observe that compared to the baseline B, our approach significantly improves performance on u and d, and to a lesser extent on v. Considering the relative improvement in terms of R^2 of each variable, where B represents the 0% reference and UB is considered as the maximum achievable performance (100%), our approach reaches 87% improvement on u, 32% on v and 88% on d.

The u and d variables are most directly related to audio features: for example, u affects the intensity difference between the two audio channels, and d the intensity on both channels; on these two variables, SaP yields large improvements over the baseline. This is because the pretext task induces the model to learn visual features that capture well the horizontal position of the quadrotor and its distance. In contrast, the v variable is only weakly and non-monotonically related to audio features (and in fact, v can not be estimated well by the AO model). Still, our pretext task significantly improves performance on the v variable; one possible explanation is that the same visual features that capture u and d are also useful to estimate v.

Figure 7.4 shows scatter plots comparing ground truth to predictions of the baseline, our approach, and the upper bound, as well as a qualitative comparison of their predictions on 35s of data taken from the testing set. Considering the scatter plots on the left, our approach improves over the baseline, having a tighter distribution that is closer to the diagonal line representing the ideal case; this confirms the quantitative evaluation of Figure 7.6. It can be noticed how all models correctly predict the distance d when the drone is close to the camera, while longer distances are harder to estimate. Regarding the time plot on the right in Figure 7.4, comparing different model predictions on a portion of $\mathcal{Q}$ shows that our approach follows closely the upper bound, while the baseline struggles when the drone is not in the central area of the image. Between seconds 4 and 5 all models struggle in predicting the horizontal location of the drone: this is explained by the drone moving close to the edge of the field of view; similarly, between seconds 17 and 20, the drone briefly exits the camera's field of view, causing all models to predictably fail. In Figure 7.5 we present a qualitative evaluation of SaP on 10 camera frames taken from the testing set. Failure cases, in which the model's prediction (green) does not overlap with the ground truth (blue), occur when the drone blends with a cluttered background, or when it reaches a distance greater than 3m, rendering the quadrotor recognition from very few pixels difficult.

Impact of sound features

To further explore this fascinating finding, we also trained SaP-Mono, the version of our approach that uses as the target of the pretext task only 3 features. Without access to stereo information, audio features do not allow discriminating whether the drone is at the left or right side of the image, but still allow good resolution concerning the drone distance; therefore, the beneficial effects of the cross-modal pretext task are reduced on u and v, when compared with its stereo counterpart, while on d SaP-Mono and SaP perform equally well. Predicting richer audio features as the pretext (SaP-Mel) still exhibits improvements over the baseline, but not as much as with our features; this is probably due to the larger dimensionality of Mel features, which exposes the model to overfitting issues while containing limited additional useful information when compared to our features. Whereas solving an autoencoding pretext task

(AaP) yields very poor results, with a performance well below the baseline B. The reason is that autoencoding is a poor pretext task in this scenario: the drone is most often small in the input image and covers a small fraction of pixels; the autoencoding loss minimizes the image reconstruction error, and will not promote representing meaningful information concerning the drone position. Instead, we expect that the learned features will be dominated by modeling the different possible backgrounds, which cover most of the image and exhibit very high contrast. Unfortunately, the image background is exactly what we want our features to be insensitive to.

Impact of the labeled set size

To explore the impact of the amount of available labels for the end task, we trained the SaP model on decreasing amounts of labeled data while keeping the unlabeled data fixed. The second panel of Figure 7.6 reports the respective results; thanks to the effectiveness of the sound pretext task in learning meaningful visual features, the u variable can be estimated well even with 10% of the labeled data, corresponding to just 800 frames, while d requires a larger amount of labels. In contrast, the performance on v rapidly drops as fewer labeled instances are considered.

Strategies using sound as input

The previous analysis compared our approach against others using just images as input; we now extend our focus on strategies that utilize different modalities, whose results are reported in the third panel of Figure 7.6. The AO model shows promising results on the variables u and d while having no predictive power on the v. This is easily explained by the microphone's geometry, for which the difference between sound intensity in the two channels is highly informative on the horizontal axis but not on the vertical axis; humans and animals overcome this issue by accounting for different spectral filtering of the ear geometry on sound coming from different elevations, or by actively tilting the head to better localize sound sources. Model SF can leverage images to estimate v, resulting in a higher R^2 score on that variable; while on u SF has a similar performance as AO, it is penalized on the distance d. In fact, images yield little additional information to audio when predicting distance, especially when the drone is far from the camera. Compared to all alternatives, SaP performs better on all three variables.

7.3 Discussion

We presented Sound as Pretext, an approach for tackling visual object localization problems by employing a cross-modal pretext task, well suited to many applications of self-supervised robot learning. By collecting images of the quadrotor as well as its noise, we alleviate the need for a large labeled training dataset, and provide supervision to the model by adopting the auxiliary pretext task of predicting audio features from images. The approach requires only that the object to be localized produces sound during training data collection – a condition that, for silent objects, could be satisfied with the help of a wireless speaker – while during deployment no audio information is necessary. An extensive evaluation shows that our approach outperforms a supervised baseline, a standard image-reconstruction pretext task, and approaches that directly use audio, or a combination of vision and audio, to solve the same task.

The sound-based pretext task is powerful but relies on the assumption that the source produces a stationary sound, such as that of drone's rotors or other motors. If, instead, the sound is

non-stationary and dynamic, the relationship between the source's sound and its location would be weaker because it would also depend on time and other factors. Thus, the effectiveness of the pretext task may vary based on the sound profile of the object or robot to be localized. In the next chapter, we explore a purely-visual pretext task completely decoupled from the OoI's movements.

Chapter 8

Supervision From LEDs

In the previous chapter, we considered the use of self-supervised robot learning for the automated collection of labels for a sound-based pretext task. By solving this pretext task, a model learns a better feature space, leading to improvements in performance and lessen the reliance on end task labels. Despite the success of the approach, its scope is limited to distinctive and constant sound sources, such as drone's rotors, and fails to consider that robot movements affect the sound it makes. These issues imply that, for very dynamic robots or when using recurrent neural network models, the effectiveness of this pretext task is remarkably reduced. In this chapter, we decouple end and pretext tasks, studying the use of controllable LEDs as the base of a vision-only pretext task. Controllable LEDs are independent of the robot's position or movements, are featured on practically all robot platforms and, therefore, are a convenient source of supervision.

8.1 Background

The ability to estimate the position of a target robot in a video feed is crucial for many robotics tasks [165, 131, 32]. SoA approaches use deep learning techniques based on CNNs [92]: given a camera frame, they segment the target robot, regress the coordinates of its bounding box or its position in the image. Training these approaches to handle new robots or environments requires extensive labeled datasets, which are time-consuming and expensive to acquire, often relying on specialized hardware, e.g., motion tracking systems, to generate ground truth labels.

We present an approach to drastically reduce the labeled data required to train such models, building upon recent results in Self-Supervised Learning [80]. In the robotics literature, the term Self-Supervised denotes two distinct paradigms. In the first, a robot system autonomously generates labeled data for the task of interest, named *end task*, and is trained in a standard supervised way. This paradigm has been used in robotics since the mid 2000s [43, 98, 68, 200]. As a recent example, Li et al. [92] use nano-drones equipped with SoA algorithms to automatically acquire camera frames and the corresponding relative location of the target drone. In the second paradigm, a robot system autonomously generates abundant labeled data for a *pretext task*: the pretext task requires similar perception skills as the end task while relying on cheaper ground truth that is easier or free to collect. Then, a model is trained to solve both tasks simultaneously using an additional dataset containing only few labels for the end task. Despite not being useful during deployment, the pretext task forces the model to learn mean-

ingful features, boosting the performance on the end task. This paradigm is widely successful in the deep learning literature [80] and has only recently been adopted for robot perception applications [116, 142].

This work introduces a novel approach based on the second paradigm, tailored to robotics applications, and suitable for deployment on resource-constrained platforms. Our **contribution**, presented in Section 8.1.2, is the use of target robot LED state prediction (on or off) as a pretext task to improve the learning process of a visual localization end task. By learning to predict the state of the LEDs aboard, the model learns features that are also useful to localize the target robot. The idea is compelling because most robot platforms feature controllable LEDs: during data collection, the target robot blinks its LEDs and radio-broadcast their state; at the same time, another robot automatically collects images annotated with LED state ground truth.

We instantiate this general idea to a specific, challenging end task: predict the image-space position of a target nano-drone given a low-resolution, low-dynamic-range image acquired by the camera of a peer nano-drone. A Fully Convolutional Network (FCN) model [97] simultaneously learns to solve pretext and end tasks using the dataset described in Section 8.2, containing only few samples labeled with the drone's location. We provide detailed comparisons and an ablation study on the Bitcraze Crazyflie 2.1[1] nano-drone in Section 8.3. Results show the proposed pretext task to significantly improve performance over a supervised baseline, different pre-training strategies, and an autoencoding pretext task. The model generalizes well to unseen environments, and is capable of localizing multiple drones simultaneously. Finally, we deploy the model aboard the target platform to complete a vision-based position tracking task.

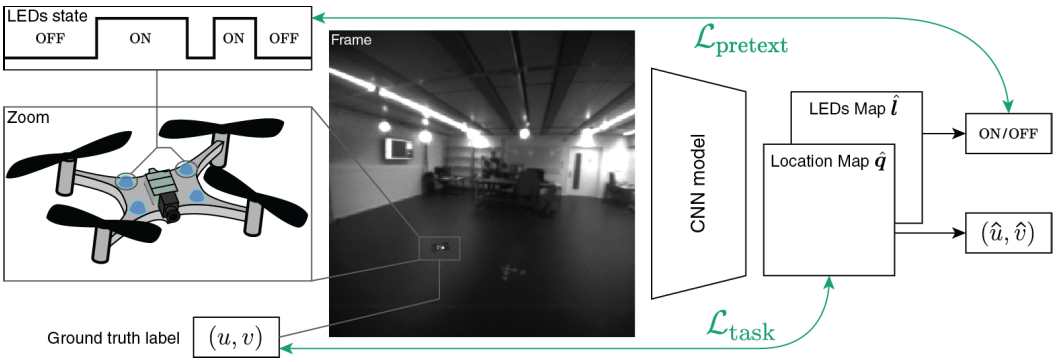


Figure 8.1. A fully convolutional network model is trained to predict the drone position in the current frame by minimizing a loss $\mathcal{L}_{\text{task}}$ defined on a small labeled dataset $\mathcal{T}_l$ (bottom), and the state of the four drone LEDs, by minimizing $\mathcal{L}_{\text{pretext}}$ defined on a large dataset $\mathcal{T}_l \cup \mathcal{T}_u$ (top).

