COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Nantes Université

THÈSE DE DOCTORAT DE

NANTES UNIVERSITÉ

ÉCOLE DOCTORALE N° 641
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *« voir liste sur le site de votre école doctorale »*

Par

Kevin Riou

-

-

Thèse présentée et soutenue à Nantes, le 30/04/2024
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N)
Thèse N° : « si pertinent »

Rapporteurs avant soutenance :

Prénom NOM Fonction et établissement d'exercice
Prénom NOM Fonction et établissement d'exercice
Prénom NOM Fonction et établissement d'exercice

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	Prénom NOM	Fonction et établissement d'exercice *(à préciser après la soutenance)*
Examinateurs :	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
Dir. de thèse :	Patrick LE CALLET	Professeur, Nantes Université, France
Co-dir. de thèse :	Kevin SUBRIN	Maitre de conférence, Nantes Université, France

Invité(s) :

Prénom NOM Fonction et établissement d'exercice

ABSTRACT

ACKNOWLEDGEMENT

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

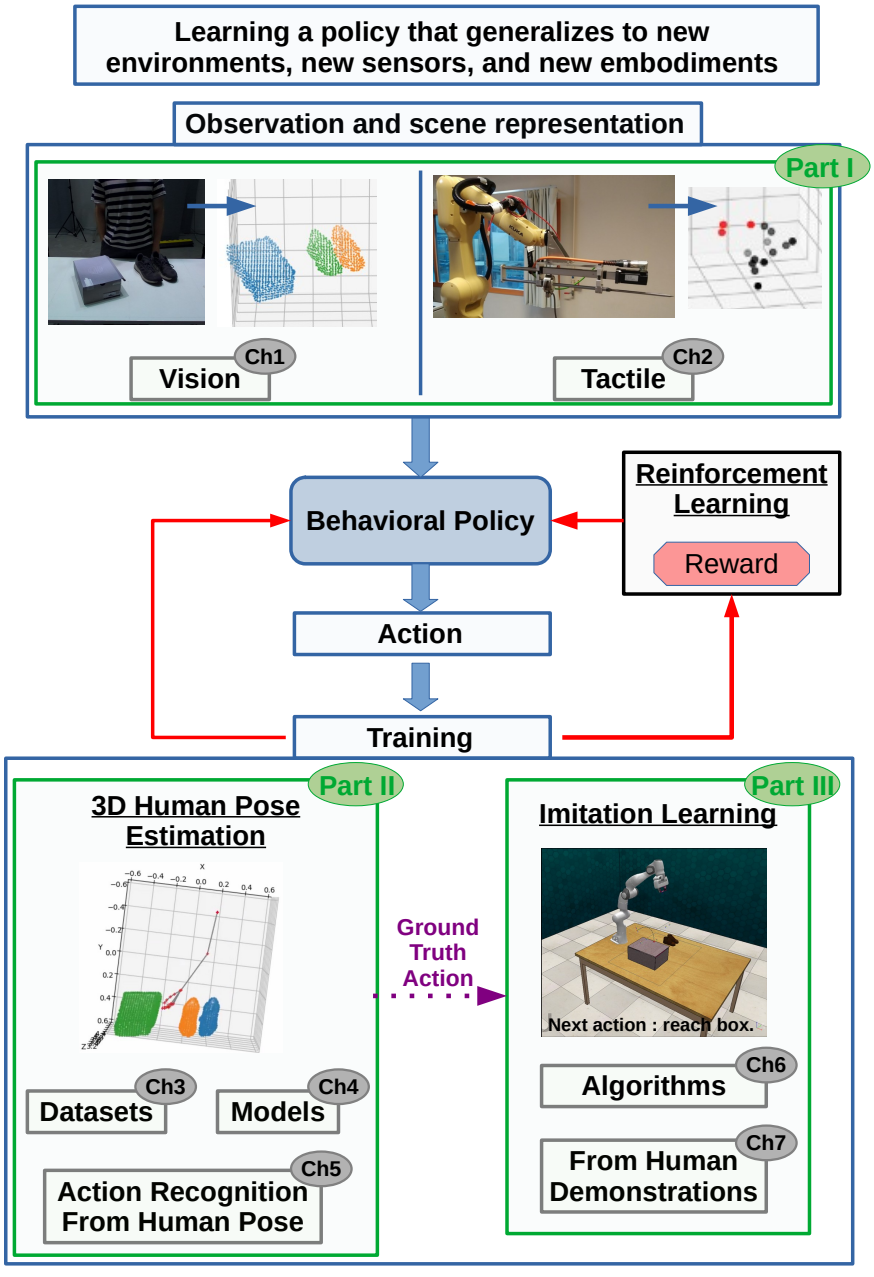


Figure 1.1 – Overview of the manuscrupt parts and chapters.

PART I

Observation space and scene representation for robotic manipulation

Overview

In the realm of robotic manipulation task learning, several fundamental components are essential: the action space, the observation space, the policy (a model trained to plan actions from observations), the objective function encoding the task's goal, and the learning algorithm. Essentially, the observation space is the input of the policy, while the action space is the policy's output. The objective function supervises the learning algorithm used to train the policy.

The observation space is the robot's perception of its environment. In imitation learning, the policy training is guided by task demonstrations provided by a human operator. Here, we define two types of observation spaces and two types of action spaces: those of the human operator and those of the robot. Ensuring alignment between the robot's and the human operator's observation and action spaces is crucial for successfully deploying a policy trained on human demonstrations to a robot.

The discussion in this Part focuses on the robot's observation space and the challenge of representing the scene in a manner invariant to the embodiment. Specifically, the scene representation should not depend on the sensor's type and configuration, nor should it highlight features unique to the operator performing the task (human or robot). We concentrate on two primary sources of information for fine robotic manipulation tasks: vision and tactile sensing.

Chapter 2 delves into vision, while Chapter 3 addresses tactile sensing. We propose a point-cloud-based observation space applicable to both vision and tactile sensing, which can serve as input to control models. Additionally, Chapter 3 employs reinforcement learning to simultaneously learn a scene representation and a tactile exploration policy for the point-cloud-based observation space. The insights gained from Chapter 2 will be leveraged in part III to bridge the gap between human demonstrations and robot execution in imitation learning.

CHAPTER 2

VISION

Overview

Contents

2.1 Introduction

Robot vision refers to the capability of a robot to visually perceive the environment and use this information for executing various tasks [90]. However, raw vision data presents significant challenges due to its high-dimensional and redundant nature. This data often contains noisy information irrelevant to the task at hand, such as lighting conditions or background clutter. Consequently, training models on raw vision data is computationally expensive and requires a large amount of data to manage the high dimensionality and noise.

An additional challenge in robot vision is the embodiment mismatch, where a model trained on images of a specific robot may not generalize well to other robots. This issue is particularly pronounced in imitation learning, the focus of Part III, where models trained on images from

human demonstrations may struggle to perform accurately when deployed on a robot. Viewpoint invariance further complicates this problem, as models trained on images from a specific viewpoint may not generalize well to other perspectives.

To illustrate, consider a scenario in robotic manipulation where a human operator provides a few demonstrations of a repetitive task to a robot. The robot must quickly learn the task from these demonstrations, but it faces the embodiment mismatch between human and robot, viewpoint variance, and the need to focus on the task's relevant aspects from a few examples. Preprocessing raw vision data is crucial to reduce its dimensionality and redundancy, and to make the model invariant to both viewpoint and embodiment.

Moreover, perceiving the environment in three dimensions offers significant advantages for robotic manipulation, as 3D information is essential for a robot to interact effectively with its surroundings. Providing control models with direct 3D information can be more efficient than having the models infer it from 2D data. Additionnally, we would like the model to be able to operate even if the background environmemt changes, e.g. if the robot is moved to a different room.

In section 2.2, we start by reviewing pretrained visual representations that allow to extract a latent representation directly from images, and that can be used as input to the control models and highlight their potential and limitations, especially focusing on the embodiment mismatch and viewpoint and environment invariance problems. The section 2.3 reviews existing methods for object detection, segmentation and tracking, and how they can be used to preprocess the raw vision data. We especially narrow the review on open-vocabulary methods that generalize well to unseen objects without finetuning, by just specifying the names of the objects of interest to the model. The most promising methods are further tested on human demonstrations that we collected to train the control models and that are presented in Part III. Finally, the section 2.4 presents solutions to project 2D information to 3D, and how it can help for viewpoint invariance. We also review solutions that allow to recover depth maps of the scene, either using depth sensor, or using deep learning models that can infer depth from 2D images, and discusses the choice of the solution retained for the data collected in Part III.

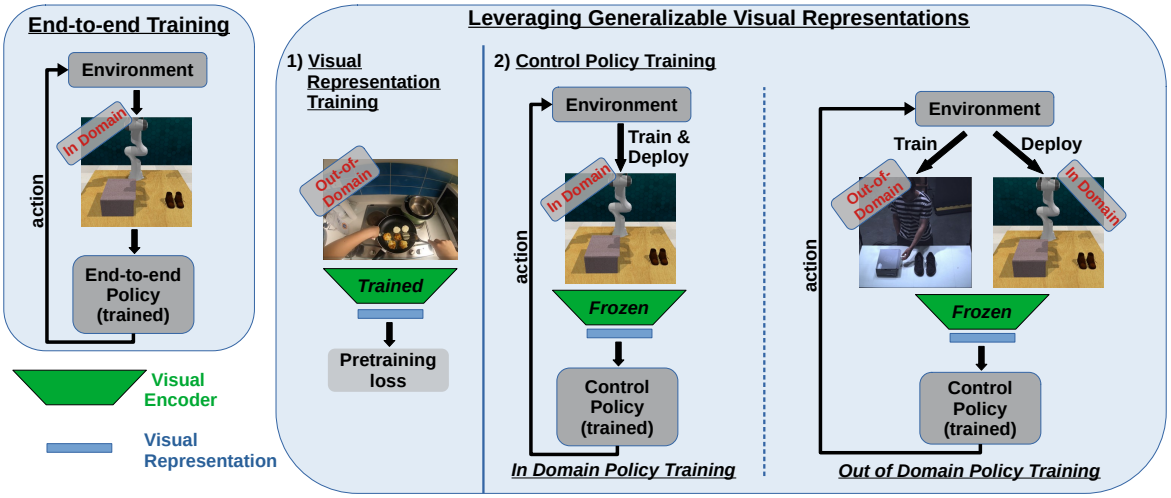


Figure 2.1 – End-to-end training vs pretrained visual representations based approaches for control policies training.

2.2 Pretrained Representations for robotic manipulation

2.2.1 Review of existing methods

The ability to quicly learn diverse manipulation skills from vision is a long-standing problem in robotics. Training end-to-end models from scratch on raw vision data is computationally expensive and requires huge amount of data, usually in the order of hundreds of demonstrations [117] and days of training to learn a single pick-and-place-like task. Therefore, the development of scalable and generalizable visual representations for robotic manipulation is not a novel problem. Inspired by the success of pretrained representations in computer vision [14, 67, 68, 5] and natural language processing [33, 131], rencent works have proposed to pre-train visual representations on large-scale human data, such as, e.g., Ego4D dataset[55], and further leverage these representations to learn robotic manipulation skills [151, 125, 119, 189].

As illustrated in Fig. 2.1 approaches leveraging pretrained visual representations for robotic manipulation usually work in two steps: 1) a deep learning model is pretrained on large scale datasets (image or video) using a pretext task. The pretext task could be supervised, meaning that the training data is labeled, e.g., image classification, or unsupervised, meaning that the training data is not labeled, e.g., contrastive learning. 2) The pretrained model is used as a visual encoder to extract visual features from the raw vision data, and an additionnal model, the control policy, is further trained to predict actions related to a robotic manipulation task from

these features. The encoder is usually frozen during the training of the control policy.

As depicted on Fig. 2.1, the datasets used for pretraining are considered as 'out-of-domain' data regarding the manipulation task. They are indeed usually made of everyday life human activities, which are not directly related to the robotic manipulation tasks, especially when considering the embodiment mismatch between the human and the robot. This is even more true if considering simulated environments during the control policy training, which are usually not representative of the real world. However, the expectation is that the pretrained model had to learn to extract generalizable visual features from the raw vision data since the training data is diverse and large-scale, and that part of these features are still relevant to the robotic manipulation tasks. As illustrated on Fig. 2.1, two scenarios can be considered for the control policy training. The first scenario is 'In Domain Policy training', where the control policy is trained on the 'In Domain' data, i.e., the data collected during the robotic manipulation tasks. Most existing imitation learning approaches leverage 'In Domain' data to train the control policy. Those training data are usually obtained by teleoperating the robot to perform the task, or by using a simulator or an expert model to generate the data [117]. During deployment, the trained model is executed on the same setup, with the same sensors and the same embodiment, to perform the task. When using a pretrained visual representation for 'In Domain Policy training', we expect the control policy to learn to isolate features that are relevant to the task at hand, and to ignore the irrelevant features that are specific to the 'out-of-domain' data. This way, the model trained on these features requires less data to learn the manipulation tasks, since it doesn't have to learn them from scratch, and can directly focus on conceptual understanding of the task. The second scenario is the 'Out of Domain Policy training', where the training data does not come from the target domain, e.g. data collected from human video demonstrations. This scenario will be discussed in the next section 2.2.2 that discusses the limitations of pretrained visual representations.

Various pretext tasks have been proposed for pretraining visual representations. The Fig. 2.2 is an overview of the catergories of pretext tasks typically used. In the supervised category, the model is trained to predict the labels of the training data, e.g., image classification. After training, the last layer of the model, which is specific to the task at hand, is removed, and the model is used as a visual encoder to extract visual features from the raw vision data. Shah et al. [151] work is an example for the supervised category. They pretrained a ResNet model [65] on the ImagetNet classification dataset [31], and further trained a reinforcement learning agent to perform robotic manipulation tasks from this representation. Many other approaches can be included in the supervised category, especially language and vision approaches. CLIP [132] is

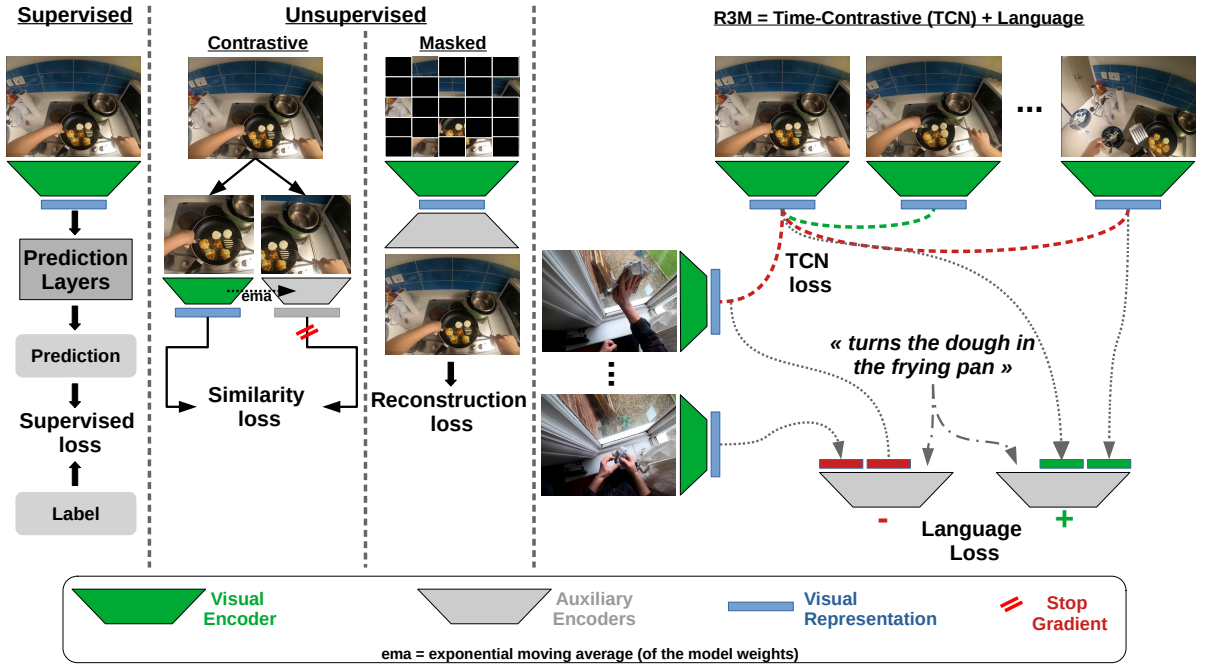


Figure 2.2 – Overview of the visual representation learning approaches.

one of the most famous example of language and vision approaches, which is trained to recover (image, language description) pairs from a shuffled batch of images and language descriptions. The approach is training a language description encoder, and an image encoder, so that the representations of the two encoders are similar if the image and the language description are related, and different otherwise. The visual encoder can be further used as a pretrained visual representation.

In the unsupervised category, no label is used during the training. Therefore, a pretext task is created to train the model. In the Fig. 2.2, we highlight two main categories of unsupervised pretext tasks: 1) Contrastive learning, where the model is trained to bring together the representations of augmented versions of the same image, and to push apart the representations of different images. 2) Masked Autoencoders, where the model is trained to reconstruct the original image from a version of it where some parts are masked.

The Momentum Contrastive (MoCo) approach, proposed by He et al. [68], is an example that uses contrastive learning to train a ResNet [65] model on the ImageNet [68] dataset. Concretely, during training, the visual encoder receives an image, while an auxiliary encoder receives either an augmented version of the same image, or a different image. The visual encoder is trained to bring together the representations of the augmented versions of the same image,

and to push apart the representations of different images. The auxiliary encoder has the same structure as the visual encoder, but its weights are updated using a moving average of the visual encoder weights. Eq. 2.1 shows the training loss corresponding to the MoCo pretext task, which is a form of InfoNCE loss [121]. q represents the representation of an image sent in the visual encoder, and k represents the representation of an image sent in the auxiliary encoder. k+ is the representation of the augmented version of the same image as q. K is the number of negative samples considered per image, i.e., the number of images that are not the augmented version of the same image as q. The loss can viewed as the log loss of a (K+1)-class softmax-based classifier, that aims to classify the query q as the positive sample k+ among the K negative samples k.

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i/\tau)} \tag{2.1}$$

The Dino model [14] further proved that the negative pairs, representing different images, are not necessary during the training, and that the model can be trained to predict similar representations of augmented versions of the same image only. In this case, setting the auxiliary encoder weights to the exponential moving average of the visual encoder weights prevent mode collapse where the visual encoder and the auxiliary encoder are the same and output the same embeddings regardless of the input.

The Masked Autoencoder models [67] can be divided in two parts: the encoder, which encodes images altered with masked patches into a latent representation, and the decoder, which reconstructs the original image from the latent representation. After training, the encoder is used as a visual encoder and the latent representation of the autoencoder is used as the pretrained visual representation.

Parisi et al. [125] demonstrated that visual representations trained with unsupervised objectives performed usually better than those trained with supervised objectives, though the difference was not significant. However, note that both supervised and unsupervised models were trained on the same datasets for fair comparison, but unsupervised models could be theoretically trained on larger unlabelled datasets, which could lead to better generalizability. They also found that using pretrained visual representations significantly improved the performance of the robotic manipulation models, compared to training the models from scratch. Additionnally, they further trained the robotic manipulation models from oracle features obtained using priviledged information from simulated environments. Those oracle features included positions and quaternions of the agent and target areas or objects in the scene. While they refer to those features as oracle ones because they are not available in the real world, they may not be unobtainable in the real world, especially with the avenue of general-purpose object detection and track-

ing models, as described in section 2.3. The manipulation models trained from oracle features showcased better performances than the ones trained from off-the-shelf the pretrained visual representations. However, they performed an additionnal experiment using the pretrained visual representations. In all previous experiments, the manipulation models were only using the visual features extracted from the last layer of the pretrained visual representations, which is known to encode high-level semantics about the scene [149]. However, robotics manipulation tasks could also benefit from earlier features that encode more spatial information about the scene. This trade-off between high-level semantics and spatial information is well known in computer vision, and is usually addressed by using so called pyramid features in object detection and segmentation models [103]. Parisi et al. [125] demonstrated that by carefully selecting a hierarchical set of features from the pretrained visual representations, they could further improve the performance of the robotic manipulation models, and even match the performances of the models trained from oracle features in some cases. Using the best combination of MoCo features, they were able to achieve a success rate of 81% on the Franka Kitchen fine manipulation benchmark [58], when pre-trained on ImageNet. The model trained from raw images and from oracle features achieved a success rate of 30% and 71% respectively.

The last category in Fig. 2.2 aims to highlight time-constrastive approaches, originally introduced in the context of imitation learning by [150]. In this case, the model is trained to bring together the representations images from a given video that are temporally close, and to push apart the representations of images that are temporally far, while also pushing apart the representations of different videos. In Fig. 2.2, the red dashed line highlights representations that shoud be pushed apart, while the green dashed line highlights representations that should be brought together. The model that is presented in this last category, Visual Representation for Robot Manipulation (R3M) [119], is actually not just a time-contrastive approach. It also incorporates a language loss, which makes it a language and vision approach, and a vision loss using the time-contrastive pretext task. The auxiliary encoder is trained to predict a high score if the language caption describes well the transition from one of the image representation to the other. Concretely, the representation of two images from the same video and temporally close do not represent well the task description, since they probably encompass a short part of the task execution only. On the contrary, the representation of two images from the same video and temporally far represent well the task description, since they probably encompass the whole task execution. Additionally, the representation of two images from different videos do not represent well the task description, since they probably encompass different tasks. Concretely, the training loss corresponding to the time-contrastive pretext task is defined as follows. A batch of

triplets of images $[I_i, I_{j>i}, I_{k>j}]^{1:B}$, where images I_i, $I_{j>i}$ and $I_{k>j}$ are from the same video b is sampled. i, j and k are indices of the images in the video. An additionnal batch of images $I_i^{\neq b}$ from different videos is also sampled. The corresponding visual representations z_i^b, $z_{j>i}^b$, $z_{k>j}^b$ and $z_i^{\neq b}$ are extracted from the visual encoder. The TCN loss is by Eq. 2.2, which is also a form of InfoNCE loss [121].

$$\mathcal{L}_{\text{tcn}} = -\sum_{b \in B} \log \frac{e^{\mathcal{S}(z_i^b, z_j^b)}}{e^{\mathcal{S}(z_i^b, z_j^b)} + e^{\mathcal{S}(z_i^b, z_k^b)} + e^{\mathcal{S}(z_i^b, z_i^{\neq b})}} \tag{2.2}$$

where S is a similarity function, e.g., negative L2 distance.

The language loss is defined in a similar fashion by Eq. 2.3, where l^b is the language description of the video b, and $\mathcal{G}_\theta$ is the auxiliary encoder, and z_0 is the representation of the first image of a video.

$$\mathcal{L}_{\text{language}} = -\sum_{b \in B} \log \frac{e^{\mathcal{G}_\theta(z_0^b, z_{j>i}^b, l^b)}}{e^{\mathcal{G}_\theta(z_0^b, z_{j>i}^b, l^b)} + e^{\mathcal{G}_\theta(z_0^b, z_i^b, l^b)} + e^{\mathcal{G}_\theta(z_0^{\neq b}, z_{j>i}^{\neq b}, l^b)}} \tag{2.3}$$

Finally, more recently, VIP [115] proposed to cast the visual representation learning as an offline goal-conditioned reinforcement learning problem. The idea is to train a goal-conditionned policy, π^H, on a large-scale human dataset, so that the actions taken by the policy should be similar to the actions a^H taken by the human, for a given goal g : $a^H \sim \pi^H\left(\phi(o) \mid \phi(g)\right)$. Here, ϕ is the visual encoder that we seek to train. g is the goal image, which is easily defined as the last image of a video demonstration. However, a^H is defined on the Human action space, which can not be obtained from the human datasets, which are only made of video data. Nevertheless, the offline RL objective that would be optimize to train this policy can be defined by setting up dummy human actions $\tilde{a}^H$. Eq. 2.4 shows the optimization objective of the VIP model. In that equation, the main objective of the policy is to maximize the discounted sum of rewards, $[\sum_t \gamma^t r(o; g)]$, where $r(o; g)$ is the reward function that provides a negative reward (-1) if the goal is not reached, and a positive reward (0) if the goal is reached. γ^t, where the discounting factor $\gamma < 1$, essentially says that immediate rewards are more important than later rewards. In other words, the closer we are to the goal, the higher the sum of discounted rewards. This may already remind the reader of the time-contrastive objectives described earlier in this section. The policy is also trained to minimize the KL divergence between the human policy π^H and the policy π that is trained on the offline RL objective, which ensures that the policy's actions are

similar to the human actions.

$$\max_{\pi_H,\phi} \mathbb{E}_{\pi_H}\left[\sum_t \gamma^t r(o;g)\right] - D_{KL}(d^{\pi_H}(o,a^H;g)\|d^D(o,\tilde{a}^H;g)) \tag{2.4}$$

From Eq. 2.4, two questions arise: 1) Where is the visual representation learning in this equation? 2) How can we optimize this objective without access to human actions $\tilde{a}^H$?

Ma et al. [115] demonstrated that the optimization objective of Eq. 2.4 can be rewritten as defined in Eq. 2.5. The first noticeable change is that the policy is now defined as a function of the visual representation $\phi(o)$, and the goal representation $\phi(g)$, where ϕ is the visual encoder. The second change is that the optimization objective now does not require the human actions $\tilde{a}^H$. The function V introduced in Eq. 2.5 is a value function that is trained to estimates the discounted total number of steps required to reach the goal g from the observation o. Note that o' represents the obseration after taking an action a from the observation o. In other words, o is the image at time t, and o' is the image at time $t+1$. Since the value function V leverages the visual representation $\phi(o)$, the visual encoder is trained to provide a visual representation that is useful to estimate the discounted temporal distance between two observations in a video demonstration.

$$\max_{\phi}\min_{V} \mathbb{E}_{p(g)}\left[(1-\gamma)\mathbb{E}_{\mu_0(o;g)}[V(\phi(o);\phi(g))] + \log \mathbb{E}_{(o,o';g)\sim D}\left[\exp\left(r(o,g)+\gamma V(\phi(o');\phi(g)) - V(\phi(o),\phi(g))\right)\right]\right] \tag{2.5}$$

Interestingly, by rephrasing the VIP training objective defined in Eq. 2.5, the authors demonstrated that this objective can be seen as a form of time-contrastive learning. The Eq. 2.6 is the optimization objective of the visual representation ϕ only (considering that the optimal Value function $V*$ is already found), and after a few rephrasing steps. Recall that the value function V estimates a form of distance between two observations in a video demonstration. If we denote (o, g) as positive pairs and (o', g) as negative pairs, since (o, g) are temporally closer than (o', g), then the visual encoder is trained to predict representations that are further away for the positive pairs of observations (o, g) than for the negative pairs of observations (o', g). This is a form of time-contrastive learning, but in this opposite way than the one presented in the Fig. 2.2, where the representations of temporally close images are brought together, and the representations of

temporally far images are pushed apart.

$$\min_{\phi}(1-\gamma)\mathbb{E}_{p(g),\mu_0(o;g)}\left[-\log\frac{e^{V^*(\phi(o);\phi(g))}}{\mathbb{E}_{D(o,o';g)}\left[\exp(\tilde{\delta}_g(o)+\gamma V^*(\phi(o');\phi(g))-V^*(\phi(o),\phi(g)))\right]^{\frac{-1}{(1-\gamma)}}}\right] \tag{2.6}$$

The strength of this representation compared to previous time-contrastive approaches is its high temporal smoothness. If one computes the distance between observations o_t and goal image in a video, the distance will decrease smoothly as the time t increases, almost monotically, while other approaches, like R3M [119], ehibit bumps (positive slope) on the distance curves. This is particularly interesting is one wants to use the visual representation not only as input to the control policy, but also to define a reinforcement learning reward from it. The authors indeed defined a reward function $R(o_t, o_{t+1}; \phi, \{g\}) := S_\phi(o_{t+1}; g) - S_\phi(o_t; g)$, where $S_\phi(o_t; g) = -||\phi(o_t) - \phi(g)||_2$, which allows to train an offline RL policy if one has access, e.g., to a dataset of teleoperated robot demonstrations, where observations and actions are available, and where the goal images g are the last images of the demonstrations, but where the rewards are not available.

Ma et al. [115] demonstrated that their model outperformed the above mentioned pretrained visual representations on sevral tasks of the FrankaKitchen [58] by dozens of success rate points, both when using the visual representation as input to a control policy, and when using the visual representation to define a reward function for offline RL.

2.2.2 Limitations of Pretrained Representations

The pretrained visual representations provide a generalizable and scalable way to extract visual features from the raw vision data. Training models on in-domain data, i.e., data collected by teleoperating a robot, allows the control policy to learn the relevant features for the task at hand, among the generalizable features extracted by the pretrained visual representations. However, our goal is to train the control policy from human demonstrations, which is out-of-domain data regarding the task at hand. It is out-of-domain data in terms of embodiment (human-robot), environmenment (real-simulated), and viewpoint (that are not the same between the human and the robot). In this scenario, depicted as 'Out of Domain Policy training' in Fig. 2.1, the control policy will learn the relevant features to solve the manipulation task from the out-of-domain data. This way, during deployment, the control policy will face out-of-training-

distribution data (the in-domain data), and may not generalize well.

Sevral solutions have been proposed to bridge the gap between the out-of-domain data and the in-domain data, such as Sim-to-Real or Human-to-Robot transfer. We review some of these approaches in Annex A. However, all these approaches require to have access to in-domain data for training, which means that the robot has to be teleoperated to collect the data, which is what we are trying to avoid by training the control policy from human demonstrations. This motivates the need to develop visual representations that are invariant to the embodiment and to the environment, and that can be used to train the control policy from out-of-domain data. This will be the focus of the next section. Neverthless, the VIP representation will be benchmarked as an alternative to our proposed visual representation in Part III.

2.3 Detecting and segmenting objects of interest

An ideal scene representation should focus on the objects relevant to the task at hand, track them over time, and abstract the operator from the scene. In this section, with review existing models for object detection, segmentation, and tracking, and discuss their ability to provide a scene representation that is invariant to the embodiment, to the environment and that can later be extended to the 3D space.

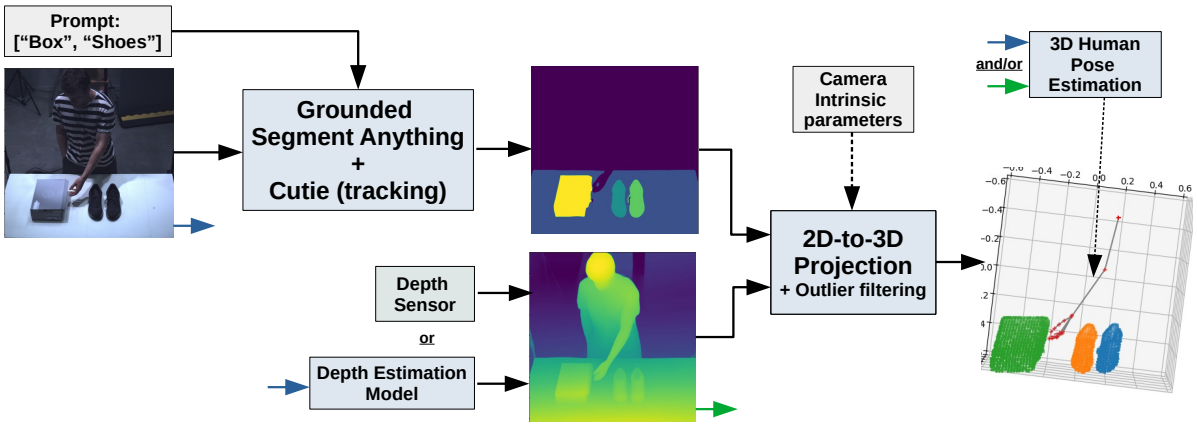


Figure 2.3 – Overview of our pipeline to exract the scene representation from the raw vision data.

2.3.1 Definitions, benchmarks and metrics

Tasks definitions

The **object detection** problem consists in providing a bounding box around the objects of interest in images, along with the object category for each bounding box.

More recently, the **Open-Set Object Detection** problem has regained interest in the community, where the object detection model is trained on a set of known classes, and is able to detect objects of unknown classes during testing. The Openset learning problem is by no means a new problem [148], but was recently improved for object detection by the introduction of large language and vision models, such as CLIP [132], which can leverage the language information to detect objects of unknown classes during testing [201]. Tackling the Open-Set Object Detection problem with the help of language generalization is denoted as **Open-Vocabulary Object Detection**.

The object **segmentation** problem consists in providing a mask around the objects of interest in images, along with the object category for each mask. In other worlds, the object segmentation problem is a pixel-wise classification problem, where each pixel of the image is classified as either belonging to an object of interest or not. Two main types of segmentation exist: instance segmentation, where each object of interest is segmented separately, so that different instances of the same object category are segmented separately, and semantic segmentation, where all objects of the same category are segmented together.

Benchmarks and metrics

Since we are not going to train and benchmark object detection and segmentation models we relegate a more detailed review of benchmarks and metrics to Annex B. Chronologically, the PASCAL VOC [36] (VOC2007 then VOC2012), and later the COCO [105] datasets have been the most used benchmarks for object detection and segmentation. The PASCAL VOC2007 also introduced the mean Average Precision (mAP) metric, which is the most used metric for object detection and segmentation. The mAP is basically impacted by two main factors: the precision and the recall. The precision checks how many objects detected by the model are actually objects of interest, while the recall checks how many objects of interest are detected among all the objects of interest. The mAP ranges from 0 to 100, where 100 is the best possible score. The COCO dataset introduced a refined versions of the mAP metric, which averages different mAP scores for different thresholds on deciding if an object was correctly detected or not. In the following, we denote the PASCAL VOC mAP as VOC-mAP@0.5, and the COCO

mAP as either COCO-mAP@0.5 or COCO-mAP@[0.5:0.95]. More details on the mAP metric can be found in Annex B.

2.3.2 Detection

Traditionnal vs Deep-learning based object detection

The object detection developments can be divided in two main periods [212]. The “traditional object detection" period, before 2012 and the “deep learning-based detection" period, after 2012. The traditional object detection methods were based on hand-crafted features [174, 29, 39], extracted from various regions of the image, and usually fed to a classifier to predict the object categories of the regions from the extracted features. The hand-crafted features showed limitations in terms of generalization to new object categories, as it is nearly impossible to design features that are relevant to all object categories.

After rebirth of convolutional neural networks (CNNs) [92], the object detection developments shifted to deep learning-based methods. This way, the features are learned from the data, which eliminates the need to design hand-crafted features for each new object category.

Two stage detectors

The deep learning-based object detection methods can be divided in two main categories: one-stage detectors and two-stage detectors. The two-stage detectors essentially follow a coarse-to-fine approach, where the first stage extracts a set of region proposals, where objects of interest are likely to be, and the second stage refines the proposals and predicts the object categories. Proposed in 2014, the Region-based CNN (R-CNN) [50, 49] is one of the first two-stage detectors. It uses a selective search algorithm [172] to extract region proposals, a CNN pretrained on ImageNet [31] to extract features from each of the selected regions, and a classifier to predict if the chosen regions contain objects of interest, and the object categories. The introduction of R-CNN improved the state-of-the-art performances on the PASCAL VOC2007 dataset from about 33% VOC-mAP@0.5 (with former traditionnal object detection approaches) to 58.5% VOC-mAP@0.5. SPPNet [69] (2014) proposed to extract features from the whole image, and to use a spatial pyramid pooling layer to aggregate the features from the regions of different sizes. This way, the features are only extracted once from the whole image, which makes SPPNet 20 times faster than R-CNN, while achieving 59.2% VOC-mAP@0.5 on the VOC2007 dataset. Moreover, this approach allowed to extract fixed-length feature vectors from the input images, regardless of the image size and aspect ratio.

Ross Girshick proposed the Fast R-CNN [48] model in 2015, which improves upon the R-CNN and SPPNet models by sharing the convolutional layers between the region proposal network and the classifier, and achieved 70.0% mAP on the VOC2007 dataset and was 200x faster than R-CNN.

The Faster R-CNN [137] model further improves upon the Fast R-CNN model by proposing a Region Proposal Network (RPN), which is a CNN that predicts the region proposals from the input image, and shares the convolutional layers with the classifier. The Faster R-CNN achieves 73.2% mAP on the VOC2007 dataset. The COCO benchmark started to lead the object detection developments from that time, and the Faster R-CNN achieved 42.7% COCO-mAP@0.5 on the COCO dataset. All the modules from the region proposal to the classifier have basically been integrated into an end-to-end trained model. The Faster R-CNN is the first near-real time two stage detector (up to 17 FPS, on a K40 GPU).

In 2017, Lin et al. proposed the Feature Pyramid Network (FPN) [103]. Before FPN, the feature were extracted from the last layer of the CNN, which is known to encode high-level semantics about the scene but looses the spatial information. FPN proposes a mechanism to upsample the features from the last layer (semantically rich) to the features from the first layers (high spatial resolution), and to combine them. This way, the features are extracted at various scales, and are semantically rich and spatially detailed. Unsurprisingly, the FPN showed strong performances on detecting objects of various sizes. The FPN achieved 59.1% COCO-mAP@0.5 on the COCO dataset.

One stage detectors

In parallel to the two-stage detectors, one-stage detectors were also developed. The coarse-to-fine approach used in the two-step approaches allows to achieve high precision but are rarely used in real-time applications, due to their high computational cost. The one-stage detectors, on the contrary, are usually faster, as they can retrieve the objects in one step.

One of the first one-stage detectors is the YOLO (You Only Look Once) model [136]. The YOLO model divides the input image into a grid. For each cell in the grid, the model predicts a set of bounding boxes, each with a confidence score, which represents the probability that the bounding box contains an object. The model also predicts a set of object class probabilities for each bounding box. The bounding boxes are defined by their center coordinates relative to the center of the cell, their width and their height. This way, the YOLO model can be seen as an end-to-end regression model that regresses the bounding boxes coordinates, relative to the cells centers. The original YOLO model was already able to achieve real-time object detection,

but showed limitations, especially in terms of detecting small objects. The first YOLO models offered trade-offs ranging from 52.7% VOC-mAP@0.5 on VOC2007 at 155 FPS to 63.4% VOC-mAP@0.5 on VOC2007 at 45 FPS, on a Titan X GPU. Results on the COCO dataset were not provided in the original YOLO paper. These models are to be compared to the Faster R-CNN model from the above section (73.2% VOC-mAP@0.5 on VOC2007, 17 FPS at most).

The SSD (Single-Shot Multibox Detector) model [107] improved upon the YOLO model by using a set of default bounding boxes, called anchor boxes, that are defined at different scales and aspect ratios. The model predicts the offsets of the anchor boxes to fit the objects in the image. The SSD model showed better performances than the YOLO model for small objects, and already achieved competitive performances with the two-stage detectors of that time, while being faster. The SSD model achieved 74.3% VOC-mAP@0.5 on VOC2007, and 41.2% COCO-mAP@0.5 on the COCO dataset while running at 59 FPS on a Titan X GPU.

Lin et al. noticed that the extreme foreground-background class imbalance during the training of previous models was a limitation to the detection performances. They proposed the "focal loss" that allows to down-weight the loss assigned to well-classified examples, and to focus on the hard examples, and integrated it in the RetinaNet model [104]. The RetinaNet model achieved COCO-mAP@.5 = 59.1% on the COCO dataset.

More recent works tend to work as anchor-free detectors, which do not rely on predefined anchor boxes. CornerNet [97] and CenterNet [210] are examples of anchor-free detectors, achieving respectively 57.8% and 61.1% COCO-mAP@0.5 on the COCO dataset. These models predict the bounding boxes by regressing the corner points of the bounding boxes, or the center points of the bounding boxes, respectively, but in the absolute coordinates of the image. They further regress attributes of the bounding boxes, like the width and the height, and the object class.

Transformer-based object detection

Additionnally, recent works incorporate the transformer architecture [173] in the object detection models. The DETR (DEtection TRansformer) model [13] is an example of such models. The DETR model extracts features from the input image using a CNN, and then uses a transformer encoder-decoder architecture to predict the bounding boxes and the object classes from the extracted features. DETR also works as an end-to-end model, and does not require the predefined anchor boxes.

The swin transformer [110] is another example of transformer-based object detection model. It applies the transformer architecture directly to the image, but in a hierarchical way, where the

image is divided into patches to mitigate the quadratic complexity of the self-attention mechanism. The patchs are processed individually by the transformer, pooled and processed by a new transformer layer, several times until the final output.

However, the latest version of the YOLO model, YOLOv7 [176], surprisingly outperforms the DETR and most of the swin transformer models on the COCO dataset [105], while maintaining real-time performances. The YOLOv7 runs at 161 FPS on a Nvidia V100 GPU and achieves 74.4% COCO-mAP@0.5 and 51% COCO-mAP@[0.5:0.95] on the COCO dataset. The YOLOv7-E6E model, which achieves 57% COCO-mAP@[0.5:0.95] on the COCO dataset, can only be outperformed by the dual swin transformer [102] (59% COCO-mAP@[0.5:0.95]) which can barely achieve 10 FPS on the same Nvidia V100 GPU, while the YOLOv7-E6E model runs over 30 FPS on the same GPU.

Open Vocabulary Object Detection

Open-Vocabulary Object Detection models aim to detect objects described by arbitrary textual descriptions, and not only the objects from a predefined set of classes, which was the case for the above mentionned approaches.

Several architectures have been proposed to align the text and image embeddings, especially using a pre-trained CLIP model [132], which is trained to predict similar embeddings for pairs of images and texts that are semantically similar, and different embeddings for pairs of images and texts that are semantically different. OV-DETR [200] conditions a DETR model on the text embeddings of the CLIP model, and ensures that the objects detected by DETR have the same embeddings as the text embeddings of the CLIP model. The ViLD [56] uses a two-stage object detector, where the regions of interest proposed by the first stage are aligned to match the embeddings of the clip model on those regions. This way, the regions of interest can be matched to a set of proposed textual descriptions of the objects of interest.

The Grounded Language-Image Pre-training (GLIP) architecture takes both the image and the text as input, and predicts the bounding boxes of the objects described by the text in the image. More specifically, the GLIP architecture is composed of a series of textual information processing layers, a series of visual information processing layers, along with fusion modules that combine the textual and visual information after each layer. The visual layers ultimately predict a set of regions of interest in the image with corresponding embeddings, and the textual layers predict a set of textual embeddings. The GLIP is trained to minimize the distance between the embeddings of the regions of interest and the embeddings of the textual descriptions of the objects of interest, but is also simulatenously trained to predict the bounding boxes and classes

of the objects of interest in the image.

Grounding DINO [106] uses the "DETR with Improved deNoising anchOr boxes" (DINO) architecture in a similar fashion to the GLIP architecture, but incorporates the feature fusion between the textual and visual embeddings from the very first modules of the model (image and text encoders) to the very last ones (the bounding box and class predictors). To the best of our knowledge, as of the time of writing this thesis, the Grounding DINO model achieves the state-of-the-art performances the Open-Vocabulary Object Detection tasks on COCO and LVIS datasets adapted to the Open-Vocabulary Object Detection tasks (see Annex B).

Limitations and benefits of object detection models for scene representation

The open vocabulary object detection models provide a way to detect objects of interest in the scene, without requiring to retrain the model on the new object categories. This is particularly interesting, since it alleviates the need to collect and annotate data for the new object categories, which can be time-consuming and expensive. However, the object detection outputs are usually 2D bounding boxes, which do not provide the 3D information of the objects in the scene. To project a 2D point to 3D, we need to know the depth of the point, which can be recovered from depth sensors or from depth estimation models, as discussed in section 2.4. However, since the edges of the bounding boxes are not always aligned with the edges of the objects, the depth estimation from the bounding boxes would not correspond to the depth of the objects. A finer segmentation of the objects would be required to estimate the depth of the objects in the scene. The next section will discuss the segmentation models that can be used to segment the objects of interest in the scene, and to further project them to 3D. Neverthless, we will see that the segmentation models can be coupled with open vocabulary object detection models to segment arbitrary objects of interest in the scene, without requiring to retrain the segmentation model on the new object categories.

2.3.3 Segmentation

Segmentation models following the evolution of detection models

The development of segmentation models followed the evolution of the detection models, with the predominence of traditionnal approaches, almost until 2015, including, e.g., graph theory [40, 98], Clustering [26] and Random walks [53].

The rebirth of CNN similarly impacted the segmentation field, with the introduction of Fully Convolutional Networks (FCN) [111] in 2015.

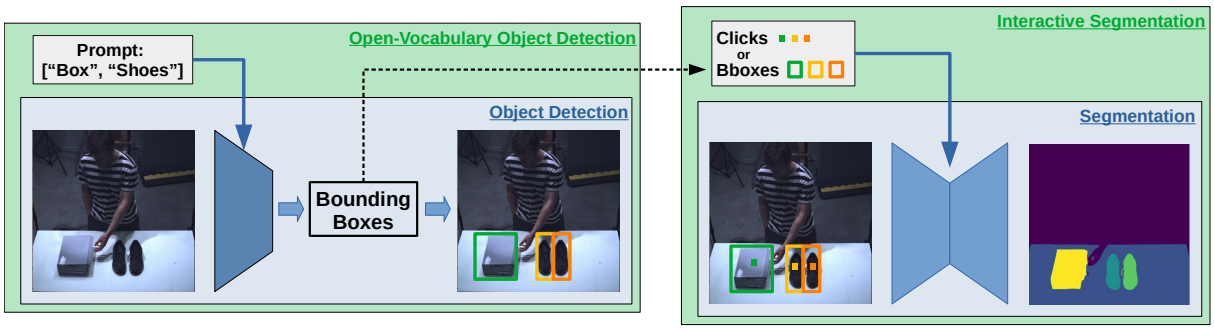


Figure 2.4 – Overview of the object detection paradigm extended to open-vocabulary object detection and instance segmentation paradigm extended to interactive segmentation. The dashed arrow further illustrates how to use open vocabulary object detection models to generate interaction prompts for the segmentation models, as proposed by the Grounded-SAM [138] authors.

Following the R-CNN series [66] in object detection, Mask R-CNN [50, 49, 137] was propsed in 2017, which extends the Faster R-CNN model to predict the object masks in addition to the bounding boxes and the object categories.

The segmentation models also benefitted from the FPN [103] with its high resolution yet semantically rich features.

The main difference between the detection and the segmentation models is that the detection model only regresses a few points (the bounding box coordinates) for each object, while the segmentation model has to reconstruct a mask with the same resolution as the input image. Several solutions have appeared to decode back a full resolution mask from the features extracted by the CNN. The FCN [111] was initially using a deconvolution layer, also called transpose convolution layer, to upsample the features to the original resolution of the image. Other up-sampling techniques have been proposed, like the unpooling [2] or the dense up-sampling convolution (DUC) [177].

Additionnaly, skip connections have been introduced by the UNET [146] to combine the features from the encoder with the features from the decoder, which allows the decoder to recover the spatial information lost during the downsampling of the features.

Finally, transformer based architectures like the ViT [34] or the swin transformer [110] have taken over the segmentation field, and have shown strong performances on the segmentation benchmarks.

While the detection models recently pivoted to open-vocabulary object detection, the segmentation models have also been extended to segment arbitrary objects of interest in the scene, but mostly using user interactions to guide the segmentation process. The interactions can be

in the form of clicks, scribbles, or bounding boxes, and can be used to segment the objects of interest arround those interactions. The Fig. 2.4 illustrates the parallel between interative segmentation and open-vocabulary object detection models, where in both case, the model is prompted with a description (text, click, bounding box) of the objects of interest.

The next section will discuss existing interactive segmentation models, and solutions to generate interaction prompts for the segmentation models from the open-vocabulary object detection models, as illustrated by the dashed arrow between left and right parts of the Fig. 2.4.

Interactive Segmentation

The interactive segmentation models leverage the same architectures as the segmentation models, but with the addition of user interactions to guide the segmentation process. For instance, the RGB channels in the FCN architecture were extended with two additionnal channels : one representing positive clicks (the user clicked on an object/region to be segmented) and one representing negative clicks (the user clicked on a region that should not be segmented) [191]. The VGG architecture was also exploited to predict various plausible masks corresponding to a set of positive and negative clicks, before chosing the most plausible mask [101]. Coarse-to-fine approaches were also proposed [18, 202] to refine the segmentation masks from the user interactions.

ViT based approaches like CFR-ICL [21] or Segment-anything [87] finally appeared and outperformed the previous interactive segmentation models. Segment-anything (SAM) in particular, allows for various interaction types including clicks, scribbles, and bounding boxes. The bounding boxe prompt is particularly interesting, as it can be coupled with open-vocabulary object detection models, as illustrated in Fig. 2.4. Grounded-SAM [138] is an example of such coupling, where a grounding dino model predicts the bounding boxes of objects described by the text, and the SAM model segments the objects described by those bounding boxes.

Our proposed pipeline to extract object of interest in the scene, illustrated in Fig. 2.3 is based on the Grounded-SAM model. We collected a dataset of human demonstrations that will serve to train imitation learning algorithms in Part III. The dataset is fully described in Chapter 8. We used the Grounded-SAM model to segment the objects of interest in the first frames of the demonstrations, and further tracked the objects in the rest of the video frames using algorithms descibed in the next section. In this section, we evaluate the performances of the Grounded-SAM model on our dataset, and especially investiguate the impact of the textual prompt on the segmentation performances.

The Fig. 2.5 shows a few examples of the segmentation masks obtained by the Grounded-

SAM model. In the top part of the figure, the model is naively prompted with a textual description of the objects of interest only. This leads to false positive detections, such as in the third image from the left, where the "shaker" is detected as a trashbin. In the bottom part of the figure, the model is prompted to segment all the objects present in the foreground scene. For instance, for the third image from the left, we explicitely ask the model to find the "shaker" along with the object of interest, which are the "trashbin" and the "crumpled paper" in the demonstrated task. That way, the detection model can realize that there's both a "shaker" and a "trashbin" in the image, and can segment them separately. The objects of interest are further filtered by class name from the other objects.

Additionnally, we can notice some false negatives in the top part of the figure, where the "trashbin" is not detected in the image on the very right. However, by introducing the "pot" keyword in addition to the "trashbin" keyword, the model is able to detect the trashbin (as a pot) in the image, as shown in the bottom part of the figure. The "pot" class is further added to the list of objects of interest during the filtering step.