8.1.1 Related Work

Drone-to-drone relative localization approaches rely on various sensors, including microphones, infra-red sensors, UWB, color and depth cameras; examples of such tasks are described in Section 2.5.1. In particular, microphones can be used for localization, integrating distance estimates from a drone beacon emitting a specific sound [12]. Multiple infra-red sensors with known geometry allow the triangulation of a drone equipped with infra-red emitters [147].

[1] https://www.bitcraze.io/products/Crazyflie-2-1

Camera-based approaches rely on visual fiducial markers such as circles printed on paper [155], light-emitting markers [46], by detecting the drone in depth images with handcrafted [181], or learned [25] models. In our work we use monocular grayscale images as the model's input. LEDs, which come already integrated with the adopted platform, are exclusively used to generate data for the self-supervised pretext task and are not used during inference.

UWB is a wireless communication technology recently adopted for localization tasks [65, 117, 39, 179, 196], enabling communication between multiple robots and providing a distance measurement through the Received Signal Strength Intensity (RSSI). RSSI measures the amount of radio signal received from a source and is used to derive its distance. Using three non-colinear UWB sensors enables the triangulation of robots [65]. A single sensor requires more complex approaches, such as integrating distance measurements from UWB beacon drones moving in a pattern [117]. Communication is used during localization to combine distance measurements with broadcasted state-estimates [39, 179] and optimizing a camera-based initial guess [196].

Self-supervised relative drone localization approaches focus on learning a model with limited access to labeled data, using UWB to provide ground truth [92], or a stereo microphone for an audio-based pretext task, described in Chapter 7. In detail, Li et al. [92] pre-train a purely visual estimator using synthetic data, then fine-tune it using a small labeled dataset generated autonomously from UWB nodes [179]. In contrast, our approach introduces a pretext task defined on images with no ground truth positions; our pretext is based solely on LED state estimation and does not require additional hardware.

We explored cross-modal self-supervised learning of visual quadrotor localization in Chapter 7, using images acquired by a ground robot equipped with a stereo microphone. The pretext task consists of predicting features (intensity in various frequency bands) of the perceived sound of a quadrotor, given an image. By solving this pretext task, the model is forced to learn features of the perceived sound that, in turn, are informative of the drone's location.

The present work proposes a more general pretext task that does not rely on additional sensors (e.g., a microphone), is suitable for applications with limited power budget and supports the localization of multiple robots. Further, it only assumes that the target robot is able to vary its appearance using actuators, with controllable LEDs being a natural and convenient choice.

8.1.2 Model

We consider visual robot-to-robot localization problems, in which an observer robot has to predict the position of a target robot on the image plane. The observer robot takes a monocular image from its forward-looking camera and predicts the position of the target robot visible in the image. Additionally, we require the target robot to be equipped with controllable LEDs.

We collect tuples consisting of

$$\left\{\langle \boldsymbol{i}_j, \boldsymbol{o}_j, l_j \rangle\right\}_{j=1}^{N} \tag{8.1}$$

where $\boldsymbol{i} \in \mathbf{R}^{whc}$ denotes a camera frame of $w \times h$ pixels and c channels, $\boldsymbol{o} \in \mathbf{R}^2$ the image-space position of the robot, and l the state of the four LEDs on the robot, which can be either all off or on, represented respectively by 0 and 1.

In the following, we call samples *labeled* when the drone's position $\boldsymbol{o}$ is known or *unlabeled* otherwise. We denote the set containing the (possibly small) amount of labeled samples with $\mathcal{T}_\ell$ and the unlabeled set with $\mathcal{T}_u$. We also collected a separate labeled set $\mathcal{Q}$ that serves as a testing set and on which we compute performance metrics.

Using this data, we learn an NN model $\boldsymbol{m}$ that, given a monocular image, predicts two maps: the location map $\hat{\boldsymbol{q}}$ containing likely drone locations and the LED state map $\hat{\boldsymbol{l}}$ the probability of seeing a drone with its LEDs on,

$$\left(\hat{\boldsymbol{q}}, \hat{\boldsymbol{l}}\right) = \boldsymbol{m}\left(\boldsymbol{i}|\boldsymbol{\theta}\right) \tag{8.2}$$

where $\boldsymbol{\theta}$ is the set of trainable network weights.

Using a map to represent the drone's location has two advantages compared to using the drone's coordinates [20]. First, it allows one to handle images with zero, one or more visible drones [92]. Second, it enforces an inductive bias by limiting the receptive field of each cell of the output map; in fact, we expect the target drone to cover a small portion of the input image [152].

A ground truth location map $\boldsymbol{q} \in [0,1]^{wh}$ of $w \times h$ cells is generated from the robot's position $\boldsymbol{o} = (u, v)$: we start with a map filled with zeros and place a circle of radius $r = 4$ pixels centered in $\boldsymbol{o}$, filled with ones and with a soft-edge transitioning to zero.

We train $\boldsymbol{m}$ by optimizing the weights $\boldsymbol{\theta}$ through gradient descent steps, minimizing the loss function $\mathcal{L}$. The loss, in turn, is defined as the weighted sum of two terms: the first term $\mathcal{L}_{\text{task}}$ consists of a regression loss computed on the labeled training set $\mathcal{T}_\ell$, whose aim is to learn the robot localization task, and defined as

$$\mathcal{L}_{\text{task}} = \frac{1}{|\mathcal{T}_\ell|} \sum_{i=1}^{|\mathcal{T}_\ell|} \text{mean}\left(|\hat{\boldsymbol{q}}_i - \boldsymbol{q}_i|^2\right) \tag{8.3}$$

where mean is the average of the map cells. The second term $\mathcal{L}_{\text{pretext}}$ consists of a classification loss defined on the union of the training sets $\mathcal{T}_\ell \cup \mathcal{T}_u$, learning the LED state prediction task, and defined as

$$\mathcal{L}_{\text{pretext}} = \frac{1}{|\mathcal{T}_u \cup \mathcal{T}_\ell|} \sum_{i=1}^{|\mathcal{T}_u \cup \mathcal{T}_\ell|} \text{BCE}\left(\text{mean}\left(\hat{\boldsymbol{l}}\right), l\right) \tag{8.4}$$

where BCE is the binary cross-entropy. To obtain the scalar $\hat{l}$ representing the probability of the drone's LEDs being on, we compute the average of the LED state map $\hat{\boldsymbol{l}}$[2].

The complete loss function is $\mathcal{L} = (1-\lambda)\,\mathcal{L}_{\text{task}} + \lambda\,\mathcal{L}_{\text{pretext}}$, where $\lambda \in [0,1]$ controls the tradeoff between the two loss terms during training. In (8.3) and (8.4), each loss is weighted by the reciprocal of the dataset size on which it operates, ensuring that the impact of each loss during training is comparable when working on differently-sized datasets.

8.2 Experiments

In the following, we instantiate the presented approach to the challenging task of drone-to-drone localization, as shown in Figure 8.2. This task represents the broader set of image-based robot localization tasks, as many mobile robots feature cameras and controllable LEDs. Among platforms to which our approach is applicable, we specifically selected nano-drones for our experiments: they are difficult to localize due to their small dimensions and complex shape. Additionally, they have constrained resources: the camera is low resolution, low dynamic range, and has a limited field of view; the onboard microprocessor imposes limitations on the breadth and depth of the NN, especially for real-time applications.

[2] Training attempts with an average of $\hat{\boldsymbol{l}}$ weighted by the location map $\hat{\boldsymbol{q}}$ were unsuccessful; it resulted in less stable training and a lower performance.

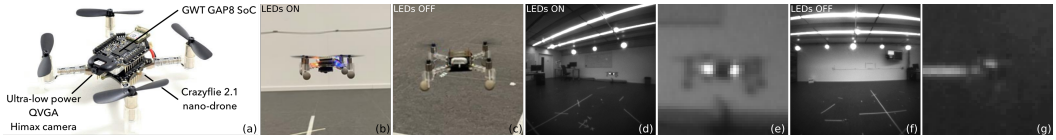


Figure 8.2. The palm-sized Bitcraze Crazyflie 2.1 nano-drone platform (10 cm in diameter). (a) The drone's hardware and its four controllable LEDs; (b, c) high-resolution pictures of the flying drone; (d, f) samples from our dataset; (e, g) zoom-in on the drone using the model's receptive field (45 × 45 pixels).

8.2.1 Robot Platform

The platform of choice is the Bitcraze Crazyflie 2.1, a nano-drone measuring 10cm in diameter and weighing only 27g, extended by the Ai-deck companion board, see Figure 8.2. The Ai-deck provides a forward-looking monocular camera, acquiring 320×320 pixels grayscale images, and a GWT GAP8 PULP System-on-Chip (SoC) [122] extending the basic computational capabilities, i.e., state estimation and low-level control, offered by the STM32 microcontroller available on the nano-drone. We employ the GAP8 SoC to boost the execution of NNs, which require integer quantization to exploit its 8-core general-purpose cluster due to the lack of floating point support. The GAP8 SoC has two levels of on-chip memory: 64kB low-latency L1 memory and 512kB L2 memory. Additionally, the drone features on its main body four controllable LEDs, which we exploit to define the pretext task.

We consider a scenario in which two identical Crazyflie drones fly in the environment: one drone takes the observer role, acquiring camera frames in which the other drone (target) is visible.

8.2.2 Datasets

Our experimental validation is based on Nano2nano[3], a dataset collected in a 10 × 10m lab equipped with a motion-tracking system and consisting of 72 different sequences. For each sequence (average length of 210 seconds and 830 frames), the target Crazyflie flies a pseudo-random trajectory with the four controllable LEDs switched either on or off; meanwhile, the observer continuously moves to increase the variability of represented backgrounds. The trajectory is computed so as to keep the target in the camera view and cover the image space as uniformly as possible, with distances ranging from 0.2 meters to 2 meters. Each frame is labeled with the target pose relative to the observer's camera, its position in the image, and the state of the LEDs. Half of the 72 sequences are used as the testing set $\mathcal{Q}$ (30k samples). Data from remaining 36 sequences is partitioned into the labeled training set $\mathcal{T}_\ell$ (1k samples) and the unlabeled training set $\mathcal{T}_u$ (29k samples).

For training one approach described in Section 8.2.3, we employ the synthetic training dataset proposed in [92, Section IV], named $\mathcal{T}_s$, consisting of 800 random-background images depicting the nano-drone placed at a random pose. Images are converted to grayscale and padded with a solid random gray value to match size and channels of our data. Additional generalization experiments are reported in Section 8.3.4 and shown in the supplementary video; these experiments use data recorded in different rooms, without ground truth for the drone location.

[3] https://github.com/idsia-robotics/drone2drone_dataset

8.2.3 Alternative Strategies

We assess the validity of our approach, named LED state prediction Pretext (LED-P), against various alternatives. First, we consider a naive model (DUMMY) that always predicts the mean position on the labeled training set $\mathcal{T}_\ell$. The Baseline (BAS) strategy involves training using only $\mathcal{L}_{\text{task}}$ (achieved with $\lambda = 0$) on the labeled training set $\mathcal{T}_\ell$. The UB strategy is used to estimate the maximum achievable performance. It minimizes only $\mathcal{L}_{\text{task}}$, assuming to have access to ground truth position labels for both $\mathcal{T}_\ell$ and $\mathcal{T}_u$, representing a fully-supervised scenario where ground truth is cheap and abundant.

We also consider alternative strategies: Autoencoding Pretext (AE-P), Contrastive Language-Image Pre-training (CLIP) and Efficient Deep Neural Networks (EDNN).

Undercomplete autoencoders are a frequently-adopted strategy for taking advantage of unlabeled data, defining an image-reconstruction pretext task [80]. The intuition is that by learning to compress and decompress an image, autoencoders learn a high-level representation that can be useful to solve the end task. In AE-P, we train an autoencoder on $\mathcal{T}_\ell \cup \mathcal{T}_u$ by minimizing the MS-SSIM [188] between input and reconstructed images; then, an additional NN head learns the localization task using $\mathcal{L}_{\text{task}}$ on $\mathcal{T}_\ell$, taking as input features computed by the autoencoder's bottleneck.

CLIP is a powerful bi-modal feature extractor trained to minimize the distance of embeddings between an image and its caption [141]. The learned image encoder is shown to outperform supervised models in many zero- and few-shot tasks. In CLIP, we take the features extracted from the pre-trained image encoder and pass them to a NN head trained for the localization task using $\mathcal{L}_{\text{task}}$ on $\mathcal{T}_\ell$.

In EDNN, we consider supervised pre-training on synthetic images, as described in [92], to cheaply generate labeled data. The strategy consists first in training using the synthetic dataset $\mathcal{T}_s$ on the localization task with a focal loss [95], and then fine-tune the NN parameters using $\mathcal{L}_{\text{task}}$ on $\mathcal{T}_\ell$.

8.2.4 Network Architectures and Training

BAS, UB, EDNN and LED-P share the same tiny FCN [97] architecture consisting of nine convolution blocks, in order: two blocks with 8 channels, 2× max-pooling, three with 16, 2× max-pooling, three with 32, 2× max-pooling and one with 2 channels as the output, totalling 22.1k trainable parameters. Convolution blocks consist of a convolution layer, batch normalization and ReLU activation. The model's input is a grayscale image of 320×320 pixels, normalized between zero and one. The model produces two maps of 40×40 cells, each cell with a 45×45 pixels receptive field, as illustrated in (e, g) of Figure 8.2. Cells of the first map denote drone presence in the corresponding area of the input image[4], while cells of the second denote whether the drone has its LEDs on (1.0) or off (0.0); cells with no visible drone are expected to have a value close to 0.5.

AE-P uses an encoder with four convolution blocks with 4, 8, 16 and 32 channels, interleaved by 2× max-pooling layers, and terminating with the Fully-Connected (FC) bottleneck; in our experiments we considered bottlenecks of 512 and 1024 neurons. The decoder uses four convolution blocks with channels symmetrical to the encoder and interleaved by 2× bilinear upsampling. We attach to the bottleneck a convolution head responsible for localizing the drone,

[4] Map cells are independent one another and assume values in the range from 0 (no drone) to 1 (drone present in given cell).

consisting of two convolution blocks with 32 and 1 channels and interleaved by 2× bilinear upsampling.

CLIP uses the pre-trained image encoder of the homonymous model [141] as a feature extractor; specifically, we adopt the variant using the vision transformer ViT-B/32 [50] producing 512 features. We follow a similar approach to what is presented in [141, Section 3] but keeping CLIP parameters frozen and replacing the logistic regression with a FC head. A FC head uses two blocks composed of a FC layer, batch normalization and ReLU activation. We conducted many trials to find performing architectures that use the 512 CLIP features; we report here the best two: the top performer with 16 neurons and the second with 512, followed by the output block of 400 neurons reshaped into a 20×20 grid.

Finally, we compare LED-P with Frontnet [20], an approach that directly regresses the drone's coordinates.

All NNs are trained using Adam [83] as the optimizer, running for a total of 200 epochs. We adopt a scheduler that divides the learning rate by a factor of 5 every 50 epochs, starting with a learning rate $\eta_{\text{start}} = 1e^{-2}$ and reaching a final learning rate $\eta_{\text{final}} = 8e^{-5}$. In each epoch we randomly draw mini-batches of 64 examples from the two joined training sets and minimize the loss function described in Section 8.1.2. Specifically, we minimize $\mathcal{L}_{\text{pretext}}$ (setting $\mathcal{L}_{\text{task}}$ to 0) for examples taken from $\mathcal{T}_u$ since there are no known labels, and the complete loss $\mathcal{L}$ when fed samples from $\mathcal{T}_\ell$.

Additionally, to increase the variability of the drone's visual appearance, we apply the following augmentations: horizontal flip (50% probability), random rotation (uniform ±9°), random translation (uniform ±32 pixels), and apply multi-frequency simplex noise.

8.2.5 From Grid Map to Robot Position

We consider two approaches to recover $\hat{\boldsymbol{o}}$ from the model's predicted map $\hat{\boldsymbol{q}}$ named argmax and barycenter: the argmax approach selects the coordinate of the cell of $\hat{\boldsymbol{q}}$ whose value is largest; barycenter computes the expected drone position by averaging the coordinates of each map cell weighted by the corresponding probability of depicting a drone [56]. The probability of each map cell is obtained by normalizing $\hat{\boldsymbol{q}}$ such that its sum equals one.

As shown in Figure 4, barycenter returns more conservative estimates, biased towards the mean of the dataset. In contrast, argmax yields unbiased results at the expense of larger errors for frames with no detected drone. Banding artifacts are present with argmax since it cannot represent positions inbetween cells of the 40×40 map, i.e., it discretizes the input image coordinates into 8-pixel-wide bins. In the following experiments we use the argmax approach.

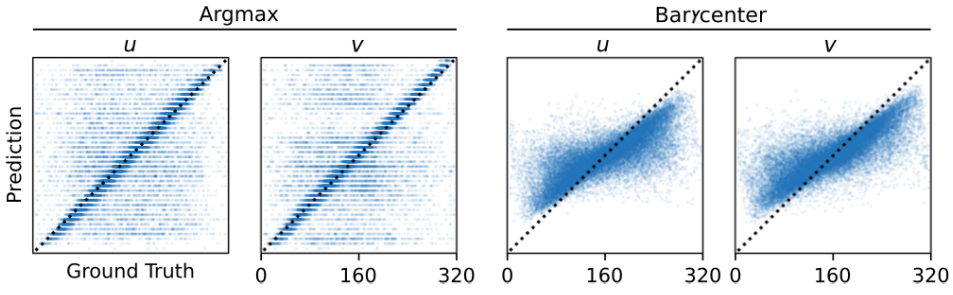


Figure 8.3. LED-P model predictions on the testing set $\mathcal{Q}$ with argmax and barycenter approaches for the u and v components of the drone's position.

8.2.6 Evaluation Metrics

We compare models on different metrics computed on the testing set $\mathcal{Q}$. The Pearson correlation coefficient ρ, computed separately for the horizontal u and vertical v components of $\boldsymbol{o}$; it measures the linear correlation between predicted and ground truth values. We also compute the model's error distribution using the euclidean distance between $\boldsymbol{o}$ and $\hat{\boldsymbol{o}}$. From this distribution, we derive the median value in pixels $\widetilde{\mathrm{D}}$ and a precision score P_k. We chose median instead of mean for being a robust central tendency estimate for skewed distributions. P_k is the fraction of samples whose position error is lower than k pixels, considering predictions with a distance smaller than the threshold k as correct, similarly to ADD for 6 Degrees of Freedom pose estimation [194]. Additionally, we report P_k^+, defined as the relative improvement of LED-P models with respect to the corresponding baseline BAS, such that BAS represents 0%, and UB 100%. In our experiments we consider $k = 30$ pixels, i.e., approximately 10% of the edge length of images.

Even though LED state prediction is not our end task, we also report the Area Under the Receiver Operating Characteristic Curve (AUC) for the LED classification output: it measures how well a model distinguishes the two classes at various thresholds, i.e., telling between a drone with LEDs off or on. A random classifier achieves an AUC score of 50%, while an ideal classifier achieves a score of 100%.

8.3 Experimental Results

This section reports the performance of our proposed strategy (Section 8.3.1), how performance changes as a function of λ and of the amount of labeled examples (Section 8.3.2), a comparison with alternative approaches (Section 8.3.3), generalization ability of our proposed strategy (Section 8.3.4) and a deployed in-field experiment on nano-drones (Section 8.3.5).

8.3.1 LED State Prediction Improves Performance

In Table 8.1, first and last panels, we report the performance of our LED-P strategy against a dummy model (DUMMY), baseline (BAS) and an upperbound (UB). We observe that LED-P-100, trained leveraging unlabeled images, performs moderately better than BAS-100 across all evaluation metrics. The P_{30} metric indicates that 86.9% of predictions fall within 30 pixels from the respective ground truth position, with a median error of only 8 pixels out of a 320×320 pixels image. We also computed P_{30} scores on two subsets of $\mathcal{Q}$ containing examples with LEDs off and on, on which our LED-P-100 model scores 83.3% and 91.2% respectively, showing more difficulties in localizing drones with LEDs off. On the same metric, we report LED-P-100 to score higher than BAS-100 with a p-value of 0.029, computed with the non-parametric one-sided Mann-Whitney U test.

8.3.2 Impact of λ and Amount of Labeled Examples

In Figure 8.4, we inspect the LED-P-30 ($\lambda = 0.001$) prediction against BAS-30 on samples taken from $\mathcal{Q}$, where it scores 30.9% in P_{30}^+, a considerable improvement over the respective baseline. Our model demonstrates an overall good performance when the target drone flies at or below the camera height and, to a lesser extent, when the target flies higher – which hinders the visibility of the LEDs. The model has a harder time localizing the drone when its LEDs are

off. Regarding the AUC for the LED state prediction, our model scores 74% despite these two challenging scenarios being frequently present in $\mathcal{Q}$.