This prompt engineering process can significantly affect the segmentation performances of the model in some cases. However, it doesn't help for cases where the objects are barely visible to the model, or when two objects are too similar to be distinguished by the model. For instance, on the second image from the left, the paper is not detected, with and without prompt engineering, probably because it is too far, too small and too similar to the background (table). Similarly, the "white toy" on the first image from the left is detected as a "crumpled paper" with and without prompt engineering, because they are too similar in terms of shape, color and texture.

Finally, some objects in the background are detected as either "trashin" or "crumpled paper", which is not an issue, because they will be later filtered out by depth estimation models, as discussed in the next section.

We can further notice that the open-vocabulary object detection is the bottleneck of the Grounded-SAM model on our data, since the segmentation is almost perfect when the objects are correctly detected.

Therefore, we evaluated the Precision and Recall for the two objects of interest in the scene, the "trashbin" and the "crumpled paper", using the following strategy. For each object, we counted the number of True Positives (TP), False Positives (FP) and False Negatives (FN) in the segmentation masks. The True Positives are the object of interest that are found with the right class by the detection model, without regard to the quality of the subsequent segmentation, which is not the bottleneck in our case. This eliminates the need to manually annotate the

ground truth masks, for the whole dataset, which is time-consuming and expensive. Similarly, the False Positives are the objects that are detected by the model, but that are not the object of interest, and the False Negatives are the objects of interest that are not detected by the model. The Precision and Recall are then computed as described in Annex B.

We further counted the number of examples where all objects are correctly detected, and computed the success rate as the ratio of the number of successful examples to the total number of examples. This represents the number of demonstrations where no human intervention is required to process the demonstrations, appart from the initial prompt engineering, which is done once for the whole dataset.

The results are summarized in the Table 2.1. The only metric that is not improved by the prompt engineering is the Recall of the "crumpled paper" object, since most of the False Negatives are due to the fact that the object is quite small and ressemble that background table, which makes it difficult for the model to detect it, even with the prompt engineering.

While 76% of the examples successfully annotated represents a significant reduction of the human effort required to annotate the dataset, it means that this approach is still not robust enough to be implemented in a zero-shot manner, where no huamn intervention would be required to process the demonstrations. However, with the pace of improvement in the open-vocabulary object detection models, we can expect to reach zero-shot annotation pipelines in the near future.

Table 2.1 – Evaluation of the Grounded-SAM model on the demonstration dataset proposed in Chapter 8. The model is prompted either with a naive description of the objects of interest, or with an engineered description of the objects of interest, as depicted in Fig. 2.5. The success rate represents the number of examples where all objects are correctly detected, and where no human intervention is needed to annotate it.

	Trashbin		**Crumpled Paper**		**Succ. Rate**
	Recall	Precision	Recall	Precision	
Naive Prompt	0.84	0.95	0.9	0.79	52
Engineered Prompt	0.92	1.0	0.9	0.92	76

2.3.4 Tracking models

The segmentation models described in the previous section can be used in conjunction with tracking models to track the objects of interest in the scene. The tracking models provide 2 main

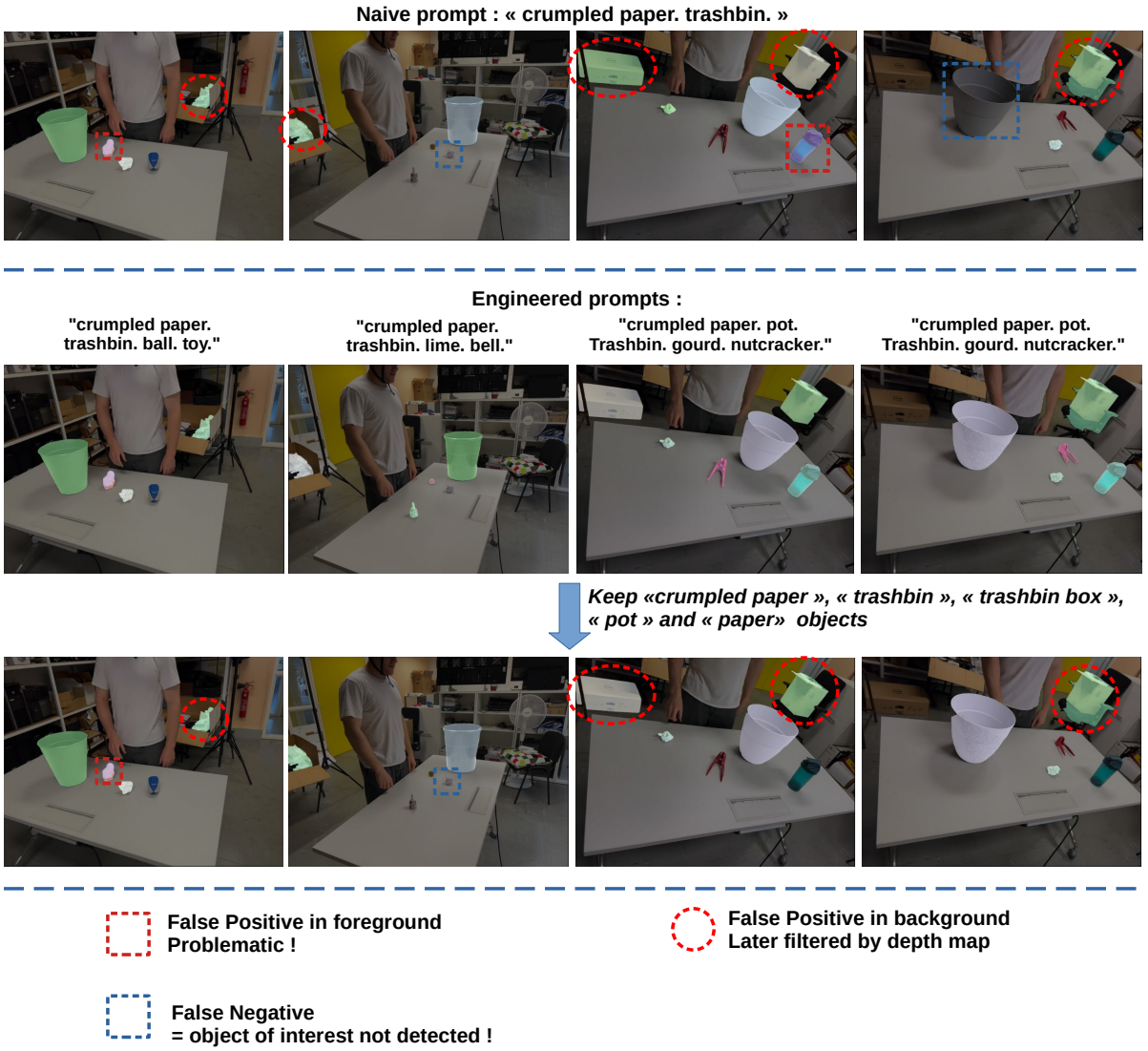


Figure 2.5 – Visualization of the segmentation masks obtained by the Grounded-SAM model on the first images of the demonstration dataset proposed in Chapter 8. It compares the results obtained with and without refining the prompt sent to the object detection model, respectively in the bottom and top parts of the figure. On the two right columns, we highlight examples improved by the prompt engineering, while on the two left columns, we highlight examples where the prompt engineering did not help.

benefits. First, the segmentation models detect the objects of interest in independent frames, but do not allow to re-identify the objects from one frame to another. For instance, if there are two "shoes" to segment in the scene, the segmentation model will detect the two shoes in the first frame, but will not be able to tell which shoe is which in the second frame. By definition, the

tracking models can be used to re-identify the objects from one frame to another. The second benefit is that the tracking models can run in real-time on the video frames, while the whole grounded-SAM pipeline takes several seconds per images, even on a high-end GPU (NVIDIA RTX A5000 for instance).

Luo et al. [114] categorized the tracking aproaches using 3 criterias. The initialization method, the processing mode (online or offline), and the type of output (stochastic or deterministic) of the tracker. The initialization is considered a detection-based method if the objects are first detected in all the frames and then linked into trajectories [10, 155]. On the contrary, the initialization is considered a detection-free method when a fixed number of objects is detected in the first frame, and subsequently re-identified in the following frames [23, 74, 203]. The processing mode is considered online if the tracker has only access to the past frames [187, 199, 7, 23], and offline if the tracker has access to both past and future frames [71, 73]. The type of output will basically be deterministic if the tracker outputs the same results for the same input, and stochastic if the tracker outputs different results for the same input, if the tracker is run multiple times.

Since the goal of the tracking model for our use case is to replace the object segmentation models in the video frames, we are interested in the detection-free models, and that can run online for further deployment on a robot. We further seek for tracking models that can run in real-time on a high-end GPU, like the NVIDIA RTX A5000.

The recent XMem [20] approach is particularly interesting, as it leverages multiple independant memory storages, a rapidly updated sensory memory, a high-resolution working memory, and a compact thus sustained long-term memory. Memory consolidation and forgetting mechanisms are used to keep useful information over time while preventing memory overload. This system allows to keep tracking performances stable over time, even in the presence of occlusions, which was a drawback of the previous tracking models like STCN [22] or even AOT [197] models.

Cutie [23] is an extension of the XMem model, but dissociate two categories of memory, a pixel feature memory, which is directly derived from XMem, and an object feature memory, which allows to integrate pixel features at an object level. Cutie runs at 45.5 FPS with the small model and 36.4 FPS with the large model on an Nvidia 32GB V100 GPU, while achieving state-of-the-art performances on all MOSE, DAVIS-17 and YouTubeVOS-2019 video object segmentation benchmarks, even compared to recent offline and detection-based methods like DEVA [24].

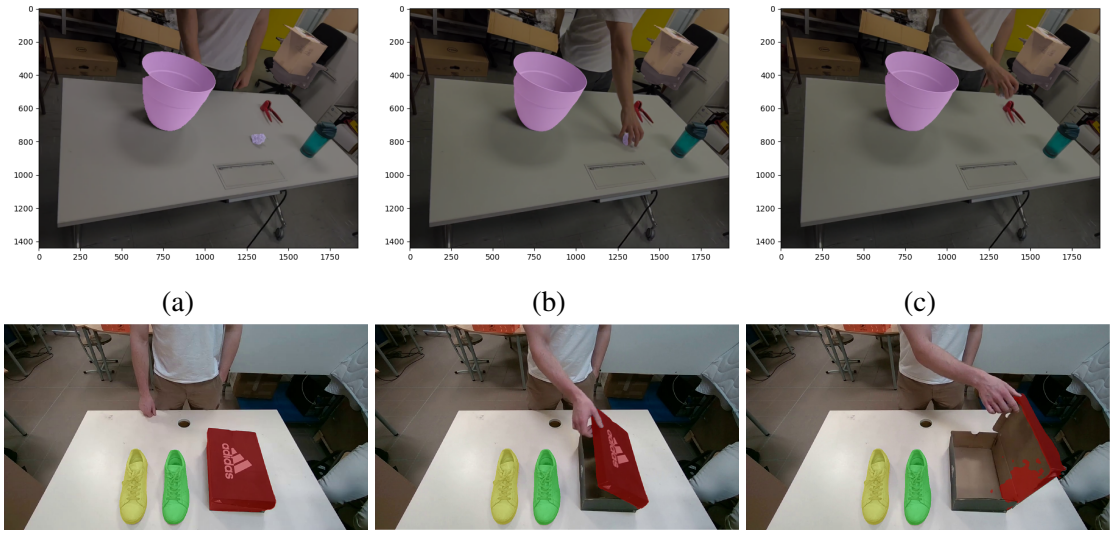
We ran the Cutie model on our set of demonstrations to annotate the objects of interest

in the video frames. The Cutie model was initialized with the segmentation masks obtained by the Grounded-SAM model for the 76% of the examples where the objects of interest were successfully annotated, and with the interactively annotated masks for the remaining 24% of the examples.

We further ran cutie on a set of 30 videos demonstrating a "shoes packaging" task, where the shoe-box is closed at the beginning of the video, and opened at the end of the video. We especially chose a shoe-box featuring significantly different colors between the inside and the ouside of the box, to evaluate the ability of the model to track on the object level, rather than simply relying on the pixel level.

The top part of the Fig. 2.6 shows the segmentation masks obtained by the Cutie model on the first set of demonstrations. We can see that Cutie is able to track the objects of interest over time, even when the objects are strongly occluded by other objects, like the "crumpled paper" almost completely occluded by the hand of the operator in the middle frame. Overall, it was properly tracking the objects of interest in the scene for all the 50 demomstrations of the first set.

The bottom part of the Fig. 2.6 shows the segmentation masks obtained by the Cutie model on the second set of demonstrations. In this case, Cutie shows its limitations since it looses the tracking on the inside of the shoe-box when opening it.



(a) (b) (c)

(d) (e) (f)

Figure 2.6

Overall, the Cutie model is suitable for tracking objects, even under occlusions, unless

strong deformations or state changes of the objects occur, like in the case of the shoe-box opening.

While this section of our pipeline allows to extract the pixels corresponding to the objects of interest in the demonstrations images, the next section will discuss the methods to project these 2D pixels to 3D points, and to further represent the scene in a 3D space.

2.4 Projecting 2D information to 3D

Given the pixel coordinates (u, v) and depth z of a point in the image, the 3D coordinates (x_c, y_c, z_c) of that point in the camera coordinate system can be computed by Eq. 2.7.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = K^{-1} \begin{bmatrix} z \cdot u \\ z \cdot v \\ z \end{bmatrix}, K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

K is the intrinsic parameter matrix of the camera, with f_x and f_y being the focal lengths of the camera, and c_x and c_y are the coordinates of the principal point of the camera, usually at the center of the image. Distortion parameters can also be added to the intrinsic parameter matrix.

The 3D coordinates of the point in the camera coordinate system can be further transformed to a world coordinate system, for instance the robot's base coordinate system during deployment, or in the coordinate system of the first frame of the video if we consider a moving camera. The transformation from the camera coordinate system to the world coordinate system is defined by Eq. 2.8.

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = E^{-1} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{2.8}$$

E is the extrinsic parameter matrix of the camera, which is composed of the rotation matrix R and the translation vector T of the camera with respect to the world coordinate system, i.e., $E = [R|T]$. This matrix can be obtained by calibrating the camera with respect to a known reference frame [170], like a checkerboard pattern, or by using the camera pose provided by the camera itself, if it is equipped with an IMU sensor.

The Fig. 2.3 illustrates the complete pipeline to project the 2D segmentations to 3D point clouds. Once the pixels corresponding to the objects of interest are extracted from the images,

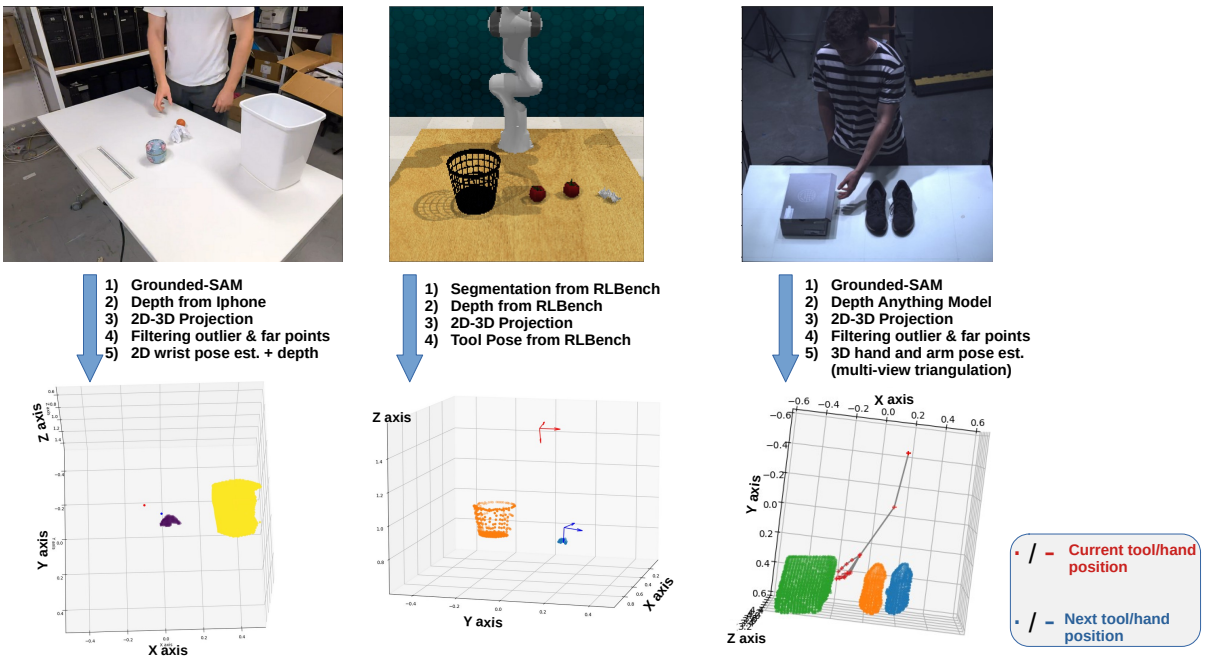


Figure 2.7 – Visualization of the point clouds obtained by projecting the 2D segmentations to 3D using depth sensor, depth estimation models or simulated depth maps from the RLBench simulation.

we can recover their depth from depth sensors, or from depth estimation models. The Annex C provides an overview of the depth estimation models and sensors that can be used to recover the depth maps of the scene and discusses their respective advantages and limitations. The point-clouds are then obtained by projecting these relevant pixels to 3D using Eq. 2.7. The points of which the depth is too far from the camera are filtered out, as they correspond to objects in the background. For instance, in the case of our demonstration dataset proposed in Chapter 8, points that are further than 2 meters from the camera are filtered out, as they correspond to objects outside of the table. An additionnal fitlering step is added to remove outlier points that might be due to noise in the depth maps, especially arround the edges of the objects when using the iphone depth sensor. In the case of our demonstration dataset, we typically use the radius outlier removal filter from the Open3D library [209] to remove the points that have few neighbors in a given sphere around them (less than 10 neighbors in a sphere of radius 0.05 meters in our case).

The Fig. 2.7 shows a few examples of the point clouds obtained by projecting the 2D segmentations to 3D using depth sensors (left side), depth estimation models (right side), or simulated depth maps from the RLBench simulation (middle). We also included the pose of either

the robot tool or the human hand in the point clouds. Strategies to obtain the Human pose are discussed in Part II. Note that when using the RLBench simulation, both the depth map and the segmentation masks are available without using any real sensors or models, which makes it a convenient way to generate large amount of data to validate our imitation learning algorithms before starting to work on actual human demonstrations.

The point-clouds presented in Fig. 2.7 provide a representation that is not impacted by the background of the scene, and where the operator (robot or human) can be abstracted as the position or the pose (position + rotation) of the operator's tool (human hand or robot gripper). Lastly, the 2D segmentations obtained in the previous section where highly dependent on the viewpoint, since the scale and position of the objects in the 2D images can vary depending on the viewpoint position relative to the scene. On the contrary, except for occlusions, the 3D information is more stable across when the viewpoint changes. The objects of interest are always at the same position in the 3D space, even when the viewpoint changes, if the coordinate system remains the same.

Keeping the same coordinate system accross environments and from the human demonstrations to the robot deployment setup can be challenging. It can be done by defining a fixed coordinate system in the environment, which can be difficult to maintain when the environmement changes. Alternatively, the pose of the tool (human hand or robot gripper) can be used as a reference to define the coordinate system, which would be common to both the human demonstrations and the robot deployment setup, and agnostic to the environment changes. Additionnally, another solution would be to collect the demonstrations in a moving camera setup, and to define the coordinate system as the camera's one. That way, the imitation learning models will be trained on data containing a multitude of viewpoints, and will be more robust to viewpoint changes during deployment. Those strategies will be further tested and discussed in Part III.

Lastly, note that the point cloud presented on the right side of the Fig. 2.7, which is obtained using depth maps generated using Depth-Anything [196] is one of the most accurate point clouds obtained on those data, but as discussed in Annex C, the Depth-Anything model is still far from consitently providing such accurate depth maps.

Depth sensors, despite being slightly more expensive than RGB cameras, are still far more reliable than depth estimation models for now, and are the most suitable solution to project the 2D segmentations to 3D point clouds for our use cases.

2.5 Conclusion

Pre-trained visual representations, such as the R3M or the VIP models have proven to outperform raw vision data when used as input to control models, as they allow to improve the model's sucess rate by several dozens of percents. However, such representations are not agnostic to the embodiment, the background environmemt or the viewpoint, which can limit the generalization of the control models trained from them. Since our goal is to train a model on human data and to deploy it on a robot, such pretrained representations are not suitable for our use case. The trained model would encounter out-of-distribution data during deployment, which would lead to a significant drop in the model's performances.

On the other hand, open-vocabulary object detection models coupled with interactive segmentation models, like the Grounded-SAM model, allow to extract the objects of interest in the scene from the human demonstrations, by simply specifying the objects categories to the model. The Grounded-SAM tested on our data suggests that it is still not robust enough to be implemented in a zero-shot manner, where no human intervention would be required at all to process the demonstrations. On our scenario featuring 50 demonstrations involving 9 categories of objects, among which we seek to detect 2 categories of interest, the Grounded-SAM model required human intervention in 24% of the cases to correctly detect the objects of interest, after optimizing the textual prompt sent to the model. However, with the pace of improvement in the open-vocabulary object detection models, we can expect to reach zero-shot annotation pipelines in the near future. In terms of deployment on a robot, human intervention to process the images obtained from the robot's camera is not possible. However, with a few annotated demonstrations, eventually even from the 76% of the demonstrations where the objects of interest were automatically detected, a traditionnal segmentation model could be retrained in a few-shot manner [141, 181, 85]. The few-shot model could even be used to annotate the remaining 24% of the demonstrations, thus eliminating the need for human intervention to process the demonstrations.

Additionnally, using a tracker model like Cutie, we were able to track the objects of interest in the scene over time, even when the objects were occluded. The Cutie model was able to track the objects of interest in the 50 demonstrations of our main scenario, but showed limitations on a second set of 30 demonstrations where the opening of a shoe-box was disturbing the tracker. This shows that the tracker still heavily relies on colors and textures rather than understanding the concepts of objects, despite the efforts made in that direction by the Cutie model. Cutie is however perfectly suited for our use case, both for annotaion and deployment on a robot, as it can run in real-time on a high-end GPU, and can track our objects of interest on our main

scenario.

Finally, by projecting the 2D segmentations to 3D, we can obtain a point-cloud of the objects of interest in the scene, which is less impacted by viewpoint changes than the 2D information, and which eliminates both the operator and the useless information from the background. The 3D point-cloud is extended to a 4D point-cloud by adding the object category as the fourth dimension to each point. The position or pose of the operator can be added to the point-cloud as well, but in a way that is invariant to the embodiment.

In Part III, we will compare the use of the VIP pretrained representation to the 4D point-cloud representation as input to the control models, with and without embodiment changes, environmemt changes (simulation vs real) and viewpoint changes.

The depth maps used to project the 2D segmentations to 3D can be acquired using depth sensors or deep learning models. Depth sensors can be costly, particularly in multi-view setups or challenging environments like outdoor environments which require more expensive sensors. However, they provide more reliable 3D data compared to current depth estimation models, which have shown limitations on our data, including depth inconsistencies, particularly with occlusions. We chose to use iphones to collect our demonstrations, as their are equipped with RGB sensors, depth sensors and IMU sensors, while being very easy to use and to carry around, which is necessary if we later want to collect larger datasets in various environments.

In addition to visual perception, tactile sensing offers valuable information for robotic manipulation. The next chapter will review existing tactile sensors and demonstrate how tactile data can be integrated with the 4D point-cloud representation to enhance the control models.

CHAPTER 3

TACTILE

Overview

Contents

Part of this chapter is based on research papers [143, 140, 144].

3.1 Introduction

Chapter 2 showed that vision-based 3D sensors, including Lidars and 3D cameras, enable the scanning of 3D scenes to recover dense 3D point-clouds from them, and how 2D annotation models can be used to isolate objects or regions of interest in these point-clouds. Additionally, the widespread deployment of such 3D sensors ranging from autonomous vehicles [17], mobile phones [182] and tablets[157], and industrial scenarios[95, 123], has led to a remarkable surge in the development of deep learning models designed for the direct processing of dense 3D point clouds. In particular, deep learning models have become increasingly accurate at localizing, recognizing, and delineating objects in the 3D scene from dense point clouds. However, extreme environments might not always permit the use of such vision-based sensors to acquire point cloud information. Fig 3.1 highlights a typical extreme use case where information is required for localizing and recognizing buried objects, e.g., buried mines, but where traditional vision-based sensors are not usable. Note that regarding buried mines, Ground Penetrating Radar (GPR) [166] is a crucial tool for rapid mine sweeping in conflict zones, however, manual tactile exploration remains the preferred method for post-conflict terrain cleaning, because of GPR limitations for certain types of mines or soil conditions [30]. Other extreme environment that might preclude the use of traditional vision-based sensors include those with poor lighting conditions, high levels of dust and troubled waters. In such scenarios, tactile perception, which is the ability to perceive objects/scenes through the sense of touch [113], can substitute for vision-based sensors in recovering point-cloud information from the target environment. In particular, in this work, we study how a system similar to the one highlighted in Fig. 3.1, can be used to localize and recognize objects in these kinds of extreme environments.