Failure in detecting the drone causes the model to predict the position of similar looking areas of the image. This situation happens less frequently when LEDs are on since their presence improves the drone's visibility.

In Figure 8.5, we show the P^+_{30} relative improvement score for BAS and LED-P strategies as the amount of labeled training examples and the weight of the loss λ vary. LED-P outperforms BAS when training on as few as 100 labeled examples (10% of $\mathcal{T}_\ell$). The optimal value of λ for our loss is 0.001 for 100 and 300 labeled examples (10% and 30% of $\mathcal{T}_\ell$ respectively), and 0.0005 for the full labeled dataset.

Inspecting the two loss term values, we note that $\mathcal{L}_{\text{task}}$ is in the order of magnitude of 10^{-4} and $\mathcal{L}_{\text{pretext}}$ in 10^{-1}. The optimal λ values are scaling the loss terms to be in the same order of magnitude, striking a balance between the two.

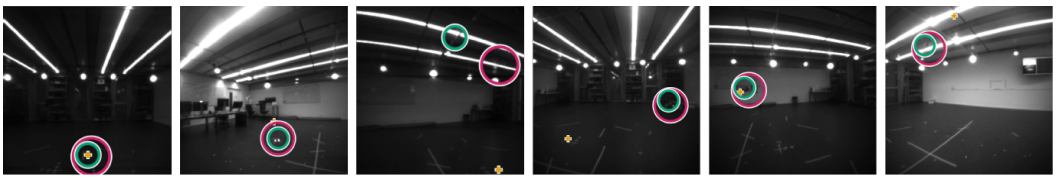


Figure 8.4. LED-P-30 with $\lambda = 0.001$ (small green circle), BAS-30 (yellow cross) predictions and ground truth (large magenta circle) on frames taken from $\mathcal{Q}$ with the drone's LEDs on (first three) and off (last three).

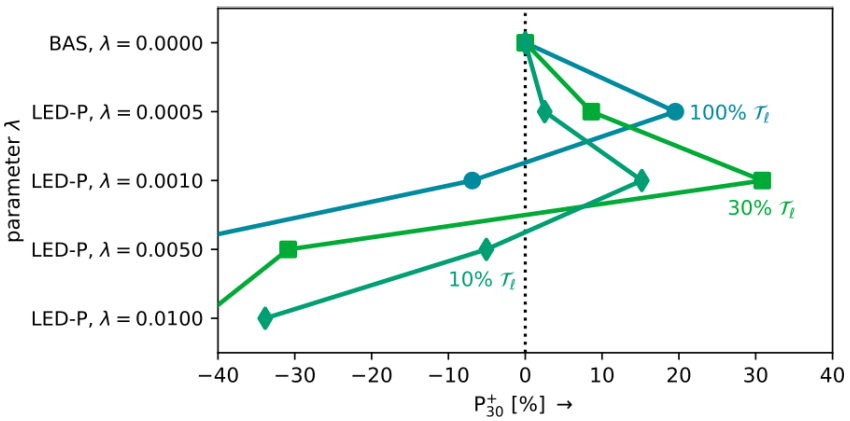


Figure 8.5. P^+_{30} score for BAS ($\lambda = 1$) and LED-P ($\lambda < 1$) strategies as the amount of labeled training examples $\mathcal{T}_\ell$ and the weight of the loss λ vary.

8.3.3 Alternative Training Strategies

We investigate strategies using a different architecture, model pre-training, or different pretext tasks, described in Section 8.2.3, and whose performance metrics are reported in Table 8.1, fourth panel. EDNN scores 10.2 pixels in $\widetilde{\text{D}}$ demonstrating some degree of accuracy; however,

it scores lower than LED-P-100 and BAS-100 that use the same amount of labeled data. This result suggests that task similarity is less influential than dataset relevance, i.e., training on the same task with data vastly different (in appearance) from testing achieves lower scores than solving a different task on similar data, e.g., our LED state prediction pretext task.

Frontnet achieves 31 pixels in $\widetilde{D}$ despite having been trained similarly to BAS-100, which achieves only 8.2 pixels. The increase in error demonstrated by Frontnet indicates that using a FCN model producing a map-based representation leads to a better performance than direct regression.

AE-P successfully learns to reconstruct input images using the bottleneck features. However, the model focuses on large-scale aspects of the environment, such as floors, walls and fixtures, distinctive of background elements and disregards high frequency elements such as the drone. This tendency results in a feature space, regardless of bottleneck size, that is *not informative* of the drone's position, rendering this pretext task inadequate for localizing small object. Even in CLIP's case, we note how the provided features do not translate in good localization performance. We speculate that CLIP's features are inadequate for very specialized vision tasks such as nano-drone localization [141, Section 3.1.5].

AE-P and CLIP highlight the importance of choosing the right kind of pretext [199], which promotes the recognition of patterns similar to those required to solve the end task.

8.3.4 Generalization Ability

In Figure 8.6, we show the prediction of LED-P-30 ($\lambda = 0.001$) on images collected in another lab environment and featuring multiple drones [110]. For this scenario, we modified the argmax approach by thresholding the localization map $\hat{I}$ with its 95-percentile, extracting the maximum value of each connected component, discarding components whose maximum is below 0.2 and returning their centroid as the drone locations. For the most part, our model correctly localizes the drones despite the motion blur and defocus, with all examples featuring at least two correct predictions. Failed detections frequently occur on the edge of the field of view, where the vignetting effect is strong.

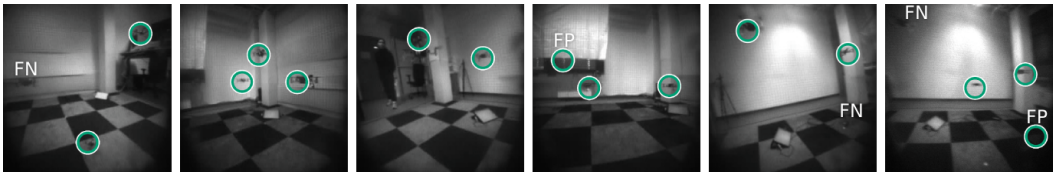


Figure 8.6. Generalization and multi-drone localization examples using LED-P-30 ($\lambda = 0.001$) on images from a different dataset and environment [110]. Errors are visually marked as false positives (FP) or false negatives (FN).

8.3.5 In-Field Experiment

The LED-P-30 ($\lambda = 0.001$) and BAS-30 models are deployed aboard the observer nano-drone, using the academic NEMO/DORY deployment framework[5]. The former tool provides post-training quantization-aware fine-tuning, to convert deep learning models from floating-point to

[5] https://github.com/pulp-platform/nemo

integer arithmetic, needed due to the absence of floating point units on the GAP8 SoC. DORY, instead, produces a template-based C implementation, which takes care of data movements across the memory hierarchy of the GAP8 SoC. This stage is fundamental in achieving fast inference as sub-optimal data tiling/transfers might lead to poor performances, waiting to refill L1 and L2 memories. Our optimized NN pipeline achieves an in-field inference rate of 21 frames per second.

In the observer drone, the model output is used as feedback to a visual servoing controller, designed to keep the target drone in the center of the image. The controller moves the observer drone on a vertical plane, orthogonal to its camera axis, while keeping a constant yaw. Without loss of generality, we consider the motion of the observer drone to take place on the yz world plane, with $x = 0$. In the experiment, the target drone follows a predefined, scripted trajectory. The ideal trajectory for the observer drone is the same as the target, projected on $x = 0$ vertical world plane.

Figure 8.7 reports the y and z components of the measured trajectory of the observer drone, controlled using position estimates from LED-P-30, compared to the ideal one. We observe that the drone follows very closely the ideal trajectory. The same experiment run using BAS-30 yields worse position tracking: the mean and standard deviation σ of the absolute position error on the yz plane is 4.2 cm ($\sigma = 4.0$ cm) for LED-P-30, and 11.9 cm ($\sigma = 8.3$ cm) for BAS-30.

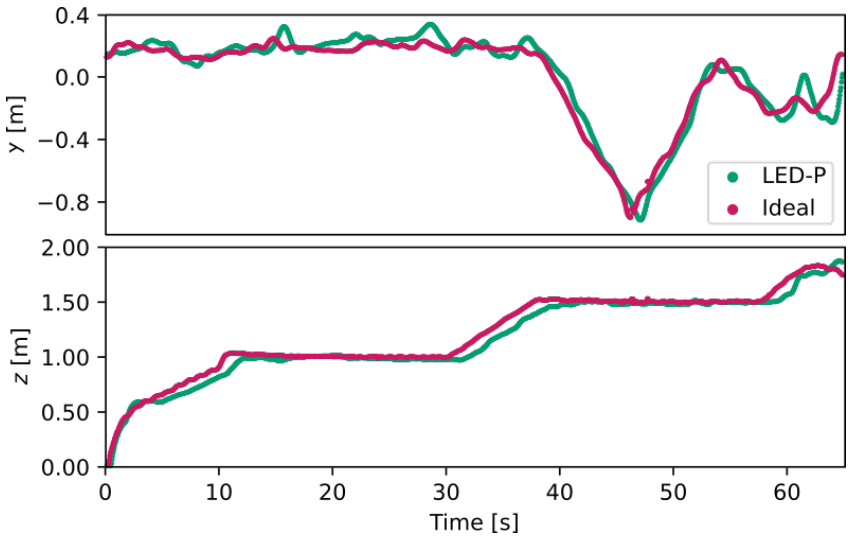


Figure 8.7. In-field experiment: measured vs ideal trajectory of the observer drone when using LED-P-30 ($\lambda = 0.001$) for estimating the target position.

8.4 Discussion

We propose LED state estimation as a self-supervised pretext task applied to the end task of visually localizing robots from small labeled datasets. The pretext task is optimized on large, cheaply-collected datasets that only have ground truth for the LED state of the observed robot. The approach is instantiated on localizing nano-quadrotors in low-resolution images, observing improved localization accuracy compared to baselines and alternative techniques for self-

supervision. In-field experiments used a 27-g Crazyflie nano-drone to track the position of a peer drone; the proposed approach reduces mean position-tracking error from 11.9 to 4.2 cm.

The proposed approach represents a small step towards more powerful pretext tasks for perception. We discuss future research directions in more detail in Chapter 9. Here, instead, we highlight some extensions to the LED-based pretext task, where we will consider the different robot LEDs to be separate and independent. First, we consider the pretext task of predicting the state of each of the four LEDs. This introduces additional complexity stemming from the robot's orientation, as some LEDs will naturally be occluded, and the visible ones require the identification of their respective assignment (also based on the orientation of the robot). By learning this pretext task, the model would learn useful features for both position and orientation of the other robot. Second, by using an asymmetric LED pattern, such as three out of four turned on, we provide a strong cue for the robot's orientation. Using such an LED pattern will simplify the prediction the robot's orientation and may also be used at inference time.