The system samples one 3D point at a time by driving a rigid stick in a given direction, until contact with an object is detected using a contact sensor [9]. A point-cloud can be further constructed by iteratively sampling points from the scene. While this system can only generate extremely sparse point clouds from the environment, it provides the opportunity to strategically select point sampling locations to effectively localize and recognize target objects. Compared with approaches described in Chapter 2, where points of interest were recovered from aquired dense point clouds, the strategies employed in this chapter allow to select the points of interest at the time of acquisition, and to collect only the points that are relevant to the task at hand. The strategy employed to control the iterative 3D point sampling, achieved by directing the tactile probe in specific directions, is the cornerstone of this Chapter. We will refer to this control strategy as the "exploration strategy", because it defines how the environment will be explored.

Figure 3.1 – In-house development of an autonomous probing system aimed at detecting, localizing and classifying underground objects.

In a preliminary work [139], we introduced an active point cloud acquisition framework for 3D object recognition. The exploration strategy was denoted as "active" because each 3D point sampling was adaptively determined based on the context provided by the 3D points that had already been collected. This contrasted with "passive" exploration strategies, where the sequence of 3D points was always collected following the same sampling routine. The proposed framework relied on a deep learning model, that simultaneously learned an exploration strategy to control the robot and a 3D point cloud classifier, both aimed at maximizing the objects recognition performance, which corresponds to the classifier accuracy. This way, the training of the exploration strategy was guided by the classification performance, ensuring that each collected point provided the most information for the 3D recognition task.

The framework was trained using a Reinforcement Learning (RL) algorithm, with the objective of maximizing two reward terms: the classification reward term, and the exploration reward term. The classification reward term provided a positive reward in the case of successful object classification. The exploration reward term was introduced to enhance the training of the sampling strategy. Its goal was to reduce "Missing Parts", which occur when the collected point-cloud doesn't uniformly cover the target object, and "Noisy Points", two types of disturbances that have been shown to affect state-of-the-art point cloud classifiers [164]. The exploration term encouraged the model to explore target objects broadly to avoid missing parts, while also ensuring as many samplings as possible on the object, as a sampling that misses the object results in a noisy point.

In this work, we build upon the original framework and propose the following novel contributions. i) We provide a detailed analysis of the contributions of the different reward terms (exploration and classification) to the overall framework's performance. ii) We introduce an

additional reward term that estimates the object's position on the workspace and further investigate its impact on the classifier's performance. iii) We compare the behavior of hard-coded passive sampling strategies with the active sampling strategies learned by our framework under both simplified and more realistic scenarios, including realistic objects and varying object positions in the workspace. Our framework demonstrates higher classification accuracy compared to previous active exploration strategies and similar or better performance than a hard-coded "Homogeneous Exploration" strategy that uniformly samples the workspace. It achieves particularly strong classification results when all reward terms are used together.

This chapter starts by reviewing related works in Section 3.2. Section 3.3 describes our approach in details, as well as the in-house simulation developed to train and benchmark our models. Section 3.4 provides experimental setup details and results. Finally, while the use case related to this chapter is not directly related to the imitation learning framework ultimately proposed in this thesis, the approach investiguated for tactile exploration has 2 benefits that are relevant to big picture in which this thesis is framed. Firstly, the proposed approach shows how the dense visual representation proposed in Chapter 2 can be enriched with sparse targeted tactile information, since both frameworks' scene representations take the form of 4-Dimensional point-clouds: 3 spatial dimensions for the 3D coordinates of the points, and a 4th dimension labeling the points, either with the object category from vision, or with tactile or contact feedback from the tactile exploration. Secondly, while the proposed framework shows the potential of reinforcement learning to learn reusable robotics primitives (tactile exploration primitive in this case), it also highlights the unrealistic amount of interractions required if one would like to rely solely on reinforcement learning to learn multiple tasks from scratch, especially on real robots. These two points will be further discussed in the Section **??**.

3.2 Related Works

In this section, we begin by surveying the existing landscape of solutions for object localization and recognition within dense fixed point clouds. Subsequently, we explore the realm of tactile perception solutions, which enable the active recovery of sparse 3D information from a given scene. In this context, we also delve into the integration strategies for training the previously mentioned object recognition and localization models within the framework of active exploration.

3.2.1 Classifying and Locating objects from point clouds

Deep learning architectures proposed for 3D point cloud classification and segmentation [57] have exploded during the last decade. The architectures used for point cloud processing can be mainly divided into 3 categories [57]: graph-based methods, convolution-based methods and point-wise Multi-Layer-Perceptrons (MLP) methods.

Graph-based methods represent the point cloud as a graph, where each node represents a point, and where edges are generated based on the points neighborhood. Feature learning can further happen either in the spatial domain, e.g., using graph-convolutional neural networks [183, 16] or in the spectral domain [165].

Convolution-based methods apply convolution kernels to the point clouds based on the spatial distribution of the points [108, 184, 109].

The PointNet architecture [128] introduced the notion of point-wise MLP methods. PointNet processes point-wise features independently using MLP and fuses them using a permutation invariant function, e.g., a max-pooling layer. The permutation invariance guarantees that permuting the points in the point cloud doesn't affect the prediction results. Dozens of architectures were proposed based on PointNet, including PointNet++ [129], which allows to capture finer geometric strutures and PointWeb [206] that leverages local neighborhood context to improve point features. Sun et al.[161] further proposed the promising Context Awareness (CA) and Self Attention Context Awareness (SACA) operations, to incorporate global context information about the point cloud into the individual points-wise features. While they applied it to point cloud generation, these operations can be also be applied on classification models. PointNet classifiers have recently been proven to be particularly robust to sparse and noisy point clouds [164] compared to other baselines, which is particularly interesting for our use case. The sparsest point cloud evaluated by Taghanaki et al. [164] however still contained 128 points per object for classification, which exceeds the number of points considered in our use case.

Neverthless, most of the existing architectures are trained on pre-collected datasets, which are constructed using either vision based sensors such as 3D cameras [27, 46, 156] or lidars [46], or synthetic data [185]. These datasets tipically contain thousands of points per object or per scene.

Regarding object localization, existing architectures designed for dense point clouds operate in 2 steps : (1) they segment sub-parts/objects within the dense point cloud [61] or utilize mechanisms to focus on relevant smaller regions of a large point cloud [124]; and (2) they predict the refined coordinates of target objects, either through an additional architecture [61], or dedicated prediction heads [124, 208]. In our case, as the exploration strategy's goal is to directly focus on

the target object, we can skip the first step and equip our architecture with both a classification and a pose estimation head.

3.2.2 Tactile Perception

Tactile Perception [113] is frequently utilized to assist robots in gaining a better understanding of their surroundings. It involves interpreting and representing tactile sensory information to observe the properties of objects. Luo et al.[113] categorize tactile sensors in 3 categories: (1) Single-point contact sensors, that can detect object-sensor contacts and eventually recover the 3D position of these contact points [75]. (2) High spatial resolution tactile sensors, including tactile arrays [167], artificial skins [8] and vision-based fingertip sensors [96], that can be easily embedded on robots grippers. (3) Large-area tactile sensor [70] that are designed to cover large, curved body parts of robots. In this study, we simulate a robust type of single-point contact sensor, similar to the one illustrated in Fig 3.1. This sensor allows us to retrieve 3D points, even in challenging environments, like when examining buried objects.

Many active tactile perception frameworks have been proposed to effectively explore and characterize an environment from touch. Gaussian processes were primarily used to model the explored space and further focus samplings on specific areas, depending on the uncertainty on the current environment modeling. Jamali et al. [78] utilized Gaussian processes in their work with fingertip sensors, and a similar approach was employed by Kaboli et al.[84] with multi-modal skin sensor to characterize objects in the target environment. Gaussian Processes were also applied to robots to simultaneously explore tactile information and refine grasp precision when dealing with unknown objects [38]. More recently, tactile exploration strategies learned through reinforcement learning (RL) algorithms, as discussed in studies such as [41, 144], have been proposed. These strategies aim to simultaneously learn an object exploration strategy and an object classifier, allowing both tasks to mutually enhance each other during the training phase. Further RL reward functions were also investigated [139] to improve the efficiency of the exploration strategy, in order to ultimately enhance the classifier's performance.

3.3 Approach

We consider a simulated workspace, where unknown objects need to be quickly identified and localized, without access to a camera. The setup is depicted in Fig 3.2. A poly-articulated robot is equipped with a laser-based distance sensor that allows to simulate the acquisition of

3D points in the workspace from a single-point contact sensor, as shown in the Fig 3.1. The robot has sufficient degrees of freedom to acquire 3D points throughout the workspace. An RL agent is trained to sample the points that are the most informative for the classification and pose estimation tasks. The setup is simulated to facilitate the training and validation of the proposed solution. Two different sets of objects can be loaded into the simulation: the realistic and the geometric object sets. Detailed descriptions of these sets can be found in section . 3.4.1.

In this section, Part A first explains the simulated setup, which is similar to the one proposed in [139], but emphasizes novel features included to enable the evaluation of more realistic use cases. Secondly, it discusses the challenges associated to a deployment on a real setup, similar to the one presented in Fig 3.2. Part B explains the framework proposed to jointly learn active exploration, localization and classification of 3D objects positioned in the simulated workspace. Part C finally provides details about the employed reinforcement algorithm and the specific reward terms designed to train the framework.

3.3.1 Active sampling setup

Simulated setup

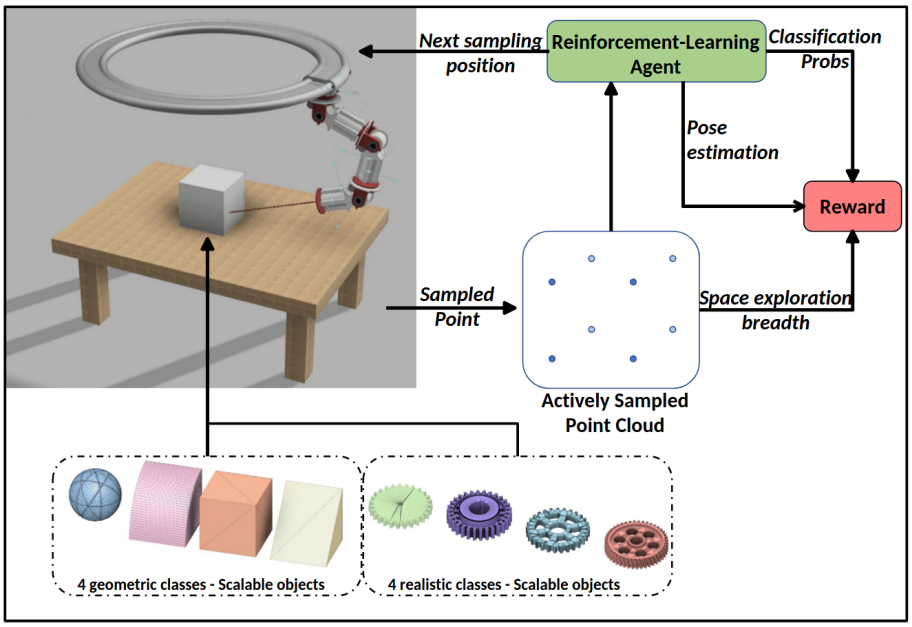
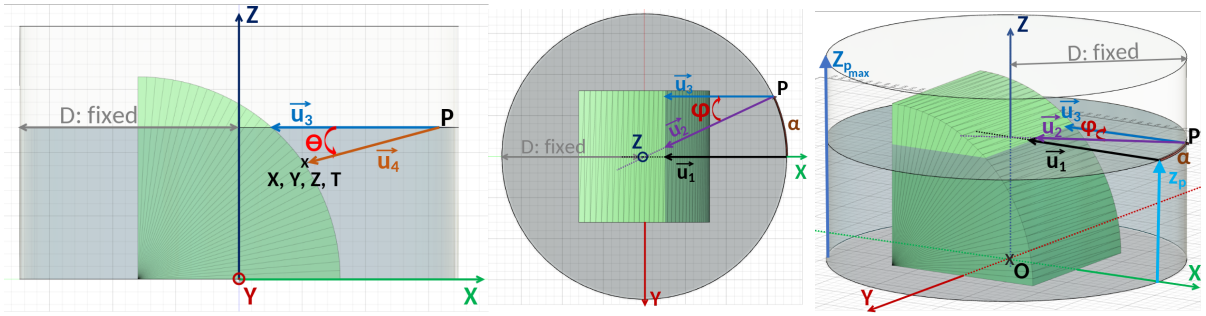


Figure 3.2 – Overview of the RL-based approach.

The simulator consists of a 3D environment where 3D objects can be loaded at the origin "O", as depicted in Fig 3.3. The Z=0 plane in Fig 3.3 is equivalent to the top of the table in

Fig 3.2. To facilitate the evaluation of the proposed models in more realistic scenarios, where objects could be randomly placed in the workspace, we can also introduce random offsets for the objects along both X and Y axes. The object position offsets along X and Y axis are respectively denoted as X_o and Y_o. The objects can be rotated around Z axis. Note that if objects are both rotated and shifted with position offsets, the rotation is executed prior to the positional shift. The objects can further be re-scaled when they are loaded in the environment. 3D points can be acquired from the environment by activating the simulated laser sensor. Therefore, the 3D workspace and the loaded object can be actively explored by instructing the simulator to perform laser acquisition sequences. This setup allows active exploration of the workspace, where a model iteratively controls the position and orientation of the laser sensor, guided by previous acquisitions.



(a) (b) (c)

Figure 3.3 – Illustration of an object loaded in the simulator from 3 views ((a) Side; (b) Top; (c) 3D).

The Fig 3.3 highlights the inputs/outputs and the operation of the simulator. The simulator works by using input values that determine the laser's position and orientation to control the laser acquisition system. It then records the 3D point that the laser reaches under this control.

More formally, the laser sensor can traverse the surface of a cylinder that encompasses the whole workspace. The diameter of the cylinder, noted as D, and the position of the laser on the cylinder, noted P, are both depicted in Fig 3.3(c). P is defined by two parameters: $Z_p \in [0, Z_{p_{max}}]$ and $\alpha \in [0, 360°]$, prior one denotes the height of the probe on the cylinder, and later one denotes the revolution angle around the cylinder. As depicted in Fig 3.3(c), vector $\overrightarrow{u_1}$ and $\overrightarrow{u_2}$ represent two laser propagation routes that run parallel to the ground and towards Z axis, respectively before and after applying revolution angle α.

Moreover, the orientation of the laser is determined by two variables: φ and θ, which represent the "yaw angle" and the "pitch angle", respectively. These angles allow the laser to deviate

from its default propagation. φ allows to deviate from default propagation towards Z axis, while θ allows to deviate from a propagation parallel to the ground. Concretely, vector $\overrightarrow{u_3}$ on Fig 3.3 represents the laser propagation after successively applying Z_p, α and φ transformations in this order. Vector $\overrightarrow{u_4}$ represents the laser propagation after applying θ transformation to $\overrightarrow{u_3}$.

When the simulator receives a query set (Z_p, α, φ, θ), it propagates the laser with this configuration until it either touches the object or reaches the maximum propagation distance. It then computes the 3D point (X, Y, Z) reached with this configuration. The computed 3D point is returned, accompanied by a boolean value, T, indicating whether the laser actually made contact with an object or not. In the following part, we refer to this querying operation as "sampling" 3D points in the environment.

Towards a real setup

Since we began developing the hardware for our tactile exploration solutions, we've gathered insights to ease the transfer to a real setup. Unlike computer vision, our tactile strategy focuses on elements common to both simulation and reality, ignoring environment-specific backgrounds. The remaining gaps between simulation and reality stem from measurement errors and robotic hardware reachability limitations.

A sim-to-real transfer procedure is needed to model measurement error on the real hardware, incorporating uncertainties in 3D position due to factors like robot precision, repeatability, positioning error, or probe deformation (for solid mediums). It should also incorporates false positive rates in touch detection. These uncertainties must be integrated into the simulator and can be overestimated as a domain randomization strategy [19]. Uncertainties calibration ensures the simulation reflects real-world conditions, aiding generalization of trained models to real setups. For instance, with our real hardware, the average measurement error was estimated at about 2mm, peaking at 5mm, limiting target objects to sizes greater than 1cm. With our 6-degree-of-freedom robot, balancing parameters D and α (Section 3.3.1, Fig 3.3) is necessary. Our hardware restricted α to $[-\pi/2, \pi/2]$, restricting to probe only on surfaces facing the robot, which enabled exploration of up to approximately 1m wide surfaces.

Controlling the real robot based on the output of the trained model is straightforward. The parameters P and $\overrightarrow{u_4}$ represent the target position and orientation of the tool, respectively, which can be reached using the robot's inverse kinematics. The operation of the contact sensor at this position is managed by a dedicated controller, which moves the rigid stick until contact is made or the maximum range is reached.

3.3.2 Framework architecture

Overview of the proposed framework

We propose a framework capable of sequentially predicting sampling configurations, so as to progressively construct a 3D point cloud. Consequently, this set of points will be used to classify and locate the 3D object in the workspace. The left side of Fig 3.4 gives an overview of the proposed framework. It iteratively predicts new points to sample based on the previously collected points. At each prediction step, the framework further predicts the class and location of the object using the acquired points. This framework is constructed with a deep neural network that can be trained end-to-end with an off-Policy RL algorithm. Importantly, the framework receives immediate rewards after each prediction, and these rewards are determined based on the classification and localization performances achieved using the current acquired point cloud. This way, each predicted sampling must provide the most relevant information for the classification and localization tasks. As a result, the exploration strategy is learned to optimize the success of 3D object recognition and localization. As illustrated in the left side of Fig 3.4, the proposed framework can be divided into 5 parts. (1) An action prediction module, that predicts the configuration of the next samplings from the latent representation of the already collected points; (2) a point cloud storage, aiming to archive the 3D points collected by the samplings accumulated throughout all preceding steps; (3) a point cloud feature extractor, that encodes the accumulated point cloud from all the previous steps into a permutation invariant latent representation; (4) the classifier, that predicts the class of the object placed in the workspace from the latent representation of the accumulated point cloud; (5) the pose estimation module, that predicts the position (X_o, Y_o) of the object in the workspace.

Predicting series of samplings

As illustrated on Fig 3.4 (left), it's important to acknowledge that the framework consistently predicts three samplings simultaneously. This choice represents a deliberate trade-off between two key considerations. Firstly, the framework receives a reward after each iteration. When a fixed total number of samplings is considered, predicting fewer samplings at once increases the frequency of reward reception, thus facilitating reinforcement learning optimization. However, predicting fewer samplings at once also leads to higher variation in point cloud densities supplied to the classifier and pose estimator. This variation can introduce challenges in their optimization processes, potentially increasing the noise in the associated reward terms. Moreover, when too few points are added to the point cloud at once, it may not significantly enhance

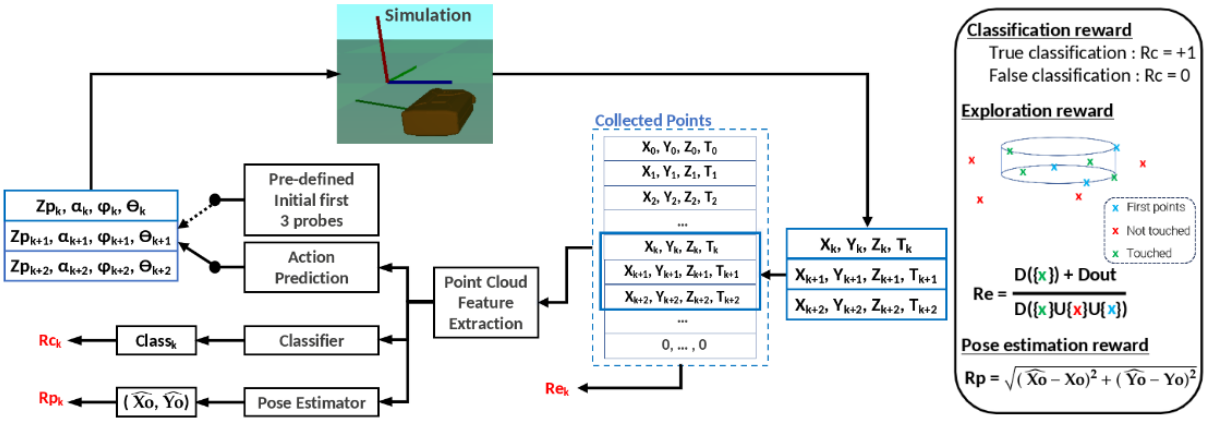


Figure 3.4 – Overview of the proposed policy network (left) and proposed reward function terms (right).

classification and localization performance.

At the very beginning of the exploration, the point cloud storage is filled with zero values, which represents an empty point cloud. The exploration starts with 3 hard-coded sampling configurations, used to initialize the point cloud: ($Z_{p_{0:2}}$, $\alpha_{0:2}$, $\varphi_{0:2}$, $\theta_{0:2}$). The 3 initial samplings are chosen to cover uniformly the ranges of α and Z_p parameters, with θ and φ set to 0. This simple initialization provides an overview of the workspace to the model to start making its first predictions. After initialization, the point cloud storage is then updated with the 3 initially sampled points ($X_{0:2}$, $Y_{0:2}$, $Z_{0:2}$, $T_{0:2}$). Consequently, for each step $k \in \{3n, n \in [1, N]\}$, the framework (1) tries to classify and to localize the object; (2) predicts the next three samplings ($Z_{p_{k:k+2}}$, $\alpha_{k:k+2}$, $\varphi_{k:k+2}$, $\theta_{k:k+2}$) and requests them to the simulator, and (3) stores the three newly acquired points ($X_{k:k+2}$, $Y_{k:k+2}$, $Z_{k:k+2}$, $T_{k:k+2}$) in the point cloud storage. A total $3 \star N$ points are sampled from the object through the execution of N acquisition steps. Details of various parameters can be found in the experiments section.

Point cloud feature extraction

The point cloud feature extraction module is described in Fig 3.5. It is inspired from the "Per-point Context Aware Representation" proposed in [144]. The input of the module is the point cloud, that contains $3 \star N$ points. As described above, each point has a dimension of 4, prior 3 values representing the 3D point reached during the corresponding sampling, the last one is a boolean indicating if an object was actually touched during the sampling. These points are independently processed by a set of MLPs sharing the same weights, in the PointNet [128]

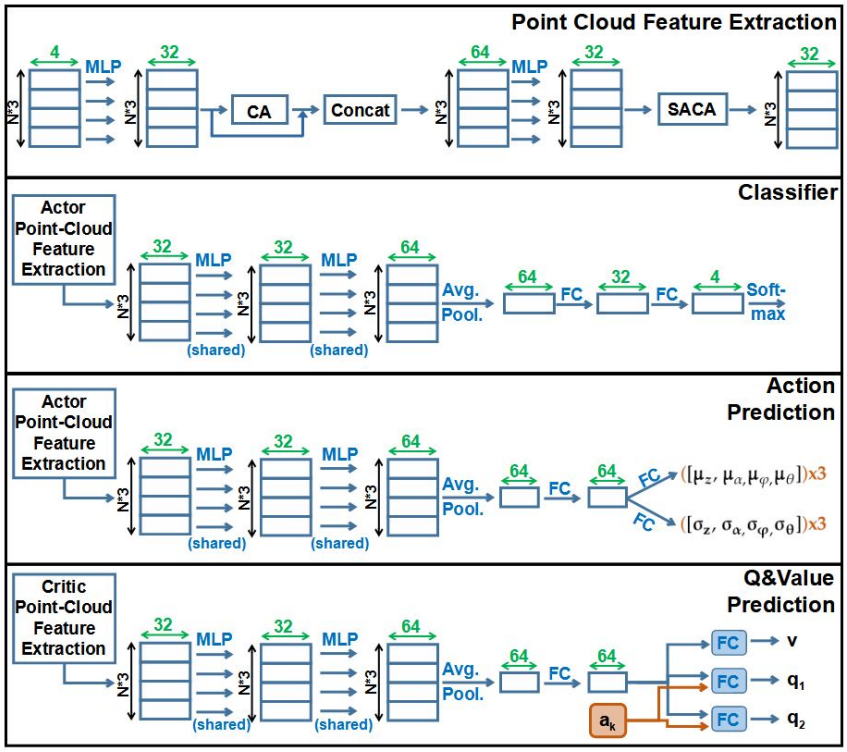


Figure 3.5 – Point cloud feature extraction, classifier, action prediction and Q&Value prediction architectures. Green values represents feature sizes.

fashion. In the following, we refer to this operation as "per-point-MLP" operation. The features obtained from the first per-point-MLP are subsequently sent to a "Context Aware" operation (CA), overlapped by a Residual connection. The CA was introduced by Sun et al. [161] with the PointGrow architecture. The CA takes $3 \star N$ point cloud features as input, and outputs $3 \star N$ embeddings. Each of these embeddings reflect the global context of the previously sampled points so far. Concretely, for each point in the cloud, CA returns the average of the features of the points collected previously. The resulting embeddings are finally sent to an additional per-point-MLP, followed by a "Self-Attention Context Awareness" operation (SACA-A). The SACA-A was also introduced in the PointGrow [161]. While the CA aggregates features of the previously collected points with a fixed average pooling, the SACA-A aggregates them with a weighted average pooling. The weights assigned to the previously collected points are predicted by a dedicated learnt sub-module.This sub-module itself is made of a CA operation overlapped by a Residual connection, followed by a per-point MLP. For both CA and SACA-A operations, we generally followed the configuration of PointGrow [161], except the feature sizes predicted by each of our MLP layers are set to 1/4 of original feature sizes. The reason behind this feature

size reduction is the sparsity of the data used in our case. Additionally, we also added batch normalization after each MLP layer.