Table 8.1. Comparison of models, five replicas per row. Pearson correlation coefficient ρ_u and ρ_v, precision P_{30} and median of the error $\tilde{D}$ computed on the testing set $\mathcal{Q}$.

Model	Training set for task: End	Pretext	λ	ρ_u [%] ↑	ρ_v [%] ↑	P_{30} [%] ↑	P_{30}^{+} [%] ↑	$\tilde{D}$ [px] ↓
DUMMY	–	–	–	–	–	8.0	–	79.0
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0100	55.2	59.0	63.4	-250.6	13.2
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0050	74.3	78.0	80.9	-49.4	8.8
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0010	79.6	81.7	84.6	-6.9	8.3
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0005	**81.3**	**83.6**	**86.9**	**19.5**	**8.0**
BAS-100	$\mathcal{T}_\ell$	–	0.0000	79.0	82.7	85.2	0.0	8.2
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0100	50.1	55.5	57.5	-43.0	14.9
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0050	51.0	57.3	60.4	-30.8	14.3
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0010	**68.5**	**71.4**	**76.2**	**30.9**	**9.9**
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0005	61.7	66.3	70.5	8.6	11.1
BAS-30	30%$\mathcal{T}_\ell$	–	0.0000	59.0	66.2	68.3	0.0	10.9
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0100	18.3	28.7	25.1	-33.9	94.1
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0050	32.3	42.2	39.9	-5.1	61.0
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0010	**42.1**	**48.0**	**50.3**	**15.2**	**33.6**
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0005	36.1	44.7	43.8	2.5	53.1
BAS-10	10%$\mathcal{T}_\ell$	–	0.0000	34.7	41.6	42.5	0.0	57.8
AE-P-512	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	–	0.5	1.1	3.7	–	137.9
AE-P-1024	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	–	-0.9	-2.2	3.8	–	130.1
CLIP-16 [141]	$\mathcal{T}_\ell$	–	–	2.0	8.9	6.5	–	102.7
CLIP-512 [141]	$\mathcal{T}_\ell$	–	–	1.4	7.8	5.7	–	109.1
Frontnet [20]	$\mathcal{T}_\ell$	–	–	**69.5**	**74.1**	48.7	–	31.0
EDNN [92]	$\mathcal{T}_\ell$	$\mathcal{T}_s$	–	64.7	68.5	**73.3**	–	**10.2**
UB	$\mathcal{T}_\ell \cup \mathcal{T}_u^*$	–	0.0000	91.7	89.3	93.9	100.0	6.8

Chapter 9

Conclusions

Self-supervised robot learning is a promising approach to overcome the lack of labeled data in real robotics applications. By leveraging a robot's autonomous capabilities, we enable the collection of training data without relying on pre-existing datasets or expensive data collection processes. In turn this makes many real-world robotics applications more practical and approachable by practitioners. Specifically, we focus on spatial perception problems, which require understanding spatial information, such as the relationship between the robot and objects, people, or other robots in the environment. Visual data, such as camera frames, lidar, or laser readings, serve as input for deep learning models that predict spatial information. Training deep learning models notoriously requires large amounts of training data supplied by the robot's own data collection. With cheap, robot-collected data, we can supervise vision models directly with labels generated from sensors' readings or by using self-supervised pretext tasks to boost performance and reduce the reliance on labeled data.

Throughout my doctoral studies, we have made valuable contributions to the state of the art from two different perspectives: first, we investigated different types of sensors and actuators used during data collection, from common ones, such as proximity sensors and odometry, to less prominent yet powerful ones, such as microphones and LEDs. Much attention has been put into practical considerations related to cost and availability of the chosen hardware components. In particular, we centered many of our articles on the robot's odometry, an ubiquitous state estimation process that generates useful information. We also proposed pretext tasks based on controllable LEDs, being fitted in most modern robot platforms, and on microphones which come in miniaturized, very inexpensive and lightweight boards, if not already present in the platform. One of the advantages of our approaches is that they leverage existing and inexpensive off-the-shelf hardware or robot-fitted sensors instead of relying on expensive and non-portable external systems. Data collection can be performed at the same speed as the robot itself, allowing for efficient and fairly quick data acquisition. This property is especially beneficial where real-time or near-real-time data processing is required, such as for online learning approaches. Furthermore, data collection can be carried out in various environments, exploring and gathering data from different real-world scenarios. As a result, models trained with such data demonstrate good generalization and perform effectively in unseen environments.

Second, we addressed some issues associated with self-supervised robot learning approaches. Specifically, in some situations a significant portion of the robot's data is only partially labeled or contains labels only for a limited number of training samples, e.g., when labels depend on

the exploration of the robot. We proposed a simple yet effective consistency loss for leveraging these partial labels for supervising a model. Another problem is the reliance on sensors' readings to derive training labels, which can be imprecise and have detrimental effects during training. The presence of noise in sensors' measurements can introduce errors and impact the training results, leading to less performing models. To tackle the issue, we proposed an uncertainty-aware approach that improves the performance of self-supervised models in the presence of noisy labels.

In this dissertation, we have made headway on the goal of alleviating the burden of large-scale data collection required for training deep perception models by enabling robots to autonomously collect their own data and employing strategies to minimize the dependency on labels. Now, we outline some of the future research directions towards this goal and to further improve perception models.

Multiple modalities to learn better features In Chapters 3 and 7, we have investigated the relationship between different perceptions of the same environment. Specifically, we trained a model to predict one modality from another in a cross-modal fashion. This approach is relevant in the field of self-supervised robot perception, where data from various modalities, such as geometric, visual, auditory, kinesthetic, and tactile, can be easily and cheaply collected using different sensors. It is well-known that fusing multiple modalities to solve a task often leads to better results than using a single one [73]. Many robotics approaches have adopted this strategy to improve their perception capabilities. However, what has not been extensively explored is the prediction of multiple modalities from a subset thereof. By successfully predicting multiple modalities, for example, from a single camera frame, a model demonstrates to have learned a general, high-level understanding of its environment. This means that the model's hidden features are highly informative and can be leveraged to solve perception tasks in a more effective way.

More powerful pretext tasks to learn better features In Chapters 7 and 8, we have proposed two different pretext tasks for robot pose estimation. These pretext tasks are successful because they take advantage of a known relationship between the end task (robot pose estimation) and the pretext task. For example, the sound of a drone perceived by a microphone is inherently related to its location in space. Despite their success, very few perception approaches explored the use of pretext tasks [116, 142]. This could be attributed to the lack of theoretical understanding and motivation behind the success of pretext tasks: this tasks originate from our understanding of the fundamental laws that govern our world without much theoretical scaffolding to support their effectiveness. Only recently, some work is shedding light on the intuition behind pretext tasks by applying the principles of information theory [160]. Therefore, more investigation is required to gain a deeper understanding of pretext tasks, their effectiveness, and the underlying information they capture. Further improvements require designing new pretext tasks that capture more relevant and informative features for the end task; this will provide insights into which pretext tasks are successful and how their success is related to this shared information.

Onboard and online learning to lower costs So far, the proposed approaches are composed of different sequential phases, starting from data collection, than dataset generation, model training and finally deployment. This design is in line with the most common robot learning paradigms. By ensuring that all available information is provided to the dataset generation

phase, the resulting dataset is representative of real-world scenarios and is varied enough for the robot to generalize to different environments. Once the dataset is complete, models are trained on dedicated deep learning machines, and only at the end they are deployed onto the robot. A prominent robot learning paradigm is the online learning one, in which a robot collects some data, trains on it onboard, and repeats the process iteratively. Online robot learning is desirable and has several advantages: robots can quickly adapt to changes in the data distribution, making them more robust in scenarios where the environment may change over time, e.g., by entering a dark tunnel; it enables continuous improvements since the model learns iteratively, resulting in better performance and doing so during deployment; due to its online nature, this paradigm eliminates the need to store data already used for training and to train a model on the robot itself, avoiding the need for expensive, dedicated deep learning machines.

Appendix A

Publications

After an extension of our master thesis work, we presented our initial results on the use of odometry and proximity sensors for self-supervision [112] at the demonstration track of the AAAI 2019 conference. With the addition of new experiments on a different robot platform on the task of predicting terrain features, highlighting the generality of the approach, described in Chapter 3, we published our work [113] in the IEEE RA-L journal and presented it at the ICRA 2019 conference. In the following work, we improved over [113] with the introduction of an additional unsupervised consistency loss, focused on enforcing consistency over geometrically-related model predictions [114], described in Chapter 4; this work is published in the IEEE RA-L journal. We also published our work on integrating self-supervised approaches with uncertainty awareness [115], described in Chapter 5, in the IEEE RA-L journal and later presented it at the IROS 2021 conference. Our work on the use of sound prediction as a pretext task [116], described in Chapter 7, is published in the IEEE RA-L journal and was presented at the ICRA 2022 conference. Finally, our latest work on using LEDs state estimation as a pretext task, described in Chapter 8, is currently under review at the IEEE RA-L journal.

In addition, we collaborated with colleagues on three projects: path planning with local motion estimates, learning a visual feature extractor to close the control loop, and studying the impact of feeding state estimates as additional information to predictive models. The former resulted in the article [67], published in the IEEE RA-L journal and presented at the ICRA 2020 conference. The second project resulted in the article [127], described in Chapter 6, presented at the IROS 2022 conference, and published in the conference proceedings. Meanwhile, the latter is currently under review. Outside the area of robotics, we collaborated with another group of our research center on an applied deep learning project. Specifically, the theme was domain adaptation with neural networks applied to the challenging problem of estimating the transmission quality in optical fiber networks. The resulting article [150] is published in the JOCN journal.

In October 2022, I started a five-month internship at Magic Leap Switzerland GmbH on the topic of self-supervised object pose estimation, experimenting ways to learn a pose estimator with little-to-none ground truth, involving the use of NeRF [107] to generate images for supervision.

Since the beginning of the Ph.D. studies, we published a total of 1 demonstration, 1 conference and 6 journal articles; we are also waiting for 2 articles, which are currently under review. All articles except [127] and [150] are open access, and feature links to dedicated pages on the

research group website, giving full access to supplementary material such as videos, developed code, and datasets used to validate the proposed approaches.

References

[112] Mirko Nava, Jérôme Guzzi, Ricardo Omar Chavez-Garcia, Luca Maria Gambardella, and Alessandro Giusti. Demo: Learning to perceive long-range obstacles using self-supervision from short-range sensors. *AAAI Conference on Artificial Intelligence*, 33: 9867–9868, 2019.