Classification, action prediction and pose estimation

The classifier and the action prediction module, also detailed on Fig 3.5, respectively infer the object class probabilities and the next samplings configurations from the current latent representation of the accumulated point cloud so far. Both the classifier and the action prediction module follow similar structures, except for the last layers. They are both made of two successive per-point-MLP layers, followed by an average pooling layer that aggregates information from the whole point cloud. Furthermore, several fully connected layers (FC) are used to reduce the aggregated features to the desired output dimensions. The very last layers structures will differ between classifier and action prediction branches. On one hand, the classifier predicts deterministic class probabilities through its softmax layer. On the other hand, the action prediction module is stochastic [1]. It predicts parameters μ and σ of a Gaussian distribution for each component of the sampling configuration, instead of deterministically predicting their values :

$$\begin{aligned} z_k &\backsim \mathcal{N}(\mu_{z_k}, \sigma^2_{z_k}), \quad \varphi_k \backsim \mathcal{N}(\mu_{\varphi_k}, \sigma^2_{\varphi_k}) \\ \alpha_k &\backsim \mathcal{N}(\mu_{\alpha_k}, \sigma^2_{\alpha_k}), \quad \theta_k \backsim \mathcal{N}(\mu_{\theta_k}, \sigma^2_{\theta_k}) \end{aligned} \tag{3.1}$$

Since the proposed framework explores the workspace 3 points by 3 points, the action prediction module predicts 3 sampling configurations at once, resulting in 3 sets of Gaussian distribution parameters predicted at once.

When the pose estimation module is employed, additional two sets of Gaussian Distribution parameters are predicted by the action prediction module, (μ_{Xo}, σ_{Xo}) and (μ_{Yo}, σ_{Yo}), so that :

$$X_{o_k} \backsim \mathcal{N}(\mu_{X_{o_k}}, \sigma^2_{X_{o_k}}), \quad Y_{o_k} \backsim \mathcal{N}(\mu_{Y_{o_k}}, \sigma^2_{Y_{o_k}}) \tag{3.2}$$

By doing so, the object position is modeled as a Gaussian noise added to the ideal position $(X_o = 0, Y_o = 0)$.

Additional module dedicated to RL training

The lower section of Fig 3.5 introduces an additional module dedicated to predicting Q-values (q_1 and q_2) and the state-value (v). These predictions play a crucial role in optimizing

our architecture through the Soft Actor-Critic (SAC) reinforcement learning algorithm [60], detailed in Section 3.3.3. Here, a_k denotes predictions encompassing object class, object pose, and next sampling configuration. Conceptually, the Q-values are trained to estimate the advantage of these predictions with respect to the previously gathered point cloud, while the state-value assesses the quality of this point cloud for classification and localization tasks. The module's input is derived from a dedicated point cloud feature extraction module, which is structured identically to the one used for classification, action prediction, and pose estimation, but without weight sharing.

3.3.3 RL algorithm and reward function terms

The proposed framework is trained using Soft Actor Critic (SAC) [60], an off-policy RL algorithm. SAC trains a policy π, the decision making agent, to take actions a_t from current state s_t, so that it maximizes the objective $J(\pi)$ defined by Equation 3.3.

$$J(\pi) = \underset{\tau \backsim \pi}{\mathbb{E}} \left[\sum_{t=0}^{T} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \Big) \right] \tag{3.3}$$

The objective represents the sum of rewards, denoted as $R(s_t, a_t, s_{t+1})$,that the agent receives for its decision-making throughout the whole sequence of visited states, spanning from step 0 to step T. The entropy term, $\mathcal{H}(\pi(\cdot|s_t))$, is added to encourage exploration of diverse policy strategies. The discount factor $\gamma < 1$ is an impatience term, favoring immediate rewards over late ones. In our use case, the state s_t is defined as the point cloud collected at time t. The action a_t is made of 3 components (1) the next 3 samplings configurations; (2) the classification probabilities; and (3) the object pose estimation $(\hat{X}_o, \hat{Y}_o)$. Equation 3.4 defines the reward as the weighted sum of three reward terms, namely, the classification reward r_c, the exploration reward r_e and the pose estimation reward r_p. The weights β_{r_c}, β_{r_e} and β_{r_p} are part of hyperparameters of the framework.

$$R(s_t, a_t, s_{t+1}) = \beta_{r_c}.r_c^t + \beta_{r_e}.r_e^t + \beta_{r_p}.r_p^t \tag{3.4}$$

r_c represents the classification performances of the framework. After each acquisition step $t \in [1, N]$, the model tries to classify the object. A successful classification is denoted as $r_c = 1$, while a failed one results in $r_c = 0$. This process serves 2 purposes: providing supervision for classifier training, and ensuring the action prediction module adheres to a sampling strategy that

facilitates the classification.

The function of r_e is to (1) ensure consistency in the exploration strategy during training, and (2) to enhance the exploration quality by penalizing missed parts and noisy points (samplings that missed the object). Formally, consider B_t as the set of three initial points, G_t as the set of points that made contact with the object, and R_t as the set of points that failed to touch the object. These sets are visually represented as Blue, Green, and Red points on the right side of Figure 3.4. We refer $D(A)$ which is defined by Equation 3.5, as the sum of inter-points distances of a set of points A.

$$D(A) = \sum_{i=0}^{|A|} \sum_{j=0}^{i-1} d_e(A_i, A_j), \tag{3.5}$$

where d_e is the euclidean distance, and $|A|$ represents the number of points in A. On this basis, r_e is defined by Equation 3.6. The left part of the equation encourages to sample points on the object of interest while simultaneously maximizing the distance between them, thereby broadening the exploration.

$$r_e^t = \frac{D(G_t)}{D(G_t \cup R_t \cup B_t)} + D_{out_t} \tag{3.6}$$

However, in certain cases. the samplings that failed to capture the object could hold relevance for the classification task. In particular, samplings that almost touched, or made slight contact with the object, could be valuable in outlining its boundaries. r_e is thus enhanced with an additional term, namely D_{out}, that gives credit to missed samplings regarding the reward they would obtain if they reached the closest point in $(G_t \cup B_t)$. Noteworthy that the reward is normalized by their distance to this closest point. Formally, each sampling is represented as a line segment, denoted as (d_k), characterized by $(P_k, \overrightarrow{u_{4_k}})$. Here, $\overrightarrow{u_{4_t}}$ represents the direction vector of the sampling line, and P_k represents a point on the this line, as illustrated in Fig 3.3. The distance d_c between a sampling (d_k) and a point A is defined by Equation 3.7.

$$d_c(A, (d_k)) = \frac{||\overrightarrow{PA} \wedge \overrightarrow{u_{4_k}}||}{||\overrightarrow{u_{4_k}}||} \tag{3.7}$$

Therefore, the closest point to a sampling (d_k) in $(G_t \cup B_t)$, denoted as $A_{min}((d_k))$, is

defined by Equation 3.8.

$$A_{min}((d_k)) = \underset{A \in (G_t \cup B_t)}{\operatorname{argmin}} (d_c(A, (d_k)) \tag{3.8}$$

D_{out_t} is finally defined by Equation 3.9, where M_{R_t} is the set of missed samplings that resulted in R_t.

$$D_{out_t} = \sum_{(d_i) \in M_{R_t}} \sum_{A_j \in (G_t \cup B_t)} \frac{d_e(A_{min}((d_i)), A_j)}{d_c(A_{min}((d_i)), (d_i))}, \tag{3.9}$$

Finally, as defined on the right side of Fig 3.4, r_p quantifies the Root Mean Square Error (RMSE) between the predicted and the actual pose of the object within the workspace.

3.4 Experiments

The experiments are divided into 6 parts. Part A describes the general experimental settings, common to all the experiments conducted. Part B recalls the main results of our preliminary work [139] and points out the limits of its experimental validations. Part C and D bring complementary studies on the initial results[139]. Part E investigates the usability of the framework in more realistic scenarios. Part F evaluates the novel pose estimator performances and its contributions to the overall framework.

3.4.1 Experimental settings

In our preliminary work [139], we built a dataset with four 3D geometric objects to conduct experiments. In this study, we have extended this dataset with a second set of "realistic objects". These objects are characterized by greater complexity and a closer resemblance to each other, thus challenging the classification task even further. The geometric and realistic sets are respectively depicted in Fig 3.6 and Fig 3.7.

For all the following experiments, during the framework training, the objects are randomly chosen among the used set, and placed with a random rotation R_z around the Z axis, which is illustrated in Fig 3.3. All experiments consider $N = 4$ iterative acquisition steps, leading to a total of 12 points sampled at the end of the exploration sequence, which is similar to [144]. The proposed RL-based framework is trained using the stable-baselines-1 [72] SAC implementation

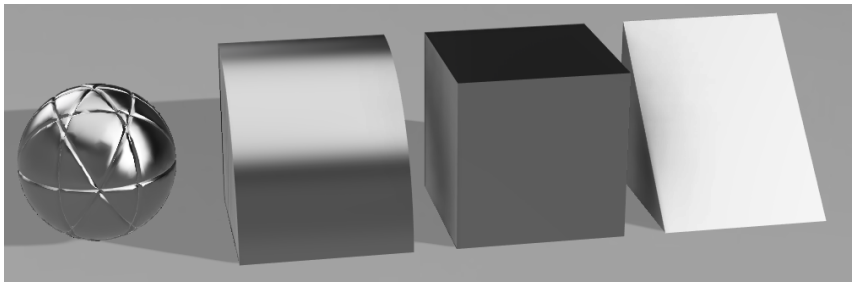


Figure 3.6 – Geometric objects : Sphere, Quarter-round, Cube and Triangle.

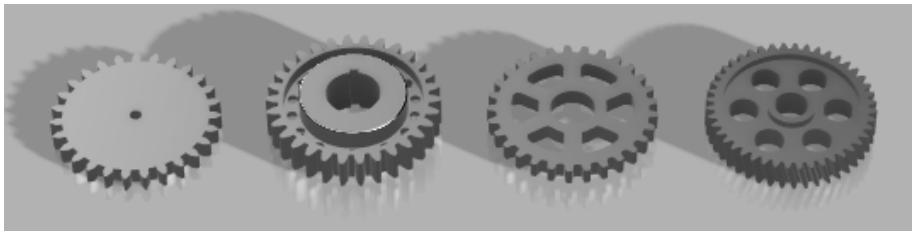


Figure 3.7 – Realistic objects : Gearwheel variations

with default hyperparameters except for the batch size, which was set to 512. In particular, $\gamma = 0.99$, learning-rate=0.0003, and α='auto', which means it is learnt during training. All RL-based trainings ran 600k steps, and were evaluated by computing the accuracy of the best model obtained during training on 1400 objects per class. Results corresponding to our framework are denoted as MTR-RL in result tables. It stands for Multi-Term-Reward Reinforcement Learning. Additionally, Table 3.1 details the simulation configuration and action ranges used in all the experiments. Finally, in experiments involving random object scaling, the loaded objects are rescaled by a randomly selected factor within the range of $[0.8, 1]$.

Table 3.1 – Simulation parameters and action ranges (depicted in Fig 3.3). “Geom." and "Real." indicate parameter values respectively for the Geometric and the Realistic object sets.

	min	max
Z_p (Geom.)	1	100
Z_p (Real.)	0	50
α	0°	360°
φ	-30°	30°
θ	-20°	20°
D	100mm	
Probe range	200mm	

We compared our framework to both active-sampling-based and passive-sampling-based exploration methods for 3D object recognition. In the case of the active-sampling strategies, we compare to the state-of-the-art LSTM-based tactile exploration model [41], which we adapted to our use case. We denote it as "LSTM" model. For the passive-sampling strategies, we feed point clouds into a state-of-the-art classifier. These point clouds are sampled from the simulator using hard-coded sampling strategies. We employed two passive sampling strategies for the comparisons: (1) "Rand. Explo + PointNet", in which point clouds are randomly sampled from the simulator, and (2) "Hom. Explo + PointNet", in which points are homogeneously sampled on the 3D object. For the homogeneous sampling, θ and φ were set to 0, and Z_p and α were uniformly chosen between their minimum and maximum values. The idea of the homogeneous sampling is to provide an exploration strategy that can scan the whole object to avoid "missing parts", known to affect point cloud classifiers. It is worth noting that the homogeneous sampling is an ideal scenario, which is possible in simulation. This is because the position of the object is known in advance, and the object is well positioned to facilitate such procedure, given the flexibility of the robotic arm movement around the object. On the contrary, our RL-based framework can be trained from scratch for any kind of scenario, adapting and learning an optimal exploration strategy tailored to each specific situation.

The proposed active-sampling based framework classifies the object after each 3 collected points, and for a total of 12 points. It means that the classification is successively made from 3, 6, 9, and 12 points. For fair comparisons, we also evaluated the passive-sampling strategies on 3, 6, 9 and 12 points densities. Concretely, we used the hard-coded sampling strategies to collect a point cloud dataset for each density and trained independent PointNet models [128] on each of these. PointNet has been chosen as it has been proven to be robust to sparse and noisy point clouds [164].

In contrast, our framework only uses one single classifier for all point cloud densities (3, 6, 9, and 12 points). Moreover, the classification utilizes a shared latent representation that is common to the classifier, the pose estimation head, and the action prediction head. This is not fair to compare our framework's classification results to the independent PointNet models trained for "Rand. Explo + PointNet" and "Hom. Explo + PointNet" experiments, since these models are density specific and fully dedicated to the classification task. We thus conducted further experiments to fairly compare the quality of the exploration strategy learned by our proposed framework. Similarly to passive models evaluation, we collected a point cloud dataset for each density, but using our trained active-sampling framework, and further trained independent PointNet models on each of these datasets. In experiments results, this scenario coupling our

framework with independent PointNet classifiers is denoted "MTR-RL + PointNet".

Every PointNet models, for all "Rand. Explo + PointNet", "Hom. Explo + PointNet" and "MTR-RL + PointNet" approaches, were trained on 22000 training point clouds balanced between the 4 classes, and on 80 epochs. All PointNet models were also evaluated by computing the accuracy of the best version obtained during training over 1400 objects per class.

3.4.2 Comparison to prior works under ideal settings

In our preliminary study [139], the proposed framework was evaluated on an ideal setup, where the position of the objects was frozen to $(X_o = 0, Y_o = 0)$, without rescaling and using only the geometric set. The resulting accuracies with both passive and active sampling-based approaches, after 3, 6, 9 and 12 points samplings, are summarized in Table 3.2. Our framework (MTR-RL) outperforms LSTM model at each probe. It also outperforms the "Hom. Explo + PointNet" approach when only 6 points are sampled, and exhibits similar, yet slightly worse performances with 9 and 12 sampled points. However, training PointNet classifiers with point clouds sampled by our framework (MTR-RL + PointNet) allows to get better overall performances than training them with point clouds sampled using the homogeneous exploration "Hom. Explo + PointNet". The performance gap especially increases when decreasing the number of sampled points. Therefore, besides being trained from scratch, our framework allows to learn a more efficient exploration strategy than the hard-coded homogeneous version. Note that performances after the 3 first sampled points are not comparable, since these 3 initial points are pre-defined rather than learned in our framework.

The Figure 3.8 shows points collected by our trained active sampling framework, projected on the 3D objects they were sampled from. On easily distinguishable shapes (cube, sphere), our model chooses to broadly explore the surface of the object, as intended, thanks to the exploration term incorporated into the proposed reward. However, on similar objects (Quarter-Round, Triangle), our model seems to seek for key areas on the objects, that would allow the classifier to discriminate them. For instance, the difference between the Quarter-Round and the Triangle comes from the curvature of the upper surface, area that the model seems to mainly focus on (areas "1", "2" and "3" on Figure 3.8). Moreover, the missed point in area "4" seems to come from a sampling that brushed the upper surface of the triangle while searching for a potential "rounded surface", which is in line with the D_{out} reward term expectations.

The aforementioned results show that the proposed framework is able to learn an efficient exploration strategy but doesn't explicitly analyze which of its components contribute the most to its success. Moreover, such framework is of greatest interest in use cases where one needs to

Table 3.2 – Accuracies of point cloud classifiers on the geometric set, with frozen objects positions ($X_o = 0, Y_o = 0$), without random object scaling, with different exploration approaches. MTR-RL models are trained using Classification and Exploration rewards (β_c=1, β_e=1 and β_p=0)

	Sampled points			
	3	6	9	12
Passive sampling methods				
Rand. Explo + PointNet	58.48	71.52	78.68	83.20
Hom. Explo + PointNet	70.29	86.48	**100**	**100**
Active sampling methods				
LSTM	56.31	81.10	87.07	90.30
MTR-RL	90.52	98.85	99.92	99.95
MTR-RL + PointNet	90.67	**99.44**	**100**	**100**

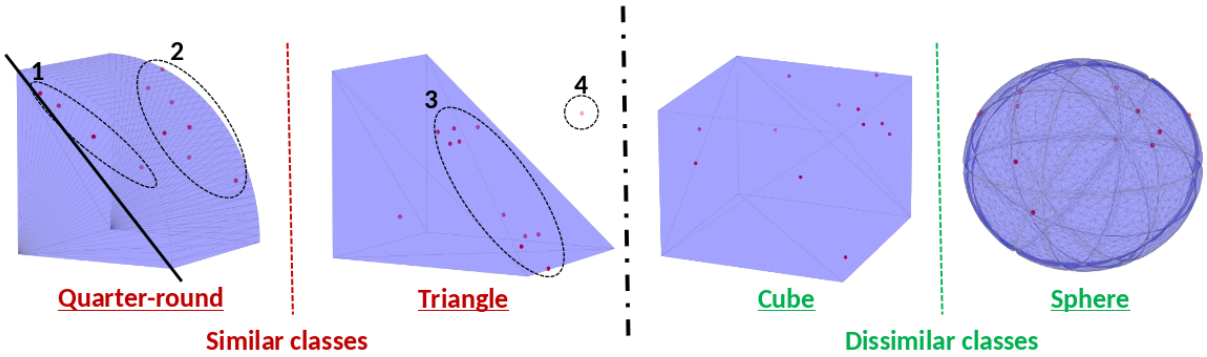


Figure 3.8 – Visualisation of 3D points sampled by our trained agent (red dots), projected on corresponding meshes. The agent was trained using both classification and full exploration reward and with no object positioning uncertainty.

learn to discriminate new objects quickly without having to label large-scale datasets each time. With the proposed framework, only 3D models of the objects of interest are needed. The framework will learn to focus on relevant areas of the object and forget noisy information from rest of the workspace, which makes it fully trainable in simulation without sim-to-real gap, as long as the object positioning can be known. However, on a real case scenario, one can not expect to know the exact position of the object in the workspace. Therefore, the performances of the proposed framework should be evaluated under varying object positioning. Moreover, in such realistic scenario, it is interesting to investigate whether our framework could be extended with a pose estimation head, that could predict the actual position of the object on the workspace.

Finally, an evaluation on realistic and more complex objects (e.g., the objects from the new set depicted in Figure 3.7) is needed to validate the usability of the framework on realistic scenarios. All the aforementioned questionings are addressed in the following sections.

3.4.3 Reward Terms Contributions

Exploration rewards are aimed to give supervision to the exploration task by rewarding the model i) when it touches the object, ii) when it broadly explores the object, iii) if a missed sampling brushed the object. Fixing the position of the 3D object at the origin facilitates the exploration task, as the sampling hardware only needs to turn around it and to target Z axis to ensure a broad exploration that always reaches the object. On the contrary, a more realistic scenario where the position of the object is unknown, hardens the exploration task by i) favoring missed samplings and ii) adding uncertainties on the explored object. Intuitively, the second scenario gives more credit to the exploration rewards, as they help to robustify the exploration strategy learning. In this section, we experimentally study the contribution of the different reward terms under varying amount of uncertainty on the position of the object, relatively to its fixed position at the origin $(X_o = 0, Y_o = 0)$.

The Figure 3.9 shows the training curves of our proposed model, trained with classification reward only, namely 'Class Rwd', and trained with both classification and exploration rewards, namely 'Class + Touch Rwd'. Training curves with varying levels of uncertainty on the objects positions are reported. "0mm noise" corresponds to the experimental settings in section 3.4.2, where the objects are always placed at the origin $(X_o = 0, Y_o = 0)$. On the extreme opposite, "40mm noise" means that the object is randomly placed with deviations to the origin, uniformly chosen in $[-40mm, +40mm]$, on both X and Y axis.

Strikingly, the training curves with 0mm positioning noise appear to be really unstable. We emit the hypothesis that it is due to the choice of Soft Actor Critic as underlying RL Algorithm, which is an off-policy RL algorithm, known for its high sample-efficiency yet high instability [64]. Moreover, with 0mm position uncertainty, it is easier for the classifier to overfit on a given point cloud distribution corresponding to a specific exploration strategy. If the exploration strategy changes because of the RL algorithm unstability, the classifier will not be able to generalize to the corresponding new point cloud distribution. Using the exploration reward seems to reduce the accuracy drops during training with 0mm position uncertainty probably by ensuring a more consistent exploration strategy. On the contrary, with higher levels of position uncertainty, the trainings are slightly more unstable with the exploration reward, yet overall quite stable compared to the 0mm position uncertainty scenario, which tends to confirm that the uncertainty on

the objects positions acts like a data augmentation strategy for the classifier.

More generally, the performance of the framework tends to decrease while increasing the amount of noise on the objects positions. However, as expected, the addition of the exploration term to the reward allows to maintain performances on par with the 0mm noise scenario, until 30mm position noise, and severely limits the accuracy drop with 40mm noise, compared to the 'Class Rwd' version.

3.4.4 Passive vs Active acquisition methods under noisy object positioning

The Table 3.3 allows to fairly compare the exploration strategy learned by our framework with the hard-coded homogeneous sampling strategy, under varying levels of noise on the object position, on the geometric object set.

The exploration strategy allows to outreach 90% accuracy from only 6 points (2 acquisition steps), and outperforms the homogeneous exploration strategy on the sparser point cloud densities, for all position uncertainty levels. However, when denser point cloud are considered (12 points densities), the homogeneous exploration strategy gives better classification results.

However, after sampling 12 points, all results are close to 100% accuracy, regardless of the object position noise level considered. To be fairly representative of a realistic use case, more realistic objects needs to be considered in the models evaluation.

3.4.5 Evaluation of the framework on realistic use case

Three levers were used to close the gap between the simulated scenario and a real-life industrial scenario : i) the uncertainty on the object positioning on the workspace; ii) the realism and the complexity of the considered objects and iii) the similarity between the objects to classify.

In this section, the realistic set of objects (Figure 3.7) was used as it contains 3D models of realistic and more complex objects Moreover, the four available objects share similar shapes, since they all represent variations of a gearwheel. To increase the difficulty of the recognition task, the objects can also be randomly rescaled when placed on the workspace, which prevents the models to bias the classification on the objects sizes, and therefore forces them to seek for key semantic differences on the objects. Finally, uncertainty can also be added on the position of the object, as done in section 3.4.4 on geometric objects. In this section, only extreme uncertainties are considered (0mm and 40mm).

The Table 3.4 shows the performances of "Hom. Explo + PointNet", "MTR-RL" and "MTR-RL + PointNet" approaches under the most difficult scenario, namely, on realistic objects, with

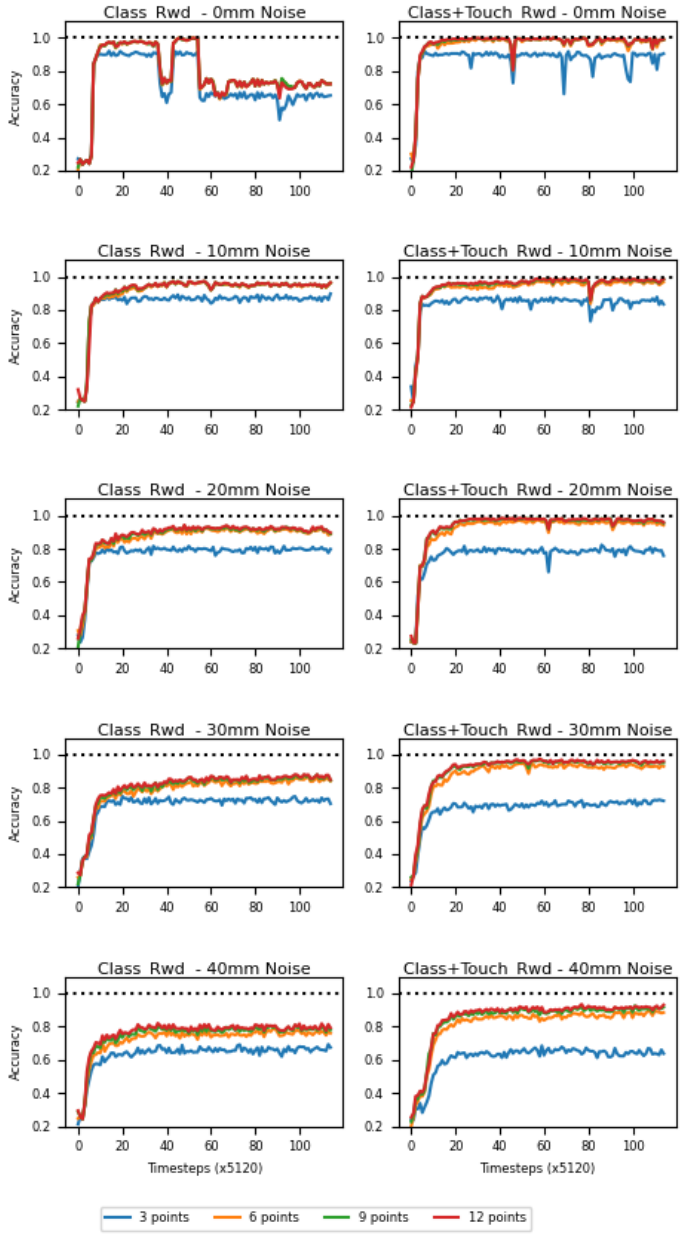


Figure 3.9 – Training curves of the proposed model without exploration reward (left column) and with exploration reward (right column) under varying uncertainty level on the object positioning (Noise). Training is done on the geometric set, without random objects scaling.

Table 3.3 – Accuracies of point cloud classifiers with Hom. Explo and MTR-RL exploration strategies, under varying levels of position uncertainty (pos. noise), on the set of geometric objects. MTR-RL models are trained using Classification and Exploration rewards (β_c=1, β_e=1 and β_p=0).

	Sampled points			
	3	6	9	12
Hom. Explo + PointNet				
0 mm pos. noise	70.29	86.48	100	**100**
10 mm pos. noise	58.48	86.46	98.55	**100**
20 mm pos. noise	54.91	83.02	97.45	**100**
30 mm pos. noise	54.69	80.51	96.02	**99.90**
40 mm pos. noise	54.39	77.51	92.32	**99.81**
MTR-RL + PointNet				
0 mm pos. noise	90.67	**99.44**	**100**	100
10 mm pos. noise	87.85	**98.36**	**99.00**	99.24
20 mm pos. noise	80.62	**98.68**	**99.48**	99.65
30 mm pos. noise	74.29	**96.02**	**98.36**	98.64
40 mm pos. noise	68.5	**91.56**	**95.28**	96.65

Table 3.4 – Accuracies of point cloud classifiers with different exploration approaches, on realistic objects, with 40mm position uncertainty, and random rescaling (Hardest Scenario). MTR-RL models are trained using Classification, Exploration and Localization rewards (β_c=1, β_e=0.5 and β_p=1).

	Sampled points			
	3	6	9	12
Hom. Explo + PointNet	56.27	**80.38**	80.3571	80.04
MTR-RL	46.0	71.66	72.52	72.52
MTR-RL + PointNet	56.92	78.37	**80.63**	**82.05**

40mm position uncertainty, and with random rescaling. The“MTR-RL + PointNet” approach shows the best results after 9 and 12 samplings, but is outperformed by the ”Hom. Explo + PointNet” approach after 6 samplings. The choice of the hyperparameter β_e=0.5 is justified from Tables 3.5 and 3.6.

The Table 3.5 summarizes the performances of "Hom. Explo + PointNet", "MTR-RL" and "MTR-RL + PointNet" approaches with different combinations of the random object scaling and

Table 3.5 – Accuracies of point cloud classifiers on realistic objects, processing point clouds from different exploration approaches. MTR-RL models are trained using Classification and Exploration rewards (β_c=1, β_e=1 and β_p=0)

	Sampled points			
	3	6	9	12
Hom. Explo + PointNet				
No scale, 0mm	100	100	100	100
Scale, 0mm	96.28	99.44	99.39	99.27
No scale, 40mm	65.55	99.72	99.8	99.81
Scale, 40mm	**56.27**	**80.38**	**80.3571**	**80.04**
MTR-RL + PointNet				
No scale, 0mm	100	99.98	100	100
Scale, 0mm	97.19	98.27	.9896	99.29
No scale, 40mm	62.97	98.62	98.42	98.47
Scale, 40mm	**56.45**	**65.83**	**65.64**	**66.31**
MTR-RL				
No scale, 0mm	100	100	100	100
Scale, 0mm	96.33	96.91	96.84	97.29
No scale, 40mm	56.69	95.49	96.85	96.2
Scale, 40mm	**46.8**	**51.38**	**52.16**	**52.04**

position uncertainty disturbances. The MTR-RL frameworks were trained with the hyperparameter settings formerly used on the geometric set (β_c=1, β_e=1 and β_p=0). First, we can notice that both "Homogeneous + PointNet" and "MTR-RL + PointNet" approaches show surprisingly high accuracies on the realistic set (nearing 100%), apart from the hardest scenario where both position uncertainty and object rescaling are applied together (Scale, 40mm). Secondly, with both position uncertainty and object rescaling applied together, our framework (both "MTR-RL" and "MTR-RL + PointNet" approaches) shows significantly worse results than Hom. Explo + PointNet. However, by further tracking reward terms evolution during training, we noticed that the hardest scenario, "Scale, 40mm", increased the classification task difficulty way more than the exploration task, as the framework was still able to quickly increase r_e, while having difficulty to improve r_c. This resulted in an imbalance between r_e and r_c, where r_c became negligible. To compensate this effect, we tried to change the reward weights hyperparameters, which, by default, were set to β_c=β_e=1. (The location reward is voluntarily ignored in the first place, which means $\beta_p = 0$). We especially played with β_e value to decrease r_e preponderance. Table 3.6

reports the evaluation results of our framework with β_e=0, β_e=0.5 and β_e=1 (3 values uniformly chosen between 0 and 1) on the "Scale, 40mm" scenario, while keeping β_c=1 and β_p=0.

The performances of our framework (MTR-RL + PointNet) varies from 66% to 79% accuracy (after 12 samplings) by only changing β_e hyperparameter. Moreover, we can notice that the best performing end-to-end model (MTR-RL) doesn't always provide the best exploration strategy for the PointNet model. As a matter of fact, the end-to-end MTR-RL model performs better with β_e=0 (classification reward only), while the exploration strategy learned by the model with β_e=0.5 allows PointNet to achieve top accuracies, matching Homogeneous + PointNet accuracy on this scenario (Table 3.5, "Scale, 40mm")

Table 3.6 – Impact of β_e hyperparameter choice on MTR-RL and MTR-RL+PointNet classification accuracies, on the set of realistic objects, with 40mm position uncertainty and applied random rescaling . (β_c=1, and β_p=0)

	Sampled points			
	3	6	9	12
MTR-RL + PointNet				
β_e=0 (classif only)	56.77	74.39	74.05	75.48
β_e=0.5	**56.34**	**79.11**	**79.8**	**79.8**
β_e=1	56.45	65.83	65.64	66.31
MTR-RL				
β_e=0 (classif only)	**48.62**	**72.133**	**73.06**	**73.42**
β_e=0.5	46.0	71.66	72.52	72.52
β_e=1	46.8	51.38	52.16	52.04

The overall best performances of our framework highlighted in Table 3.4 are further obtained by adding the localization reward term during the model training (β_p=1). The contribution of this reward term is detailed in the next section.

3.4.6 Pose estimator performances and contribution to the overall framework

In this section, we investigate the value of adding the pose estimation term to the reinforcement learning reward. The hardest scenario is considered, that is, 40mm position uncertainty with object rescaling. As demonstrated in section 3.4.5, It has shown significant potential for improvement.

Table 3.7 – Contribution of the pose estimation reward on the classification performances of realistic objects, with 40mm range of position offset and applied random rescaling . (β_c=1, and β_e=0.5)

	Sampled points			
	3	6	9	12
MTR-RL				
Pos. est. (β_p=1)	46.51	72.15	74.73	75.75
No Pos. est. (β_p=0)	46.0	71.66	72.52	72.52
MTR-RL + PointNet				
Pos. est. (β_p=1)	56.92	78.37	80.63	82.05
No Pos. est. (β_p=0)	56.34	79.11	79.8	79.8

The Table 3.7 compares the classification accuracy of our framework with and without pose estimation reward. The pose estimation reward gives more supervision during training and allows the framework to learn to figure out where the object is positioned, which results in an improvement on the accuracy of the framework, despite providing no additional information to the classification task.

Moreover, the pose estimation head, trained using the pose estimation reward, allows to estimate the position of the object on the workspace with 11mm, 10mm, 9mm and 10mm Root Mean Square Error, respectively after 3, 6, 9 and 12 points sampled.

3.5 From simulation to real robot : domain and embodiment generalization

DGA project deployment on real robot

3.6 Conclusion

Our research on active tactile exploration has led to the development of a reinforcement learning-based framework with a hand-crafted reward function. This framework allows a robot to efficiently sample a limited number of points in its workspace to recognize presented 3D objects. The crafted reward function enables the model to broadly explore objects, avoid missed

samplings, and focus on critical local parts for discriminating between similar objects. As a result, our model demonstrated superior accuracy compared to existing models in the end-to-end exploration-and-classification task.

Moreover, our model's ability to generalize to new embodiments was validated on a real robot equipped with a tactile sensor. This generalization is attributed to three factors: the model's focus on relevant information while ignoring background noise, the abstraction of the tactile effector as a ray-casting mechanism during training in simulation, and the incorporation of noise into the simulated tactile sensing system to enhance robustness against real sensor noise.

The representations of the scene obtained from Chapter 2 and from this chapter can further be combined in future work to provide a more complete representation of scenes where both vision and tactile information are available [44]. Both represenations are indeed in the form of a 4-D point-cloud, with 3 dimensions representing the spatial coordinates and the 4th dimension caraterising the point, either the object class of the parent object in the vision case, or tactile or contact feedback in the tactile case. While the tactile sensor used in this chapter was based on a single-point contact sensor, it could be extended to higher spatial resolution tactile sensors, like tactile arrays [167], artificial skins [8] or fingertip sensors [96], that can be easily embedded on robots grippers, as long as the 3D position of the sensor during the contacts is known.

More importantly, our approach highlights a significant challenge in reinforcement learning: the design of hand-crafted reward functions. Much of our effort was dedicated to this aspect, which is inherently task-specific. Moreover, the training of the model requires hundreds of thousands of interactions with the environment, which can only be achieved in simulation. All in all, the proposed framework is usefull to learn specific primitives in simulation that are massively reusable accross different tasks and easily transferable accross environment, and particularly from simulation to real robots. In this case, it is worth the effort to design a hand-crafted reward function and a simulation environment that can be used to train the model. But if one wants to quicly learn new tasks, with minimal human intervention, both the reward function design and the simulation environment design, which will be required for each new task considered, is a huge bottleneck.

To address this limitation, the subsequent parts of this work will shift focus towards learning tasks directly from human demonstrations, thereby reducing the dependence on hand-crafted reward functions. Part II will delve into understanding human behavior in demonstrations, while Part III will concentrate on learning to predict this behavior from the discussed scene representations. We will especially investiguate imitation learning algorithms that can learn from the demonstrations in an offline manner, which means that the robot does't need to interact with the

environment to learn the task, but only to observe the demonstrations. We will however leverage the 4-D point-cloud scene representations that we discussed in this part, as their allow to make abstraction of both the environment and of the embodiment in the demonstrations, and to learn the task in a more general way, that can be easily transfered to new environments and new embodiments.

PART II

Actions from Human demonstrations using Human pose estimation

Overview

Human pose estimation is the task of detecting a set of joints of a person in an image or video. This fundamental aspect of computer vision encompasses both 2D and 3D human pose estimation tasks, where the goal is to predict the coordinates of a person's joints either in two-dimensional space or in three-dimensional space within a predefined coordinate system. Multi-view human pose estimation, leverages multiple cameras to accurately estimate the 3D pose of a person. This approach effectively addresses the challenges of depth ambiguity inherent in monocular 3D pose estimation and mitigates issues related to occlusions.

Human pose estimation is a fundamental task in computer vision with applications including human-computer interaction, human behavior understanding, action recognition and ergonomics assessement. In the context of imitation learning, human pose estimation can help recognize and localize actions in a demonstration, which can further be executed as primitives by a robot. Additionally, studying human motion facilitates the modeling of human joint trajectories for specific tasks, which can then be translated into the action space of a robot.

The Chapter 4 provides an overview of the most popular datasets used for 2D and 3D human pose estimation and introduces our proposed dataset designed for multi-view human pose estimation under conditions of heavy occlusion. Addressing human pose estimation under heavy occlusions is vital for real-life scenarios, such as analyzing human behavior in cluttered environments like factories, where individuals frequently move among machines, shelves, and other obstacles. Fine manipulation scenarios, such as packaging, assembling parts or pick and place tasks, also involve occlusions with objects and tools, but also self-occlusions.

The Chapter 4 additionnally discusses the potential applications of the proposed dataset in ergonomics assessment.

The Chapter 5 compares existing and proposed solutions for 3D human pose estimation.

Finally the Chapter 6 investigates solutions for action recognition from human pose data.

CHAPTER 4

HUMAN POSE ESTIMATION DATASETS

Overview

Contents

Part of this chapter is based on a research paper submitted on ICIP 2024.

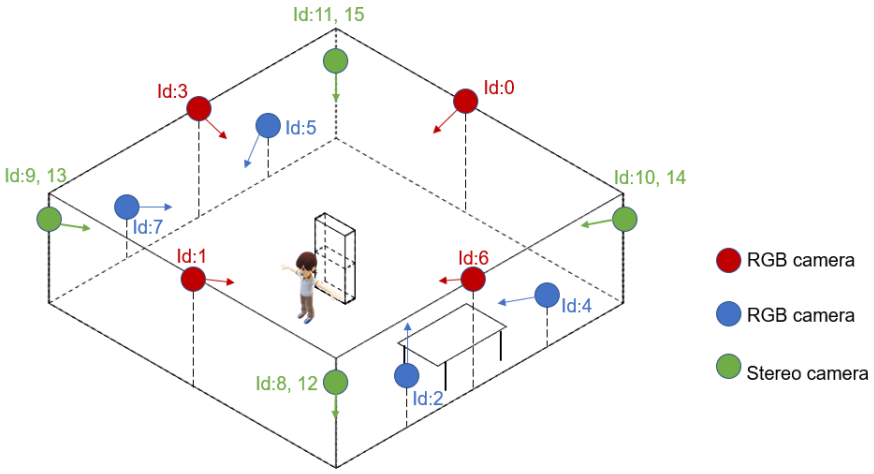


Figure 4.1 – **Overview of data acquisition setup.** 3D human pose estimation under : occlusions, various sensors, various optics, various viewpoints. How to balance views? How to generalize across views?

4.1 Introduction

The significant progress in 3D Human Pose Estimation (3D HPE) owes much to the wealth of ground truth annotations in training data. Nevertheless, the simultaneous capture of videos and corresponding ground truth 3D poses necessitates costly specialized motion capture (MoCap) equipment [51] (several thousands or dozens of thousands of dollars depending on the quality of the equipment and the number of sensors needed to cover the scene). Moreover, such equipment is intrusive, which means that the subjects must wear markers or suits, limiting the range of activities that can be captured. Additionally, MoCap systems, either bias the images if they are based on visual markers, or are subject to drift, where errors in position and orientation of sensors are cumulated over time, if they are based on IMUs [45]. On top of that, the calibration of MoCap systems with additional cameras can be complex and reduce the choice of the camera sensors that can be used to the ones compatible with the MoCap system. Alternatively, with calibrated camera parameters, the 3D pose can be directly triangulated from synchronized 2D poses detected in multiple cameras [77], albeit with a compromise in accuracy. Typical triangulation in 3D pose estimation can be divided into two stages: first estimating 2D poses in multi-view images, and then applying triangulation to derive the 3D human pose [6]. While substantial progress has been made within this pipeline, traditional triangulation methods [63] still face two constrains which hinder its application for 3D HPE: **(1) Information**

loss in the 2D pose estimation process; (2) Information loss of human skeleton prior in triangulation. To alleviate **(1)**, Iskakov *et al.* [77] and Tu *et al.* [171] opt to bypass the 2D estimation step entirely, integrating heatmaps directly into a discrete volume before regressing the 3D pose. However, these works necessitate 3D ground truth for training, which is the factor we're searching for in annotation process. Pavlakos [126] introduced a method for training 3D human pose estimation without ground truth, while eliminating the need for calibration parameters during inference. However, their performance in annotation, when calibration parameters are available, still falls behind basic triangulation methods. However, these approaches are designed for individual joints, frequently overlooking the relationship among these joints. The human body, on the other hand, inherently exhibits a structured arrangement with interrelated joints, offering robust priors for 3D HPE and introducing the constraint **(2)** to the stage. Several earlier study [11] incorporate prior knowledge of the human skeleton, surpassing the precision of earlier triangulation-based methods. [11] introduce a weighted triangulation-based approach, enhanced by incorporating bone length as a supervisory signal. In contrast to the prototype triangulation method [63], [11] employs confidence scores from the 2D pose estimator as weights for the Direct Linear Transform (DLT) process. While achieving impressive performance, the accuracy highly depends on the confidence estimation associated to the 2D pose prediction. However, bunch of factors, including, e.g., occlusions, distance from the cameras, variations in camera configurations (sensor and optics), may affect the reliability of the 2D pose prediction and associated confidence. To this end, we propose leveraging a parametric model to enhance the robustness of confidence scores by incorporating prior information derived from human skeletal structures. Consequently, a more dependable confidence score can be employed as the weighting factor in the triangulation process, thereby enhancing the overall performance of 3D pose estimation. Our contributions are threefold:

— We present a triangulation-based parametric model designed for annotating 3D Human Pose in multi-view datasets. This model enhances triangulation by leveraging more reliable confidence scores derived from the 2D pose input, estimated by an off-the-shelf estimator. It capitalizes on constraints provided by the human skeletal prior to improve accuracy.

— We introduce a novel 3D HPE dataset focused on construction scenarios, employing our annotation approach for labeling. Distinguished by heavy occlusions, a substantial number of views, challenging viewing directions, and the integration of diverse optical sensors, our dataset stands out from others.

— Extensive experiments on our dataset and Human3.6M demonstrate that our proposed

annotation method significantly enhances traditional triangulation in multi-view 3D HPE.

Code for our annotation tools and link to data can be found at:

https://github.com/KevinRiou22/supervised_hall6_pose_estimation

Regarding the context of this thesis, we must highlight that two related field of research can be explored in the litterature to analyse the pose of humans interacting with their environment. On one side, the field of 3D hand pose estimation focuses on the localization of the hand joints in 3D space, but usually depends on dataset that extend the problem to hand and object pose estimation, since these those datasets involve tasks that require the interaction of the hand with various objects. We include a sub-section on 3D hand pose estimation datasets in section 4.2, to provide an overview of the most popular datasets in this field. On the other side, the field of 3D human pose estimation focuses on the localization of the human body joints in 3D space, and usually involves datasets that capture human activities in various scenarios. We also provide a sub-section on existing 3D human pose estimation datasets in section 4.2, to provide an overview of the most popular datasets in this field.

In this chapter, we focus on the latter, and in addition to the above mentionned novelties regarding our proposed dataset, we can also highlight that despite being a human pose estimation dataset, our dataset incorporate tasks that involve human-object interactions, such as hammering, screwing, and box-shifting. We chose to focus on 3D human pose estimation instead of 3D hand pose estimation, for two reasons:

Firstly, 3D human pose estimation data is more general and can be used for a wider range of applications, including action recognition, ergonomics assessment, and human-robot interaction, while 3D hand pose estimation data is more specific to hand-object interaction tasks. In particular, our proposed dataset incorporated ergonomics-related labels for the contruction tasks performed by the participants, which can be used in future works to assess the physical strain of the participants during the tasks. We include details on those labels in section 4.5.

Secondly, the 3D human pose estimation data collected still enables us to study the performances of multi-view 3D pose estimation models in general, and especially their ability to generalize across various scenarios and camera setups. This generalization ability is key for the deployment of 3D pose estimation models in real-world applications, where the camera setup and the environment can vary significantly, but where it is not always possible to collect new data with a calibrated setup for each new scenario. In the end, the same principles developed for the pose estimation of the human body joints can be extended to the pose estimation of the additional hand joints. The collected data coupled with existing datasets will allow us to draw conclusions in Chapter 5 on the potential and limitations of multi-view 3D Human pose esti-

mation solutions for realistic scenarios. These conclusions will guide the choice of the solution leveraged to annotate our data used for imitation learning in Chapter 8.

To summarize, in this Chapter, we will start by reviewing the most popular datasets for 2D and 3D human pose estimation, as well as 3D hand pose estimation in section 4.2. We will then describe our proposed tool for annotating 3D human poses in occluded multi-view scenarios in section 4.3. We will further detail the features of our proposed dataset in section 4.4, and analyse the 3D annotations obtained using our annotation tool both on our dataset and on the Human3.6M dataset. Finally, we will discuss the potential applications of our dataset in ergonomics assessment in section 4.5.

4.2 Existing datasets for human pose estimation

4.2.1 Tasks definition

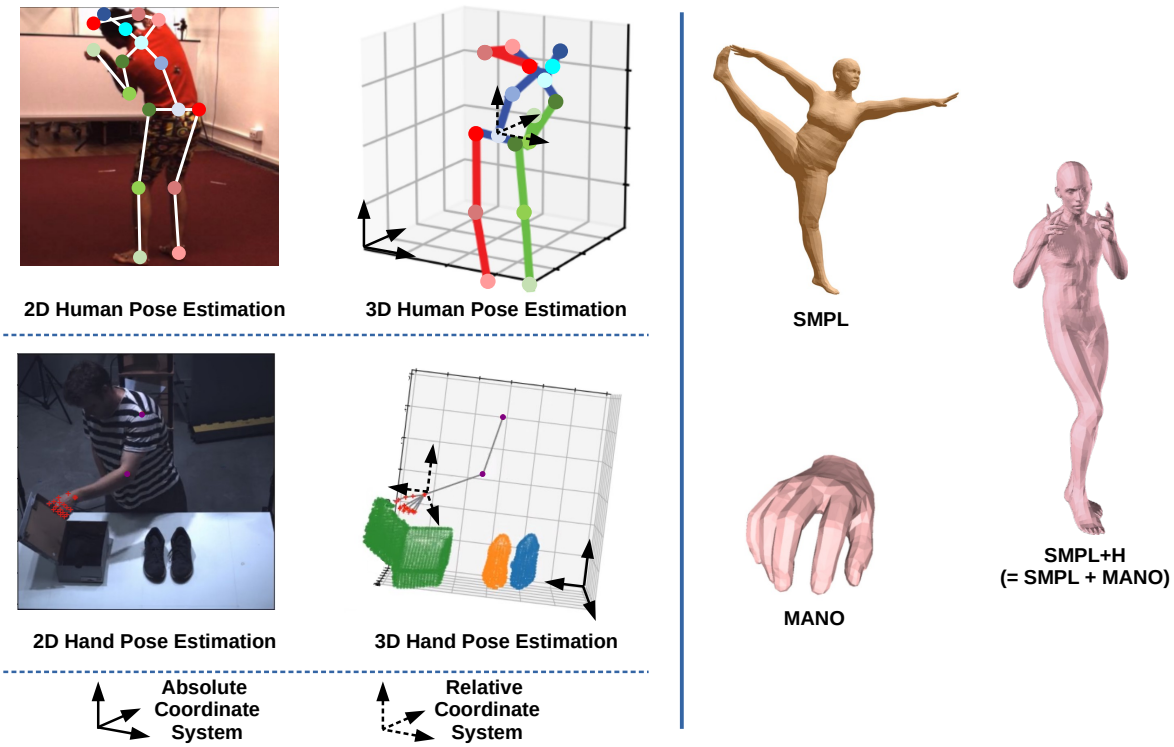


Figure 4.2

Human pose estimation can be done in the 2D space, where the goal is to predict the 2D coordinates of a set of joints of a person in an image, or in the 3D space, where the goal is to

predict the 3D coordinates of a set of joints of a person in a predefined coordinate system. The top-left part of Fig. 4.2 illustrates those tasks. Note that the 2D human pose estimation task is the first step of the two-stage 3D human pose estimation pipelines that we used in the Chapter 5. The 3D coordinates can be expressed in an absolute coordinate system, where the origin is fixed in the environment, or in a relative coordinate system, where the origin is defined in one of the joints of the person. The relative 3D human pose estimation task is particularly useful for applications where the absolute position of the person is not relevant, for instance if the person is working on a table amd not moving around, or for particular applications, such as ergonomic assessment, where the relative position of the joints is more relevant than the absolute position of the person in the environment. Relative 3D human pose estimation offers an interesting trade off between the complexity of the task and the relevance of the information for the application, and is therefore investiguated in the litterature [152] as an equally relevant task to absolute 3D human pose estimation. By omitting the absolute position of the person, the task is simplified, and the model can focus on the relative position of the joints, which is more relevant for the application. Moreover, as discussed in Chapter 5, the relative 3D human pose estimation task can help reduce ambiguities when training unsupervised models (i.e. models trained without 3D ground truth), by reducing the number of degrees of freedom in the 3D pose estimation task.

The bottom-left part of Fig. 4.2 illustrates the 2D and 3D hand pose estimation tasks. These tasks are similar to the human pose estimation tasks, but focus on the localization of the joints of the hand, and tend to be investiguated in the litterature as a separate task, on dedicated datasets and with specific models. In Fig. 4.2, we highlighted the shoulder and ankle of the operator in purple to illustrate the continuity between the human and hand pose estimation tasks, highlighting that the hand pose estimation task can be seen as a subset of the human pose estimation task, and can therefore be adressed with similar models. We also highlighted the objects of interest in the 3D space, firstly for clearer comparison with the corresponding image, and secondly to remind that the hand pose estimation task is often combined with the object pose estimation task, as the hand joints are often interacting with objects in the scene.