[113] Mirko Nava, Jérôme Guzzi, Ricardo Omar Chavez-Garcia, Luca Maria Gambardella, and Alessandro Giusti. Learning long-range perception using self-supervision from short-range sensors and odometry. *IEEE Robotics and Automation Letters*, 4: 1279–1286, 2019.

[114] Mirko Nava, Luca Maria Gambardella, and Alessandro Giusti. State-consistency loss for learning spatial perception tasks from partial labels. *IEEE Robotics and Automation Letters*, 6: 1112–1119, 2021.

[115] Mirko Nava, Antonio Paolillo, Jérôme Guzzi, Luca Maria Gambardella, and Alessandro Giusti. Uncertainty-aware self-supervised learning of spatial perception tasks. *IEEE Robotics and Automation Letters*, 6: 6693–6700, 2021.

[116] Mirko Nava, Antonio Paolillo, Jérôme Guzzi, Luca Maria Gambardella, and Alessandro Giusti. Learning visual localization of a quadrotor using its noise as self-supervision. *IEEE Robotics and Automation Letters*, 7: 2218–2225, 2022.

[67] Jérôme Guzzi, Ricardo Omar Chavez-Garcia, Mirko Nava, Luca Maria Gambardella, and Alessandro Giusti. Path planning with local motion estimations. *IEEE Robotics and Automation Letters*, 5: 2586–2593, 2020.

[127] Antonio Paolillo, Mirko Nava, Dario Piga and Alessandro Giusti. Visual Servoing with Geometrically Interpretable Neural Perception. *IEEE/RSJ International Conference on Intelligent Robots and Systems*: 5300–5306, 2022.

[150] Cristina Rottondi, Riccardo di Marino, Mirko Nava, Alessandro Giusti, and Andrea Bianco. On the benefits of domain adaptation techniques for quality of transmission estimation in optical networks. *Journal of Optical Communications and Networking*, 13: 34–43, 2021.

Bibliography

[1] E Ackerman. Turtlebot inventors tell us everything about the robot. *IEEE Spectrum*, 26: 1–10, 2013.

[2] National Highway Traffic Safety Administration. *Standing General Order on Crash Reporting for Level 2 Advanced Driver Assistance Systems*. United States Department of Transportation, DOT HS 813 325, June 2022.

[3] D J Agravante, G Claudio, F Spindler, and F Chaumette. Visual servoing in an optimization framework for the whole-body control of humanoid robots. *IEEE Robotics and Automation Letters*, 2(2):608–615, 2017.

[4] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *IEEE/CVF International Conference on Computer Vision*, pages 37–45, 2015.

[5] Guillaume Allibert, Estelle Courtial, and François Chaumette. Predictive control for constrained image-based visual servoing. *IEEE Transactions on Robotics*, 26(5):933–939, 2010.

[6] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *Advances in Neural Information Processing Systems*, 33:9758–9770, 2020.

[7] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *IEEE/CVF International Conference on Computer Vision*, pages 609–617, 2017.

[8] Relja Arandjelovic and Andrew Zisserman. Objects that sound. In *European Conference on Computer Vision*, pages 435–451, 2018.

[9] Efron B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7 (1):1–26, 1979.

[10] Manikandan Bakthavatchalam, Omar Tahri, and François Chaumette. A direct dense visual servoing approach using photometric moments. *IEEE Transactions on Robotics*, 34 (5):1226–1239, 2018.

[11] Dan Barnes, Will Maddern, and Ingmar Posner. Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy. In *IEEE International Conference on Robotics and Automation*, pages 203–210, 2017.

[12] Meysam Basiri, Felix Schill, Dario Floreano, and Pedro U Lima. Audio-based localization for swarms of micro air vehicles. In *IEEE International Conference on Robotics and Automation*, pages 4729–4734, 2014.

[13] Quentin Bateux and Eric Marchand. Histograms-based visual servoing. *IEEE Robotics and Automation Letters*, 2(1):80–87, 2017.

[14] Quentin Bateux, Eric Marchand, Jürgen Leitner, François Chaumette, and Peter Corke. Training deep neural networks for visual servoing. In *IEEE International Conference on Robotics and Automation*, pages 3307–3314, 2018.

[15] Mohammed Abdessamad Bekhti, Yuichi Kobayashi, and Kazuki Matsumura. Terrain traversability analysis using multi-sensor data correlation by a mobile robot. In *IEEE/SICE International Symposium on System Integration*, pages 615–620, 2014.

[16] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8):1798–1828, 2013.

[17] Lars Berscheid, Thomas Rühr, and Torsten Kröger. Improving data efficiency of self-supervised learning for robotic grasping. In *IEEE International Conference on Robotics and Automation*, pages 2125–2131, 2019.

[18] Lars Berscheid, Pascal Meißner, and Torsten Kröger. Self-supervised learning for precise pick-and-place without object model. *IEEE Robotics and Automation Letters*, 5(3):4828–4835, 2020.

[19] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *PMLR International Conference on Machine Learning*, pages 1613–1622, 2015.

[20] Stefano Bonato, Stefano Carlo Lambertenghi, Elia Cereda, Alessandro Giusti, and Daniele Palossi. Ultra-low power deep learning-based monocular relative localization onboard nano-quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 3411–3417, 2023.

[21] Christopher A Brooks and Karl Iagnemma. Self-supervised terrain classification for planetary surface exploration rovers. *Wiley Online Library Journal of Field Robotics*, 29(3): 445–468, 2012.

[22] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Elsevier Neural networks*, 106: 249–259, 2018.

[23] M Akmal Butt and Petros Maragos. Optimum design of chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484, 1998.

[24] Adrian Carrio, Sai Vemprala, Andres Ripoll, Srikanth Saripalli, and Pascual Campoy. Drone detection using depth maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1034–1037, 2018.

[25] Adrian Carrio, Jesus Tordesillas, Sai Vemprala, Srikanth Saripalli, Pascual Campoy, and Jonathan P How. Onboard detection and localization of drones using depth maps. *IEEE Access*, 8:30480–30490, 2020.

[26] Francesco Castelli, Stefano Michieletto, Stefano Ghidoni, and Enrico Pagello. A machine learning-based visual servoing approach for fast robot control in industrial setting. *International Journal of Advanced Robotic Systems*, 14(6), 2017.

[27] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, London, England, 2006.

[28] F Chaumette and S Hutchinson. Visual servo control. part i: Basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, 2006.

[29] F Chaumette and S Hutchinson. Visual servo control. part ii: Advanced approaches. *IEEE Robotics & Automation Magazine*, 14(1):109–118, 2007.

[30] R Omar Chavez-Garcia, Jerome Guzzi, Luca M Gambardella, and Alessandro Giusti. Learning ground traversability from simulations. *IEEE Robotics and Automation Letters*, 3(3):1695–1702, 2018.

[31] G Chesi and Y S Hung. Global path-planning for constrained and optimal visual servoing. *IEEE Transactions on Robotics*, 23(5):1050–1060, 2007.

[32] Maicol Ciani, Stefano Bonato, Rafail Psiakis, Angelo Garofalo, Luca Valente, Suresh Sugumar, Alessandro Giusti, Davide Rossi, and Daniele Palossi. Cyber security aboard micro aerial vehicles: An opentitan-based visual communication use case. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2023.

[33] Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Cambridge University Press Behavioral and brain sciences*, 36(3):181–204, 2013.

[34] Andy Clark. *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford University Press, 2015.

[35] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[36] David Coleman, Ioan Alexandru Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: A moveit! case study. *Journal of Software Engineering for Robotics*, 5(1):3–16, 2014.

[37] Christophe Collewet and Eric Marchand. Photometric visual servoing. *IEEE Transactions on Robotics*, 27(4):828–834, 2011.

[38] Francesco Conti, Daniele Palossi, Andrea Marongiu, Davide Rossi, and Luca Benini. Enabling the heterogeneous accelerator model on ultra-low power microcontroller platforms. In *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1201–1206, 2016.

[39] Mario Coppola, Kimberly N McGuire, Kirk YW Scheper, and Guido CHE de Croon. On-board communication-based relative localization for collision avoidance in micro air vehicle teams. *Springer Autonomous Robots*, 42(8):1787–1805, 2018.

[40] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB®*. Springer, 2017.

[41] Nathan Crombez, El Mustapha Mouaddib, Guillaume Caron, and François Chaumette. Visual servoing with photometric gaussian mixtures as dense features. *IEEE Transactions on Robotics*, 35(1):49–63, 2019.

[42] Kevin Crowston. Amazon mechanical turk: A research tool for organizations and information systems scholars. In *Springer IFIP Advances in Information and Communication Technology - Shaping the Future of ICT Research*, pages 210–221, 2012.

[43] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: Science and Systems*, 2006.

[44] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics*, 36(4):1, 2017.

[45] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *IEEE International Conference on Robotics and Automation*, pages 3665–3671, 2020.

[46] Duarte Dias, Rodrigo Ventura, Pedro Lima, and Alcherio Martinoli. On-board vision-based 3d relative localization system for multiple quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 1181–1187, 2016.

[47] Ernst D Dickmanns. Computer vision and highway automation. *Vehicle System Dynamics*, 31(5-6):325–343, 1999.

[48] Ernst D Dickmanns and Alfred Zapp. Autonomous high speed road vehicle guidance by computer vision. *IFAC Proceedings Volumes*, 20(5):221–226, 1987.

[49] Ernst Dieter Dickmanns and Alfred Zapp. A curvature-based scheme for improving road vehicle guidance by computer vision. In *SPIE Mobile Robots I*, volume 727, pages 161–168, 1987.

[50] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations, Virtual Event*, 2021.

[51] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374, 2014.

[52] Franka Emika. Franka emika website. `https://www.franka.de`, Accessed: 2023.

[53] Samuel Felton, Elisa Fromont, and Eric Marchand. Siame-SE(3): regression in se(3) for end-to-end visual servoing. In *IEEE International Conference on Robotics and Automation*, pages 14454–14460, 2021.

[54] Samuel Felton, Pascal Brault, Elisa Fromont, and Eric Marchand. Visual servoing in autoencoder latent space. *IEEE Robotics and Automation Letters*, 7(2):3234–3241, 2022.

[55] Eric L Ferguson, Stefan B Williams, and Craig T Jin. Sound source localization in a multipath environment using convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2386–2390, 2018.

[56] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *IEEE International Conference on Robotics and Automation*, pages 512–519, 2016.