Finally the right part of Fig. 4.2 illustrates the SMPL [112] and SMPL+H [145] models, which are parametric models of the human body and hand, respectively. The SMPL is a model of the human body that allows to generate a 3D mesh of the human body given a set of parameters, such as the pose of the person or parameters representing the shape of the person. The SMPL was first trained to model the human body in a resting pose and with a canonical shape, which can be modified using the person-specific shape parameters and the parameters representing the pose of the person. The SMPL was trained on datasets of thousands of human body scans. The

SMPL+H model is an extension of the SMPL model that includes the hands. The hand extension relies on the MANO (hand Model with Articulated and Non-rigid defOrmations) model [145], which is trained similarly on a dataset of around 1000 hand scans.

While these models extend the 3D human pose estimation task to a more complete human body model, they also introduce additional complexity in order to recover volumetric information of the human body which are not relevant for our target applications, such as action recognition, hand trajectory anaylsis, or ergonomics assessment.

The only benefit that could be drawn from these models for our target applications would be in the context of the unsupervised learning of the 3D human pose estimation models that we are investigating in Chapter 5, where shape-related losses could be used to regularize the training of the models.

The next sections will provide an overview of the most popular datasets used for 2D and 3D human pose estimation, as well as 3D hand pose estimation.

4.2.2 2D Hand pose estimation

Most 2D hand pose estimation datasets proposed before 2020 were very limited in scene diversity, number of subjects, and views and tasks diversity. The NYU Hand dataset [168] proposed in 2014 provided 72757 train images from a single user, captured in an office with one frontal and two side kinect views. Another dataset, the MSRA Hand [130] was featuring 6 subjects, captured with one camera providing depth only (3D Intel's Creative Interactive Gesture Camera), with 2400 frames and data augmentations on the hands scales. After 2017, the development of 3D hand pose estimation datasets described in the next section, allowed to use these datasets for 2D hand pose estimation as well, by simply projecting the 3D annotations to the 2D image plane. However, the release of the COCO-WholeBody [82] dataset in 2020, which provides 133 keypoints annotations for each person in the image, including 17 for the body, 6 for the feet, 68 for the face and 42 for the hands, was a game changer for 2D hand pose estimation. The COCO-WholeBody dataset is indeed based on the large-scale COCO dataset [105], which provides 250,000 person instances labeled with keypoints, in various everyday scenarios. It allowed to train very general models for 2D human pose estimation, that can be used for 2D hand pose estimation as well.

4.2.3 3D Hand pose estimation

Annotating 3D hand poses is a challenging task, since the hands are highly articulated, the joints are often occluded by the fingers or the palm, and the hands are relatively small compared to the rest of the body, which excludes the use of MoCap systems if we don't want the images to be altered by the markers or sensors.

The only way to capture the 3D hand poses along with unaltered images is to use multi-view camera setups, which can capture the hands from different angles and recover the 3D poses by triangulation and fitting a hand model by leveraging losses on the 2D keypoints.

In 2019, the FreiHand [211] dataset was proposed, which provides 37k images of hand actions including interactions with objects, captured with a multi-view setup (8 RGB cameras), and annotated with 21 joints. The authors fitted the Mano [145] human hand model on each of the images, using the multi-view setup to define loss comparing the 3D annotation reprojected to the 2D image plane with the 2D annotation. They further fitted the shape parameters of the Mano model using a segmentation-based loss. They also leveraged human in the loop corrections to improve the annotations.

Following a similar annotation setup, the HO-3D [62] dataset was proposed in 2020, which provides 78k training images arising from 27 sequences (videos) from 10 different subjects manipulating 10 different objects from YCB [188] dataset. The dataset was captured with a multi-view setup made of 5 RGB-D cameras, and annotated with 21 joints.

The DexYCB [15] dataset was proposed in 2021, which is similar to the HO-3D dataset, but provides full demonstrations of hand grasping of objects, while the HO-3D dataset only provides videos of operators already holding the objects and mostly rotating them in their hands. Moreover, the DexYCB dataset provides 20 objects from the YCB-Video dataset, and consists of multiple trials from 10 subjects. The dataset was captured with 8 RGB-D cameras, provides 528k frames from 1000 sequences, and can be used for three relevant tasks: 2D object and keypoint detection, 6D object pose estimation, and 3D hand pose estimation.

Overall, these datasets for 3D hand pose estimation are still limited in terms of the number of subjects, views, and diversity of the tasks environments considered.

More recently, the Ego-Exo4D [54] was proposed in 2024, which includes 1286.30 video hours across 5035 takes from 740 participants, 13 cities arround the world, and 123 different natural scene contexts. It provides multimodal data (audio, gaze, egocentric and exocentric video, IMUS, etc.) and the shots were all recorded from multiple views, allowing the authors to recover the 3D poses of the participants, including the hands.

VR ?

4.2.4 2D Human pose estimation

The 2D human pose estimation datasets are largely dominated by the MS COCO [105] dataset and its derivatives, such as the COCO-WholeBody [82] dataset. Additionnal datasets have been proposed, such as OCHuman [204], focusing on heavily occluded humans but featuring only 5081 images. On the contrary, the COCO-WholeBody dataset provides 133 keypoints annotations for each person in the image, including 17 for the body, 6 for the feet, 68 for the face and 42 for the hands, on 250,000 person instances, and 200k images, in various everyday scenarios.

4.2.5 3D Human pose estimation

Numerous multi-view datasets have been proposed with joint position labels, such as Human3.6M[76], CMU Panoptic [83], and HumanEva [153]. The table 4.1 provides a comparison of the most popular 3D human pose estimation datasets. However, the majority of these datasets have limitations, with few cameras and fixed viewing angles directed towards a central stage, limited occlusions, and monotonous sensor diversity. Additionally, these datasets primarily capture social interactions and basic movements such as jogging and walking. Therefore, there is a notable absence of datasets emphasizing specific scenarios, such as construction operations involving activities like shifting, lifting, hammering, etc. Accordingly, we designed a unique dataset collection setup tailored specifically for the task of multi-view 3D pose estimation in construction scenarios.

Moreover, we engineered our dataset collection process to exhibit versatility across varying motion scales and optical configurations. This involved introducing a substantial number of cameras (16 cameras), with diverse optical sensors, and incorporating challenging viewing directions, as well as occluding objects in the scene.

The next section will present the annotation tool that we developped to annotate the 3D ground truth poses of the participants in our dataset in spite of the heavy occlusions and further detail the proposed dataset features.

EGO pose estimation? meta? VR?

Table 4.1 – Comparison for existing 3D Pose Estimation Datasets. Heavy occlusions, a sufficient number of views, challenging viewing directions, and the integration of diverse optical sensors distinguish our dataset from others.

Dataset	#Frames	#Cameras	#Subj	Tricky Viewing Direction	Optic sensor Variation	Environment	Characteristics/Issues
HumanEva [153]	80K	7	4	No	**Yes**	Indoor	Inconsistent body structure bettwen 3D annotation and 2D prediction
Human3.6M [76]	3.6M	4	11(5 female + 6 male)	No	No	Indoor	-
Shelf	-	3	4	No	No	Indoor	Heavy occlusion
Campus	-	2	3	No	No	Outdoor	-
HuMMan[12]	60M	11	1000	No	No	Indoor	Involving 500 actions
CMU Panoptic[83]	154M	480	Up to 8 subjects	No	No	Indoor	Various social interactions
MPI-INF-3DHP[118]	1.3M	14	8(4 female + 4 male)	No	No	Indoor& Outdoor	-
Total Capture[169]	1.892M	4	5(4 male + 1 female)	No	No	Indoor	Shift when reprojecting 3D GT to 2D
3D-Labor (Ours)	1.98M	16	5(2 male+3 female)	**Yes**	**Yes**	Indoor	Construction operations; Heavy occlusion

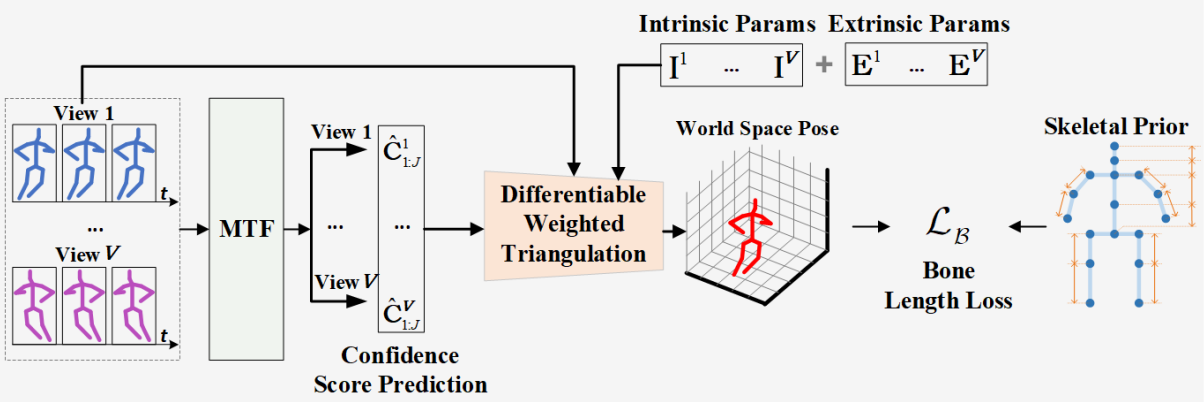


Figure 4.3 – **Overview of our method.** Rather than directly predicting 3D pose in world space, we employ a canonical weighted triangulation pipeline[11]. However, instead of using the weights provided by the off-the-shelf 2D pose estimator, we substitute them with predictions generated by the MTF [152] model. Let $\hat{\mathbf{C}}^v_{1:J}$ denote the confidence scores for the j-th joint in the v-th view, where j ranges from 1 to J, and v ranges from 1 to V. Here, J represents the total number of joints, and V represents the total number of views. Camera parameters including Intrinsic Params ($\mathbf{I}^{1:V}$) and Extrinsic Params ($\mathbf{E}^{1:V}$) are provided by multi-view camera calibration.

4.3 Data Annotation tool

4.3.1 Triangulation

for 3D Human Pose Estimation, as introduced by [11], is a 2 step process that allows to recover the 3D positions of a set of J joints defining the human skeleton. The first step consists in the simultaneous detection of the 2D joints in multiple views $\{\mathbf{u}_i^j\}_{i=1}^N$. The 2D detection of a joint j in the view $i \in \{1..N\}$ corresponds to its pixel position $\mathbf{u}_i = [u_i, v_i]$ in the corresponding image. Such detection can be done using off the shelf 2D pose estimation models, such as

HRNet [160]. A confidence in the detection C_i^j is also associated to each joint j, in each view i by the 2D detector.

The second step allows to lift the 2D joint detections to 3D, using Direct Linear Transform. The relationships between the 2D detections u_i, $i \in [1, N]$, the corresponding homogeneous 3D joint $\tilde{x} \in \mathbb{R}^4$, and cameras projection matrices $P_i \in \mathbb{R}^{3\times4}$, can be written as $A\tilde{x} = 0$, where

$$A = \begin{bmatrix} u_1 p_{1,3}^T - p_{1,1}^T \\ v_1 p_{1,3}^T - p_{1,2}^T \\ \vdots \\ u_N p_{N,3}^T - p_{N,1}^T \\ v_N p_{N,3}^T - p_{N,2}^T \end{bmatrix} \in \mathbb{R}^{2N\times4}$$

$\tilde{x}$ can be recovered as the unit singular vector corresponding to the smallest singular value of the Singular Value Decomposition of A. The final 3D position of the joint can be recovered by dividing the 3 first values of $\tilde{x}$ by its fourth value: $x = \frac{\tilde{x}}{(\tilde{x})_4}$

4.3.2 Triangulation with confidence weights

The reliability of 2D detections can vary across different views. Factors such as occlusions, joints positioned outside the field of view, distance from the cameras, and variations in camera configurations (including sensor and optics) can all contribute to this variability. To address this variability, a straightforward approach involves assuming that the confidence levels associated with the 2D detections accurately represent their reliability. Subsequently, joints in each view are weighted based on their respective confidence levels, by multiplying each row of A by a weight $w_j = \left(\frac{C_1}{\|a_1^T\|}, \frac{C_1}{\|a_2^T\|}, \ldots, \frac{C_N}{\|a_{2N-1}^T\|}, \frac{C_N}{\|a_{2N}^T\|}\right)$.

4.3.3 Triangulation with dynamic weights

Confidence levels associated with the 2D detections may not represent the optimal contribution of the corresponding joints to the triangulation, due to the high complexity of the factors that can affect the 2D detections. In this work, we propose to train a deep learning model to predict refined confidences , from the set of multi-view 2D detections and associated inital confidences :

$$\{\widehat{C}_{1:J}^{1:V}\} = MTF(\{\mathbf{u}_{1:J}^{1:V}\}, \{C_{1:J}^{1:V}\}),$$

where $\{\widehat{C}_{1:J}^{1:V}\}$, $\{\mathbf{u}_{1:J}^{1:V}\}$ and $\{C_{1:J}^{1:V}\}$ represent respectively the sets of refined confidences, 2D

detections, and initial confidences; for each joints and in each view. MTF refers to the Adaptive Multi-view and Temporal Fusing Transformer (MTF) [152], originally proposed for 3D human pose prediction from multi-view 2D skeleton data. We adapted the final convolutional layer of MTF to predict a refined set of JV confidences instead of the JV 3D poses originally output by MTF. The overall pipeline is illustrated in Fig. 5.2. We utilize the refined confidences $\widehat{C}_{1:J}^{1:V}$ to triangulate the 3D joints and supervise the entire process using skeletal priors for subjects in the dataset. Specifically, we incorporate prior knowledge of participant bone lengths, supervising the pipeline to ensure that the lengths of the bones in the triangulated skeleton $\widehat{b}_k$ match the corresponding participant bone lengths b_k^{prior}:

$$L_B = \sum_{k=1}^{B} (\widehat{b}_k - b_k^{prior})^2$$

4.4 Data analysis

4.4.1 Datasets

3D-Labor We collected a new dataset for multi-view 3D human pose estimation, incorporating several factors that complicate the assessment of view reliability for the 3D reconstruction task. The setup consists of a room equipped with 16 calibrated and synchronized cameras. Occlusions in the camera views are caused by a shelf and a table placed within the scene. The setup includes a set of 4 RGB cameras (Flir GS3-U3-41C6C-C) with 12mm focal length optics positioned above the scene. These cameras, labeled as cameras 0, 1, 3, and 6 in Fig. 4.1, are collectively referred to as "Flir Top." Additionally, there is a set of 4 Flir cameras positioned lower and closer to the scene, equipped with optics causing vignetting that narrows the field of view. These cameras, labeled as cameras 2, 4, 5, and 7 in Fig. 4.1, are denoted as "Flir Bot." Furthermore, there are 4 Stereo Cameras (ZED Mini) placed around the corners of the scene, aligned with the shelf's height. Each ZED camera provides two individual yet closely aligned views. With 8mm focal length optics, the ZED cameras offer wide fields of view. The cameras are labeled (12,13,14,15) and (8, 9, 10, 11) for the right and left views of the stereo cameras, respectively, and referred to as "Zed_r" and "Zed_l." We defined various multi-view scenarios comprising different combinations of these cameras, resulting in varying levels of occlusion and scene coverage. We recorded footage of 5 participants using the multi-view setup. Each participant completed 27 tasks, comprising 15 box-shifting tasks between the shelf and the table, along with 6 screwing tasks and 6 hammering tasks on the table. The cameras recorded at a rate of 30 frames per second.

We provide 2D joints annotations obtained using HRNet [160] pose estimation model trained on Coco Keypoint Dataset Format [82]. In section 4.4.4, we discuss triangulation approaches used to recover 3D pose ground truth from the 16 views.

Human3.6M It's a widely used benchmark dataset for 3D human pose estimation [76], comprising 3.6 million 3D human poses using four synchronized cameras at 50Hz, which are organized by 11 subjects. All the 3D pose annotations for these frames are obtained using a professional motion capture system.

4.4.2 Metrics

Percentage of Failed Parts (PFP) A bone is deemed failed if its error exceeds half the ground truth length.

3D Bone Error on data cleaned from failed bones. Although the Percentage of Failed Parts (PFP) effectively identifies instances of substantial error in 3D reconstructions, it overlooks the accuracy of successful cases. To provide a more comprehensive assessment, we introduce the "3D Bone Error" metric. This metric computes the error between ground truth bones and reconstructed bones, excluding the extreme "failed bones." By focusing on more typical scenarios, we obtain a finer estimation of 3D error.

Ground truth bone lengths are determined by triangulating from a manually selected frame, combined with manually selected views corresponding to that frame.

Reprojection error in the best view (Reproj. Error) While metrics based on 3D Bone Error provide insight into the accuracy of the reconstructed skeleton, they may not suffice to evaluate the precise positioning of 3D joints. This is particularly relevant in scenarios where a deep learning model guides triangulation to minimize bone error, yet does not guarantee correct joint positioning. To address this limitation, a complementary metric is necessary. The reprojection loss assesses how well the 3D skeleton aligns with 2D projections. It assumes the presence of at least one view with accurate 2D detections and compares the 3D reconstructed pose, when projected into 2D space, with the input 2D poses in the best view. Concretely, the reprojection loss computes the Euclidean distance between the input 2D poses and the corresponding reprojected poses from the view that yields the lowest distance.

P-MPJPE The Procrustes-aligned Mean Per Joint Position Error (P-MPJPE) is the L2-error of the 3D estimation, calculated after applying optimal rigid alignment (shift and scale) to both the predicted 3D pose and the ground truth 3D pose.

4.4.3 Benchmark Details

We train our model following the default configuration of MTF [152] for 60 epochs. Instead of splitting the dataset into training and testing sets, we train and test on all subjects for our data, focusing only on the training set subjects (S1, S5, S6, S7, S8).

4.4.4 Comparing Annotation approaches

Table 4.2 – Evaluation of 3D reconstruction approaches on our dataset.

	3D Bone Error (mm)							**Reproj. Error (best view) (px)**							**PFP**
	S1	S2	S3	S4	S5	Avg	Max	S1	S2	S3	S4	S5	Avg	Max	
16 views (basic)	30.3	32.2	44.5	33.6	33.2	25.9	2.14e+02	39.5	54.3	1.14e+02	54.1	68.0	68.2	1.06e+03	12.2
16 views (w/2DConf)	15.3	14.2	17.7	14.6	16.8	15.4	1.89e+02	4.7	4.6	6.3	4.5	4.6	5.0	6.83e+02	0.6
16 views (w/DynConf)	2.5	2.6	3.0	2.6	2.6	2.7	59.9	4.9	4.5	5.8	4.7	4.7	5.0	96.4	0.0
All Zeds (basic)	20.6	23.7	31.9	25.7	27.1	20.0	2.15e+02	28.7	42.8	1.06e+02	55.0	68.1	61.9	1.16e+03	17.8
All Zeds (w/2DConf)	16.3	15.1	20.2	16.1	19.0	16.3	1.92e+02	5.0	4.9	9.6	5.0	5.8	6.2	1.06e+03	2.6
All Zeds (w/DynConf)	5.1	5.1	7.1	5.2	5.8	5.6	1.92e+02	52.6	53.7	69.1	54.3	57.5	58.0	8.40e+02	0.2
Zed_r (basic)	20.5	23.6	30.5	24.9	26.4	20.3	2.15e+02	31.0	45.9	1.10e+02	57.4	72.8	65.3	1.07e+03	18.6
Zed_r (w/2DConf)	17.8	16.0	20.8	16.6	19.3	17.0	1.92e+02	7.1	6.6	10.8	6.6	7.1	7.8	1.01e+03	3.1
Zed_r (w/DynConf)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.0
Zed_l (basic)	20.9	23.7	30.3	25.2	26.4	20.3	2.15e+02	31.1	47.3	1.14e+02	58.8	72.4	66.7	9.68e+02	18.0
Zed_l (w/2DConf)	16.5	15.9	21.7	17.3	20.2	17.0	1.92e+02	6.5	6.3	11.2	7.1	7.5	7.9	1.31e+03	3.2
Zed_l (w/DynConf)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.0
Flirs Top (basic)	42.5	46.0	46.2	41.1	41.5	33.0	2.15e+02	1.35e+02	1.66e+02	2.17e+02	1.28e+02	1.64e+02	1.65e+02	1.02e+03	15.9
Flirs Top (w/2DConf)	20.9	21.9	27.5	22.5	22.4	21.1	1.92e+02	31.1	47.3	1.14e+02	58.8	72.4	66.7	9.68e+02	3.5
Flirs Top (w/DynConf)	5.9	6.2	6.8	6.7	6.6	6.4	1.75e+02	11.8	13.9	25.1	12.6	15.2	16.2	7.00e+02	0.2
Flirs Bot. (basic)	29.7	38.4	44.2	32.5	34.0	27.2	2.15e+02	76.0	1.04e+02	1.80e+02	1.09e+02	1.29e+02	1.22e+02	1.51e+03	16.3
Flirs Bot. (w/2DConf)	20.8	23.2	24.5	19.8	22.2	21.0	2.02e+02	13.1	13.8	14.8	12.3	13.1	13.5	1.26e+03	6.5
Flirs Bot. (w/DynConf)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.0
Zed_r, Flir Top (basic)	29.7	34.5	45.2	36.0	33.7	28.1	2.15e+02	54.1	66.3	1.19e+02	63.7	76.9	78.2	9.34e+02	14.1
Zed_r, Flir Top (w/2DConf)	16.7	15.1	19.7	16.2	17.8	16.5	1.91e+02	6.5	6.0	8.2	5.9	5.7	6.6	6.62e+02	1.0
Zed_r, Flir Top (w/DynConf)	3.5	3.7	4.3	3.7	3.8	3.8	1.06e+02	12.8	12.6	14.7	12.4	13.6	13.3	2.28e+02	0.0
Zed_r, Flir Bot. (basic)	26.3	30.9	45.8	30.9	32.8	21.8	2.15e+02	37.8	58.8	1.51e+02	64.6	83.5	82.5	1.25e+03	16.4
Zed_r, Flir Bot. (w/2DConf)	16.9	16.6	19.4	15.4	17.9	16.7	1.92e+02	6.8	6.5	9.0	6.3	6.5	7.1	8.19e+02	1.2
Zed_r, Flir Bot. (w/DynConf)	4.2	4.5	5.8	4.5	4.9	4.8	1.87e+02	15.1	13.6	16.2	13.7	14.0	14.6	3.51e+02	0.0

Table 4.3 – Validation of our DynConf weight prediction approach on H36M dataset train subjects.

	P-MPJPE	**3D Bone Error (mm)**	**PFP (%)**
basic	20.78	12.10	0.0
weighted 2DConf	27.41	16.27	0.0
weighted DynConf	17.99	5.53	0.0

The Table 4.2 reports 3D Bone Error, Percentage of Failed Parts and Reprojection Error metrics while triangulating without weights (basic), with the 2D detector weights (w/2DConf) and with the weights predicted by our trained model (w/DynConf). The results are reported for all the camera setups defined in section 4.4.1 , resulting in various levels of occlusions.

Analyzing the Impact of Occlusions on Triangulation

The presence of occlusions poses a significant challenge to triangulation, especially in scenarios with strong occlusions, resulting in substantial errors in 3D reconstruction and subsequently higher Percentage of Failed Parts (PFP).

Using PFP as a metric allows us to explore the setups' sensitivity to strong occlusions and understand how the introduction of 2D confidences in triangulation addresses this issue.

Initially sorting setups by PFP on basic triangulation results, we observe the following ranking from best to worst: "16 views," "Zed r, Flir Top" (8 views), "Flirs Top" (4 views), Flirs Bot. (4 views), "Zed_r, Flir Bot." (8 views), "All Zeds" (8 views), "Zed_l" (4 views), "Zed_r" (4 views). Notably, the use of multi-view information is not optimized, evident in instances where triangulating solely from "Flirs Bot." yields lower PFP than combining "Flirs Bot." and "Zed_r," which inherently provides more information.

Subsequently, sorting setups by the percentage improvement in PFP when using triangulation with 2DConf compared to basic triangulation, we find the following order of improvement from best to worst: "16 views" (95.1%), "Zed_r, Flir Top" (92.9%), "Zed_r, Flir Bot." (92.7%), "All zed" (85.4%), "Zed_r" (83.3%), "Zed_l" (82.2%), "Flir Top" (78.0%), and "Flir Bot." (60.1%). Sorting setups by triangulation with 2DConf PFP mirrors the order of PFP improvements.

It becomes evident that setups with more views and diverse perspectives benefit the most from weighted triangulation. For instance, "Zed_r, Flir Top" with diverse views outperforms "Zed_r, Flir Bot.," showcasing the advantage of incorporating diverse perspectives.