[57] John H Flavell. *The Developmental Psychology of Jean Piaget*. Van Nostrand Reinhold, 1963.

[58] Open Source Robotics Foundation. Robot Operating System Website. `https://www.ros.org`, Accessed: 2023.

[59] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *PMLR International Conference on Machine Learning*, pages 1050–1059, 2016.

[60] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3948–3955, 2017.

[61] Dhiraj Gandhi, Abhinav Gupta, and Lerrel Pinto. Swoosh! rattle! thump!-actions that sound. In *Robotics: Science and Systems*, 2020.

[62] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613, 2016.

[63] Alessandro Giusti, Jérôme Guzzi, Dan C Ciresan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

[64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[65] Samet Güler, Mohamed Abdelkader, and Jeff S Shamma. Peer-to-peer relative localization of aerial robots with ultrawideband sensors. *IEEE Transactions on Control Systems Technology*, 29(5):1981–1996, 2020.

[66] Jérôme Guzzi, Alessandro Giusti, Gianni A Di Caro, and Luca M Gambardella. Mighty thymio for higher-level robotics education. In *AAAI Conference on Artificial Intelligence*, 2018.

[67] Jérôme Guzzi, R Omar Chavez-Garcia, Mirko Nava, Luca M Gambardella, and Alessandro Giusti. Path planning with local motion estimations. *IEEE Robotics and Automation Letters*, 5(2):2586–2593, 2020.

[68] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Wiley Online Library Journal of Field Robotics*, 26(2):120–144, 2009.

[69] Lei Han and Lu Fang. Flashfusion: Real-time globally consistent dense 3d reconstruction using cpu computing. In *Robotics: Science and Systems*, volume 1, 2018.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[71] Weipeng He, Petr Motlicek, and Jean-Marc Odobez. Adaptation of multiple sound source localization neural networks with weak supervision and domain-adversarial training. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 770–774, 2019.

[72] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object [ose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2930–2939, 2020.

[73] Yu Huang, Chenzhuang Du, Zihui Xue, Xuanyao Chen, Hang Zhao, and Longbo Huang. What makes multi-modal learning better than single (provably). *Advances in Neural Information Processing Systems*, 34:10944–10956, 2021.

[74] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal: A highly mobile and dynamic quadrupedal robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 38–44, 2016.

[75] Tadanobu Inoue, Subhajit Choudhury, Giovanni De Magistris, and Sakyasingha Dasgupta. Transfer learning from synthetic to real images using variational autoencoders for precise position detection. In *IEEE International Conference on Image Processing*, pages 2725–2729, 2018.

[76] Ganesh Iyer, J Krishna Murthy, Gunshi Gupta, Madhava Krishna, and Liam Paull. Geometric consistency for self-supervised end-to-end visual odometry. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, Workshop*, pages 267–275, 2018.

[77] Eric Jang, Coline Devin, Vincent Vanhoucke, and Sergey Levine. Grasp2vec: Learning object representations from self-supervised grasping. In *PMLR Conference on Robot Learning*, pages 99–112, 2018.

[78] Huaizu Jiang, Gustav Larsson, Michael Maire Greg Shakhnarovich, and Erik Learned-Miller. Self-supervised relative depth learning for urban scene understanding. In *European Conference on Computer Vision*, pages 19–35, 2018.

[79] Jun Jin, Laura Petrich, Masood Dehghan, and Martin Jagersand. A geometric perspective on visual imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5194–5200, 2020.

[80] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 4037–4058, 2020.

[81] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1521–1529, 2017.

[82] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, Workshop*, pages 1–10, 2017.

[83] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[84] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154, 2004.

[85] Kishore Reddy Konda and Roland Memisevic. Learning visual odometry with a convolutional network. *International Conference on Computer Vision Theory and Applications, VISAPP*, 2015:486–490, 2015.

[86] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *Advances in Neural Information Processing Systems*, pages 7774–7785, 2018.

[87] Alexandros Kouris and Christos-Savvas Bouganis. Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–9, 2018.

[88] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[89] Alex X Lee, Sergey Levine, and Pieter Abbeel. Learning visual servoing with deep features and fitted Q-iteration. In *International Conference on Learning Representations*, 2017.

[90] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *SAGE Publications The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[91] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[92] Shushuai Li, Christophe De Wagter, and Guido CHE De Croon. Self-supervised monocular multi-robot relative localization with efficient deep neural networks. In *IEEE International Conference on Robotics and Automation*, pages 9689–9695, 2022.

[93] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *European Conference on Computer Vision*, pages 683–698, 2018.

[94] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *IEEE/CVF International Conference on Computer Vision*, pages 7678–7687, 2019.

[95] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE/CVF International Conference on Computer Vision*, pages 2980–2988, 2017.

[96] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37, 2016.

[97] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[98] Andrew Lookingbill, John Rogers, David Lieb, J Curry, and Sebastian Thrun. Reverse optical flow for self-supervised adaptive autonomous robot navigation. *Springer International Journal of Computer Vision*, 74:287–302, 2006.

[99] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[100] Siddharth Mahendran, Haider Ali, and René Vidal. 3d pose regression using convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, Workshop*, pages 2174–2182, 2017.

[101] Daniel Maier, Maren Bennewitz, and Cyrill Stachniss. Self-supervised obstacle detection for humanoid navigation using monocular vision and sparse laser data. In *IEEE International Conference on Robotics and Automation*, pages 1263–1269, 2011.

[102] Dario Mantegazza, JÃ©rÃ´me Guzzi, Luca M. Gambardella, and Alessandro Giusti. Vision-based control of a quadrotor in user proximity: Mediated vs end-to-end learning approaches. In *IEEE International Conference on Robotics and Automation*, pages 6489–6495, 2019.

[103] Éric Marchand and François Chaumette. Feature tracking for visual servoing purposes. *Elsevier Robotics and Autonomous Systems*, 52(1):53–70, 2005.

[104] Éric Marchand, Fabien Spindler, and François Chaumette. Visp for visual servoing: A generic software platform with a wide class of robot control skills. *IEEE Robotics & Automation Magazine*, 12(4):40–52, 2005.

[105] Daniel Maturana and Sebastian Scherer. 3d convolutional neural networks for landing zone detection from lidar. In *IEEE International Conference on Robotics and Automation*, pages 3471–3478, 2015.

[106] Y Mezouar and F Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics*, 18(4):534–549, 2002.

[107] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ACM Communications of the ACM*, 65(1):99–106, 2021.

[108] Enrico Mingo Hoffman and Antonio Paolillo. Exploiting visual servoing and centroidal momentum for whole-body motion control of humanoid robots in absence of contacts and gravity. In *IEEE International Conference on Robotics and Automation*, pages 2979–2985, 2021.

[109] Marwan Qaid Mohammed, Kwek Lee Chung, and Chua Shing Chyi. Review of deep reinforcement learning-based object grasping: Techniques, open challenges, and recommendations. *IEEE Access*, 8:178450–178481, 2020.

[110] Akmaral Moldagalieva and Wolfgang Hönig. A dataset and comparative study for vision-based relative position estimation of multirotor teams flying in close proximity. *arXiv*, page 2303.03898, 2023.

[111] Francesco Mondada, Michael Bonani, Fanny Riedo, Manon Briod, Léa Pereyre, Philippe Rétornaz, and Stéphane Magnenat. Bringing robotics to formal education: The thymio open-source hardware robot. *IEEE Robotics & Automation Magazine*, 24(1):77–85, 2017.

[112] Mirko Nava, Jérôme Guzzi, R Omar Chavez-Garcia, Luca M Gambardella, and Alessandro Giusti. Demo: Learning to perceive long-range obstacles using self-supervision from short-range sensors. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 9867–9868, 2019.

[113] Mirko Nava, Jérôme Guzzi, R Omar Chavez-Garcia, Luca M Gambardella, and Alessandro Giusti. Learning long-range perception using self-supervision from short-range sensors and odometry. *IEEE Robotics and Automation Letters*, 4(2):1279–1286, 2019.

[114] Mirko Nava, Luca Maria Gambardella, and Alessandro Giusti. State-consistency loss for learning spatial perception tasks from partial labels. *IEEE Robotics and Automation Letters*, 6(2):1112–1119, 2021.

[115] Mirko Nava, Antonio Paolillo, Jérôme Guzzi, Luca M Gambardella, and Alessandro Giusti. Uncertainty-aware self-supervised learning of spatial perception tasks. *IEEE Robotics and Automation Letters*, 6(4):6693–6700, 2021.

[116] Mirko Nava, Antonio Paolillo, Jérôme Guzzi, Luca M Gambardella, and Alessandro Giusti. Learning visual localization of a quadrotor using its noise as self-supervision. *IEEE Robotics and Automation Letters*, 7(2):2218–2225, 2022.

[117] Thien-Minh Nguyen, Zhirong Qiu, Thien Hoang Nguyen, Muqing Cao, and Lihua Xie. Distance-based cooperative relative localization for leader-following control of mavs. *IEEE Robotics and Automation Letters*, 4(4):3641–3648, 2019.

[118] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *European Conference on Computer Vision*, pages 631–648, 2018.

[119] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. Ambient sound provides supervision for visual learning. In *European Conference on Computer Vision*, pages 801–816, 2016.

[120] Simone Palazzo, Dario C Guastella, Luciano Cantelli, Paolo Spadaro, Francesco Rundo, Giovanni Muscato, Daniela Giordano, and Concetto Spampinato. Domain adaptation for outdoor robot traversability estimation from rgb data with safety-preserving loss. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 10014–10021, 2020.

[121] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. Ultra low power deep-learning-powered autonomous nano drones. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8357–8371, 2018.

[122] Daniele Palossi, Francesco Conti, and Luca Benini. An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs. In *IEEE International Conference on Distributed Computing in Sensor Systems*, pages 604–611, 2019.

[123] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. A 64-mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal*, 6(5):8357–8371, 2019.

[124] A Paolillo, T S Lembono, and S Calinon. A memory of motion for visual predictive control tasks. In *IEEE International Conference on Robotics and Automation*, pages 9014–9020, 2020.

[125] Antonio Paolillo and Matteo Saveriano. Learning stable dynamical systems for visual servoing. In *IEEE International Conference on Robotics and Automation*, pages 8636–8642, 2022.

[126] Antonio Paolillo, Angela Faragasso, Giuseppe Oriolo, and Marilena Vendittelli. Vision-based maze navigation for humanoid robots. *Autonomous Robots*, 41(2):293–309, 2017.

[127] Antonio Paolillo, Mirko Nava, Dario Piga, and Alessandro Giusti. Visual servoing with geometrically interpretable neural perception. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5300–5306, 2022.

[128] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[129] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[130] Mandela Patrick, Yuki M. Asano, Polina Kuznetsova, Ruth Fong, Jo ao F. Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. In *IEEE/CVF International Conference on Computer Vision*, 2021.

[131] Maxim Pavliv, Fabrizio Schiano, Christopher Reardon, Dario Floreano, and Giuseppe Loianno. Tracking and relative localization of drone swarms with a vision-based headset. *IEEE Robotics and Automation Letters*, 6(2):1455–1462, 2021.

[132] Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. Boa: The bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1–10, 1999.

[133] Valentin Peretroukhin and Jonathan Kelly. Dpc-net: Deep pose correction for visual localization. *IEEE Robotics and Automation Letters*, 3(3):2424–2431, 2017.

[134] Valentin Peretroukhin, Matthew Giamou, W Nicholas Greene, David Rosen, Jonathan Kelly, and Nicholas Roy. A smooth representation of belief over SO(3) for deep rotation learning with uncertainty. In *Robotics: Science and Systems*, 2020.

[135] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *IEEE International Conference on Robotics and Automation*, pages 1527–1533, 2017.

[136] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *IEEE International Conference on Robotics and Automation*, pages 3406–3413, 2016.

[137] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems*, 1, 1988.

[138] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[139] En Yen Puang, Keng Peng Tee, and Wei Jing. KOVIS: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7527–7533, 2020.

[140] Lawrence Rabiner and Ronald Schafer. *Theory and Applications of Digital Speech Processing*. Prentice Hall Press, 2010.

[141] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *PMLR International Conference on Machine Learning*, pages 8748–8763, 2021.

[142] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *PMLR Conference on Robot Learning*, pages 416–426, 2023.

[143] Caleb Rascon and Ivan Meza. Localization of sound sources in robotics: A review. *Elsevier Robotics and Autonomous Systems*, 96:184–210, 2017.

[144] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *IEEE International Conference on Robotics and Automation*, pages 1316–1322, 2015.

[145] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[146] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters*, 5(1):227–234, 2019.

[147] James F Roberts, Timothy Stirling, Jean-Christophe Zufferey, and Dario Floreano. 3d relative positioning sensor for indoor flying robots. *Springer Autonomous Robots*, 33(1): 5–20, 2012.

[148] Tobias Rodemann, Gokhan Ince, Frank Joublin, and Christian Goerick. Using binaural and spectral cues for azimuth and elevation localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2185–2190, 2008.

[149] Brandon Rothrock, Ryan Kennedy, Chris Cunningham, Jeremie Papon, Matthew Heverly, and Masahiro Ono. Spoc: Deep learning-based terrain classification for mars rover missions. In *AIAA SPACE*, page 5539, 2016.

[150] Cristina Rottondi, Riccardo di Marino, Mirko Nava, Alessandro Giusti, and Andrea Bianco. On the benefits of domain adaptation techniques for quality of transmission estimation in optical networks. *Optical Society of America Journal of Optical Communications and Networking*, 13(1):A34–A43, 2021.

[151] Loris Roveda, Marco Maroni, Lorenzo Mazzuchelli, Loris Praolini, Giuseppe Bucca, and Dario Piga. Enhancing object detection performance through sensor pose definition with bayesian optimization. In *IEEE International Workshop on Metrology for Industry 4.0 & IoT*, pages 699–703, 2021.

[152] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Detecting flying objects using a single moving camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5): 879–892, 2016.

[153] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021.

[154] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[155] Martin Saska, Tomas Baca, Justin Thomas, Jan Chudoba, Libor Preucil, Tomas Krajnik, Jan Faigl, Giuseppe Loianno, and Vijay Kumar. System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization. *Springer Autonomous Robots*, 41(4):919–944, 2017.

[156] M Sauvee, P Poignet, E Dombre, and E Courtial. Image based visual servoing through nonlinear model predictive control. In *IEEE Conference on Decision and Control*, pages 1776–1781, 2006.

[157] Aseem Saxena, Harit Pandya, Gourav Kumar, Ayush Gaud, and K Madhava Krishna. Exploring convolutional networks for end-to-end visual servoing. In *IEEE International Conference on Robotics and Automation*, pages 3817–3823, 2017.

[158] Daniel Seita, Nawid Jamali, Michael Laskey, Ajay Kumar Tanwani, Ron Berenstein, Prakash Baskaran, Soshi Iba, John Canny, and Ken Goldberg. Deep transfer learning of pick points on fabric for robot bed-making. In *Springer The International Symposium of Robotics Research*, pages 275–290, 2019.

[159] Christos Sevastopoulos and Stasinos Konstantopoulos. A survey of traversability estimation for mobile robots. *IEEE Access*, 10:96331–96347, 2022.

[160] Ravid Shwartz-Ziv and Yann LeCun. To compress or not to compress – self-supervised learning and information theory. *arXiv*, page 2304.09355, 2023.

[161] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2009.

[162] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[163] David Stavens and Sebastian Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. In *PMLR Uncertainty in Artificial Intelligence*, pages 469–476, 2006.

[164] Ron Streicher and Wes Dooley. Basic stereo microphone perspectives: A review. *Audio Engineering Society Journal of the Audio Engineering Society*, 33(7/8):548–556, 1985.

[165] Bilal Taha and Abdulhadi Shoufan. Machine learning-based drone detection and classification: State-of-the-art in research. *IEEE Access*, 7:138669–138682, 2019.

[166] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2759–2764, 2016.

[167] Ryu Takeda and Kazunori Komatani. Sound source localization based on deep neural networks with directional activate function exploiting phase information. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 405–409, 2016.

[168] Ryu Takeda and Kazunori Komatani. Unsupervised adaptation of deep neural networks for sound source localization using entropy minimization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2217–2221, 2017.

[169] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems*, 30, 2017.

[170] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[171] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30, 2017.

[172] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3482–3489, 2018.

[173] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in Neural Information Processing Systems*, 35:10078–10093, 2022.

[174] Stefano Toniolo, Jérôme Guzzi, Alessandro Giusti, and Luca Maria Gambardella. Learning an image-based obstacle detector with automatic acquisition of training data. In *AAAI Conference on Artificial Intelligence*, 2018.

[175] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *PMLR Conference on Robot Learning*, pages 306–316, 2018.

[176] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2897–2905, 2018.

[177] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017.

[178] Michelle Valente, Cyril Joly, and Atnaud de La Fortelle. Deep sensor fusion for real-time odometry estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6679–6685, 2019.

[179] Steven Van Der Helm, Mario Coppola, Kimberly N McGuire, and Guido CHE de Croon. On-board range-based relative localization for micro air vehicles in indoor leader-follower flight. *Springer Autonomous Robots*, 44(3):415–441, 2020.

[180] Kevin Van Hecke, Guido CHE de Croon, Daniel Hennes, Timothy P Setterfield, Alvar Saenz-Otero, and Dario Izzo. Self-supervised learning as an enabling technology for future space exploration robots: ISS experiments on monocular distance learning. *Elsevier Acta Astronautica*, 140:1–9, 2017.

[181] Matouš Vrba, Daniel Heřt, and Martin Saska. Onboard marker-less detection and localization of non-cooperating drones for their safe interception by an autonomous aerial system. *IEEE Robotics and Automation Letters*, 4(4):3402–3409, 2019.

[182] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019.

[183] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6d: Self-supervised monocular 6d object pose estimation. In *European Conference on Computer Vision*, pages 108–125, 2020.

[184] Gu Wang, Fabian Manhardt, Xingyu Liu, Xiangyang Ji, and Federico Tombari. Occlusion-aware self-supervised monocular 6d object pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[185] John Wang and Edwin Olson. Apriltag 2: Efficient and robust fiducial detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4193–4198, 2016.

[186] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *IEEE International Conference on Robotics and Automation*, pages 2043–2050, 2017.

[187] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *SAGE Publications The International Journal of Robotics Research*, 37(4-5):513–542, 2018.

[188] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *IEEE Signals, Systems & Computers*, volume 2, pages 1398–1402, 2003.

[189] Yi Wei, Shaohui Liu, Wang Zhao, and Jiwen Lu. Conditional single-view shape generation for multi-view stereo reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9651–9660, 2019.

[190] B. L. Welch. The generalization of student's problem when several different population variances are involved. *Biometrika*, 34(1):28–35, 1947.

[191] Lorenz Wellhausen, Alexey Dosovitskiy, René Ranftl, Krzysztof Walas, Cesar Cadena, and Marco Hutter. Where should i walk? predicting terrain properties from images via self-supervised learning. *IEEE Robotics and Automation Letters*, 4(2):1509–1516, 2019.

[192] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger B Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *International Conference on Learning Representations*, 2018.

[193] Shaoen Wu, Junhong Xu, Shangyue Zhu, and Hanqing Guo. A deep residual convolutional neural network for facial keypoint detection with missing labels. *Elsevier Signal Processing*, 144:384–391, 2018.

[194] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems*, 2018.

[195] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.

[196] Hao Xu, Luqi Wang, Yichen Zhang, Kejie Qiu, and Shaojie Shen. Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm. In *IEEE International Conference on Robotics and Automation*, pages 8776–8782, 2020.

[197] Nelson Yalta, Kazuhiro Nakadai, and Tetsuya Ogata. Sound source localization using deep learning models. *Fuji Technology Press Journal of Robotics and Mechatronics*, 29(1): 37–48, 2017.

[198] Cunjun Yu, Zhongang Cai, Hung Pham, and Quang-Cuong Pham. Siamese convolutional neural network for sub-millimeter-accurate camera pose estimation and visual servoing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 935–941, 2019.

[199] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[200] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *IEEE International Conference on Robotics and Automation*, pages 1386–1383, 2017.

[201] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4238–4245, 2018.

[202] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666, 2016.

[203] Wei Zhang, Qi Chen, Weidong Zhang, and Xuanyu He. Long-range terrain perception using convolutional neural networks. *Elsevier Neurocomputing*, 275:781–787, 2018.

[204] Shengyan Zhou, Junqiang Xi, Matthew W McDaniel, Takayuki Nishihata, Phil Salesses, and Karl Iagnemma. Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain. *Wiley Online Library Journal of Field Robotics*, 29(2):277–297, 2012.

[205] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[206] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

[207] Andrew Zisserman. Self-Supervised Learning Slides. `https://project.inria.fr/paiss/files/2018/07/zisserman-self-supervised.pdf`, Accessed: 2023.

[208] Jannik Zürn, Wolfram Burgard, and Abhinav Valada. Self-supervised visual terrain classification from unsupervised acoustic feature learning. *IEEE Transactions on Robotics*, pages 466–481, 2020.