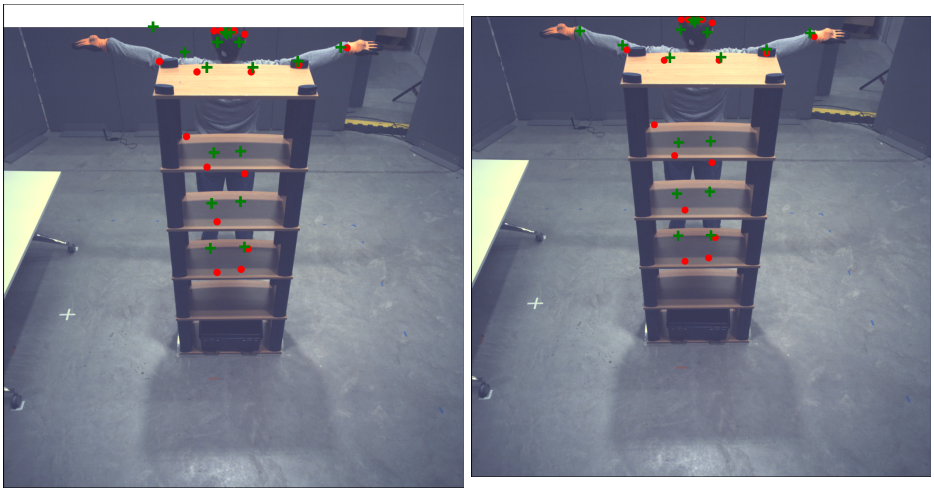
Shifting the focus to smaller occlusions, we employ 3D Bone Error on data cleaned from Failed parts as a proxy to assess sensitivity and understand how weighted triangulation influences these scenarios.

Sorting setups by 3D Bone Error on basic triangulation results, the order is as follows: "All zed," "Zed_r," "Zed_l," "Zed_r, Flir Bot.," "16 views," "Flir Bot.," "Zed_r, Flir Top," "Flir Top."

Subsequently, sorting setups by the percentage improvement in 3D Bone Error when using triangulation with 2DConf, the order from best to worst improvement is: "16 views" (40.5%), Zed r, Flir Top (41.3%), flir top (36.0%), Zed r, Flir Bot. (23.4%), flir bot (22.8%), all zeds (18.5%), zed_r (16.3%), and zed_l (16.3%).

When using triangulation with 2DConf, the setups can be sorted by 3D Bone Error, from best to worst as "16 views," "All Zeds," "Zed_r, Flir Top," "Zed_r, Flir Bot.," "Zed_r," "Zed_l, Flirs Bot.," "Flirs Top."

We observe a consistent pattern in the relationship between 3D Bone Error, derived from data cleaned of Failed parts, and the PFP metric. Notably, setups with a higher number of views



(a) 2DConf (b) DynConf

Figure 4.4 – Visualization of joints re-projected to 2D (green dots) following triangulation with 2DConf and DynConf weights (16 views). The red dots denote the initial 2D joint detections.

and greater view diversity exhibit a more pronounced benefit from incorporating confidence weighting into the triangulation process.

However, integrating 2DConf in the triangulation process appears to be less significant than in the more challenging scenarios highlighted by the PFP metric. This observation suggests that the influence of 2DConf varies based on the complexity of the reconstruction task.

An intriguing insight emerges from the results presented in Table 4.3, which showcases the outcomes of both basic triangulation and triangulation with 2DConf on the H36M dataset—a dataset exclusively featuring self-occlusions. Despite both approaches yielding a PFP of 0.0%, utilizing triangulation with 2DConf in such scenarios leads to an increase in both 3D Bone Error and P-MPJPE. This unexpected outcome suggests that in specific instances, incorporating 2DConf in triangulation may detrimentally impact the overall triangulation performance.

Enhancing Occlusion Handling through Predicted Triangulation Weights

Because of the high amount of PFP when triangulating from few views, the models trained to predict DynConf weights from 4 views failed to converge. However, except for these 4-view setups, DynConf weights allowed to reduce the PFP to almost 0% for all the studied setups. Moreover, using DynConf significantly reduces the 3D Bone Error (cleaned from Failed parts) compared to using 2DConf weights. The "16 views", "Zed r, Flir Top", "Zed r, Flir Bot." and "All zeds" setups reduced their 3D Bone Error respectively by 82.5%, 77.0%, 71.3% and 65.6%. We can notice that our approach also improves more the setups that showcase more views, and more diverse perspectives.

Overall, triangulating with 16 views and with DynConf weights allows 0.0% PFP and 2.7mm average 3D Bone Error on the whole dataset.

As a qualitative result, Fig. 4.4 highlights a triangulation that fails for the human's left wrist with 2DConf weights but succeds with DynConf weights.

On H36M dataset, as illustrated in Table 4.3, while using 2DConf weights decreased performances compared to basic triangulation, using the DynConf weights imporves both 3D Bone Error and P-MPJPE metrics.

Does our approach improve bone len at the expense of correct joint positioning?

As illustrated in Table 4.2, regarding the "16 views" scenario, using DynConf weights drastically improved the 3D bone error, while maintaining a stable re-projection error. Fig. 4.4 highlights the distribution of 3D bone errors and reprojection errors in the best view. Using DynConf weights squeezes most of the 3D Bone error distribution bellow 10mm, while showcasing a similar, and even slightly better Reprojection error distribution.

Interestingly, for setups that include Zed_r cameras, our model found a way to predict DynConf weights that reduce the 3D Bone error, while increasing the reprojection error. Fig. 4.5 highlights the re-projected joints compared to the predicted 2D joints for this case.

4.5 Use case on ergonomics?

4.6 Conclusion

In conclusion, the existing 3D human pose estimation datasets is limited by a lack of scenarios involving heavy occlusions and diverse camera configurations, including variations in camera positions, sensor types, fields of view, and resolutions. Addressing these gaps, we introduced a novel 3D human pose estimation dataset featuring five subjects performing multiple ergonomic variations of three tasks. Our dataset is distinguished by its significant occlusions, an extensive number of cameras, various optical sensors, and diverse camera placements.

We also proposed an innovative approach to enhance triangulation for better annotation of 3D ground truth poses under heavy occlusions. This method involves training an annotation model to predict weights for each camera and each joint, which are used during the triangulation process to mitigate occlusion effects. The weight prediction is optimized based on the bone lengths of the human skeleton, eliminating the need for access to the 3D ground truth poses. Our approach reduced the error on bone length by 82.5% on our dataset when using the 16 available cameras for triangulation, compared to weighting the cameras with 2D joint con-

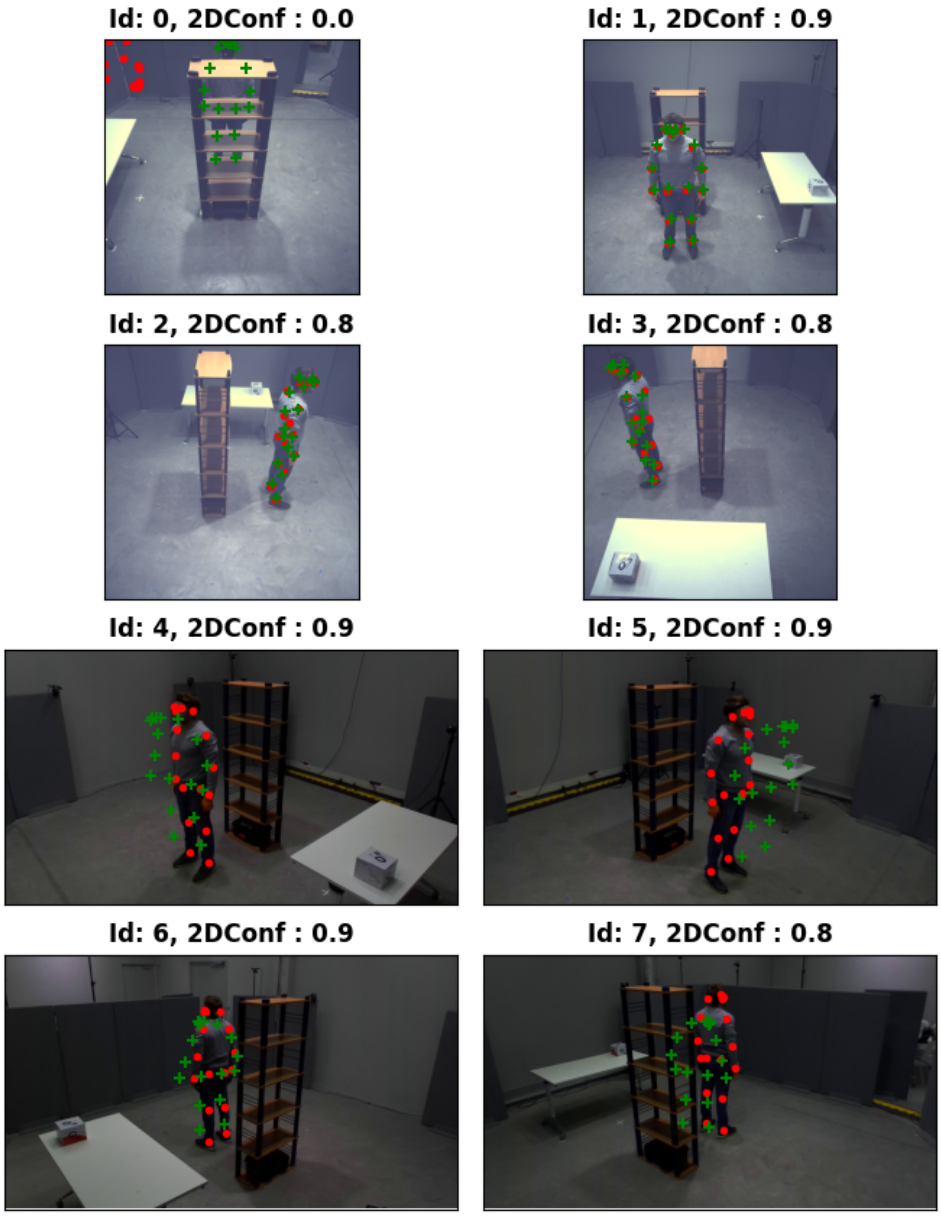
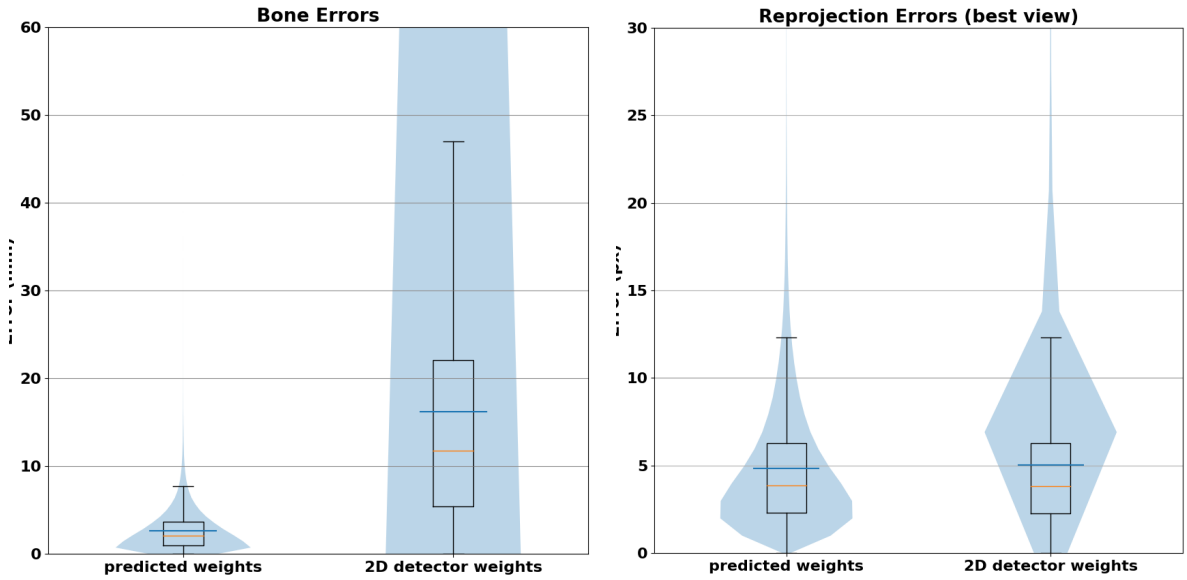


Figure 4.5 – Failure case with DynConf ("Zed r, Flir Top" setup on our dataset.)

fidence scores. We noticed especially noticed that our approach improves more the setups that showcase more views, and more diverse perspectives. Moreover, on existing datasets featuring only self-occlusions, the improvements achieved by our approach are less significant, which suggests that our approach is particularly beneficial for datasets with heavy occlusions.



(a) 3D Bone Error

(b) Reprojection Error

Figure 4.6 – Violin plot and box plot for 3D Bones and re-projection errors, when traingulating with 2DConf weights vs DynConf weights on our dataset.

Our annotation pipeline still relies on time-consuming and potentially error-prone information: the bone lengths of participants and the calibration parameters of the cameras. Consequently, our benchmarks will focus first on state-of-the-art supervised methods for 3D human pose estimation that do not require calibration parameters or bone length information once trained. Secondly, we will explore solutions that can be trained without annotated 3D ground truth poses, making them suitable for real-life applications where such poses are unavailable and difficult to estimate without additional information, like human bone lengths.

Finally, we will investigate the feasibility of training models for 3D human pose estimation without access to camera calibration parameters nor 3D ground truth poses. These investigations, to be detailed in the next chapter, will be conducted on both public datasets, such as Human3.6M, and our proposed dataset to evaluate the robustness of models against occlusions and variations in camera configurations.

CHAPTER 5

HUMAN POSE ESTIMATION MODELS

Overview

Contents

Part of this chapter is based on a research paper submitted on Transactions on Instrumentation and Measurement (Major review).

5.1 Introduction

3D Human Pose Estimation can be broadly categorized into single-stage and two-stage approaches. Single-stage methods directly regress 3D keypoints from RGB images in an end-to-end manner. In contrast, two-stage approaches, also known as 2D-to-3D lifting methods, first

estimate 2D keypoints using off-the-shelf 2D pose estimators and then train a second model to lift these 2D keypoints to 3D space.

Two-stage approaches generally exhibit better generalization to unseen environments as they are less impacted by background noise in images. They are also well-suited for incorporating additional constraints when leveraging multiple views, such as epipolar geometry, bone lengths, and camera calibration parameters. These additional constraints help mitigate the effects of occlusions and depth ambiguity in the 3D human pose estimation task.

A significant challenge in this field is the reduction of human intervention in the annotation process to enhance scalability. Obtaining 3D human pose ground truth and camera calibration parameters is time-consuming and prone to errors. Camera calibration, in particular, is often infeasible in real-life applications where cameras are not fixed and calibration parameters can change over time. Therefore, developing solutions that enable training and deploying 3D human pose estimation models without relying on 3D ground truth poses or camera calibration parameters is critical for practical applications.

In this chapter, we present both the most popular supervised 3D human pose estimation models, which require access to 3D ground truth poses during training, and unsupervised models, which do not require such access. For both cases, we explore approaches that can be trained with and without camera calibration parameters and discuss their limitations. Additionally, we propose a versatile architecture to lift 2D keypoints to 3D space, applicable not only to 3D human pose estimation but also to other tasks. This architecture has been tested in both supervised and unsupervised settings, using public datasets like Human3.6M, our proposed dataset to assess robustness to occlusions and camera configuration variations, and a simulated dataset featuring non-human keypoints.

Furthermore, we investigate an unsupervised 3D human pose estimation approach that does not require 3D ground truth poses or camera calibration parameters. We discuss the benefits and limitations of each approach, particularly in real-life applications across various use cases, including action recognition, ergonomics assessment, and human-robot interaction.

5.2 Introduction

In computer vision and robotics, 3D pose estimation has emerged as an active area of research due to its potential applications in fields such as 3D animation production, athlete posture correction, human pose measurement[**wu2022Alocalglobal**], and robot manipulation[**zhang2021safe**]. 3D Pose Estimation is the task of determining the 3D positions of keypoints on a object,

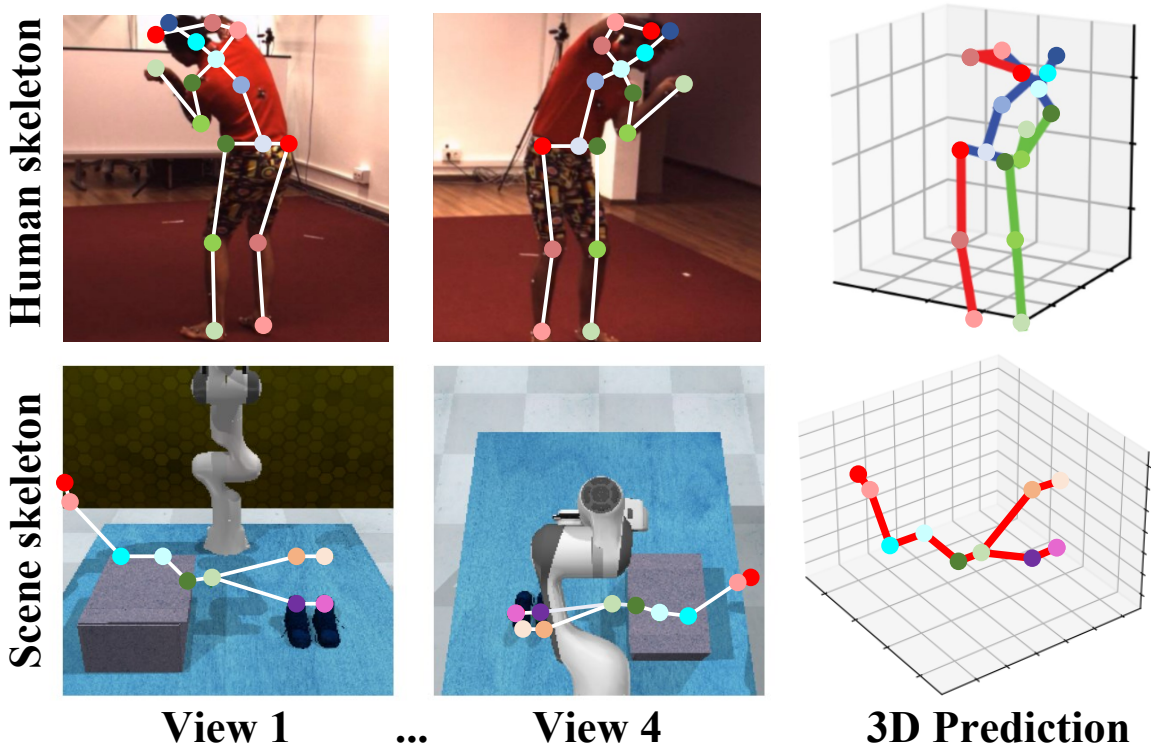


Figure 5.1 – A fundamental connection between "Human Pose Estimation" and "Robotic Manipulation" is made under the unified concept of articulated objects. Similar to the human body, we conceptualized the manipulation scene for robots as a articulated object, portraying an abstract "scene skeleton" that defines a set of 3D keypoints for a task to be executed. Building on this concept, we created a novel dataset for a "Scene Pose Estimation" paradigm under a multi-view setting. A multi-view 3D pose estimation approach is presented with the adaptation capability to both tasks mentioned.

such as human joints or keypoints on other articulated structures, from images or videos. Although it has received much attention and achieved remarkable progress, existing monocular 3D pose estimation approaches [**yin2023multibranch**] are still entangled in some intrinsic limitations, such as occlusion sensitivity and background ambiguity. To this end, multi-view-based approaches suggest using explicit geometric clues to solve this ill-posed problem. Multi-view 3D human pose estimation has been the focus of [**rhodin2018learning**, **qiu2019cross**, **zhang2021adafuse**, **shuai2022adaptive**, **qin2024inside**] since the human is the most studied object in almost every computer vision task. Most existing approaches are trained in a supervised manner [**he2020epipolar**, **shuai2022adaptive**, **castro2023crt**, 77], attempting to learn a

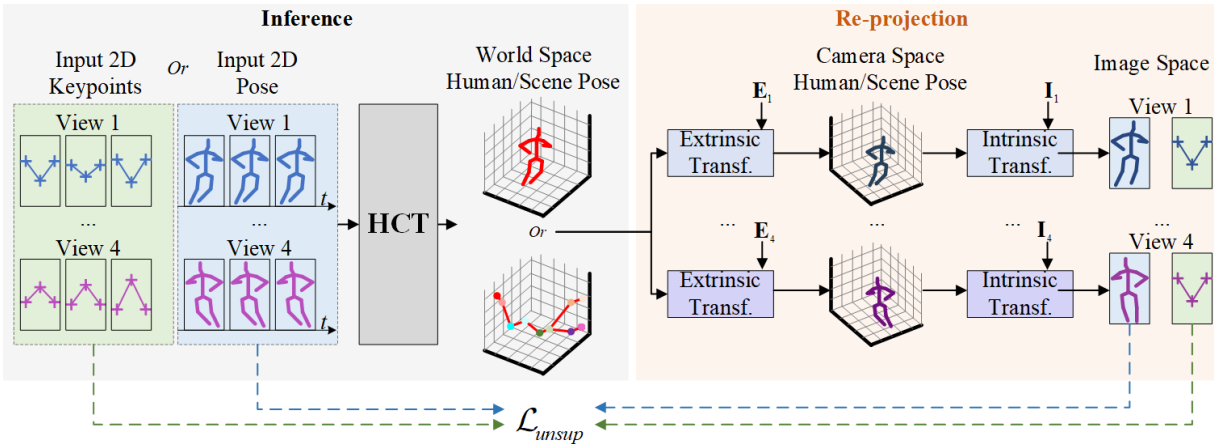


Figure 5.2 – **Our unsupervised training pipeline**: Our pipeline accepts input in the form of either **2D keypoints for scene pose estimation** or **2D human pose for human pose estimation**. The goal is to predict the 3D "Scene Skeleton" or "Human Skeleton", depending on the input type. Instead of computing the loss between the 3D ground truth and the 3D prediction, this pipeline constructs an unsupervised loss to calculate the geometrical consistency between the 3D prediction and the 2D inputs. The specifics of our proposed HCT are outlined in Fig.5.3. *Note: We refer to our pipeline as 'unsupervised' since it doesn't require any 3D annotation in the context of 2D-to-3D pose lifting.*

2D-3D mapping from an annotated dataset. However, data collection and annotation are time-consuming, especially in a multi-view scenario.

Many unsupervised and weakly supervised approaches [**chen2019unsupervised**, **kocabas2019self**, **iqbal2020weakly**, **mitra2020multiview**, **yu2021towards**] have been proposed to obtain accurate human 3D pose using minimal 3D ground truth. Despite notable progress, the performance of unsupervised approaches still lags behind fully supervised approaches. Two key challenges impede their advancement: incomplete geometric constraints and inadequate interaction among spatial, temporal, and multi-view features.

Two fundamental geometric consistencies construct the constraints of the majority of unsupervised and weakly supervised approaches: **1). Geometric consistency across multiple views**: When the same 3D pose is projected from different viewpoints and lifted, it should yield the identical 3D output in world space. **2). Geometric consistency across 3D and 2D Space**: If a 2D pose is lifted to 3D, and subsequently re-projected back to 2D, the resulting 2D pose should align coherently with the original input 2D pose. However, existing unsupervised methods struggle to ensure both types of geometric consistency simultaneously. To this end, we propose an efficient unsupervised approach that guarantees geometric consistency across different viewpoints and coordinate spaces without relying on any 3D annotations.

Additionally, existing transformer-based methods primarily emphasize multi-level spatio-temporal feature encoding but overlook the interaction and communication among these features. Therefore, we introduce a Hierarchical Cross Transformer (HCT), utilizing a Temporal Pyramid Transformer (TPT) for multi-level spatio-temporal feature generation, followed by a Cross Fusion Transformer (CFT) to facilitate interaction among these features.

Moreover, the majority of existing architectures predominantly concentrate on human pose estimation, integrating modules specifically designed for humans [**zhang2021adafuse**, **shuai2022adaptive**]. To liberate our approach from human-specific scenario, our proposed HCT doesn't rely on human-specific module, making it possible to adapt to new scenarios. To validate the adaptation of our proposed method, we establish an intuitive connection between human pose estimation and scene pose estimation, as described in Figure 5.1, and propose a new 3D scene pose estimation dataset DK-3D to benchmark the adaptation of our method for scene pose estimation. As illustrated in Figure 5.1, the scene can be intuitively represented as a articulated structure, where each component shares a skeleton defining the task to execute. In this context, the goal of scene pose estimation is to lift associated 2D keypoints on objects in the scene to their corresponding 3D positions, capturing the spatial information essential for effective robotic manipulation.

Our contributions can be summarized as:

— An unsupervised training pipeline where the model learns to predict a 3D pose and ensures geometric consistency across different viewpoints and coordinate spaces. Specifically, the re-projection from world space to camera space guarantees geometric consistency across various views, while the re-projection from camera space to 2D space ensures consistency across views, as shown in Fig. 5.2.

— To explore the interaction across various levels of spatio-temporal features from multiple views, we present the HCT as a novel backbone for multi-view pose estimation.

— Our approach is extended to Scene Pose Estimation for robotic visual manipulation tasks [**pan2023tax**], and Dynamic-Keypoints-3D (DK-3D), a new dataset, is proposed to validate its performance.

— Evaluations on Human3.6m [76], HumanEva-I [153], and our DK-3D dataset show that our approach adapts well to various scenarios, outperforming or achieving competitive results compared to state-of-the-art methods.

5.3 Related Works

In this section, we first briefly review prior approaches for multi-view 3D human pose estimation. Then, a discussion for weakly/unsupervised 3D human pose estimation is introduced to highlight their differences with our approach. Finally, we revise the keypoint based scene understanding which related to the second usecase of our approach.

5.3.1 Multi-view 3D Human Pose Estimation

Most works exploit geometrical information to explicitly or implicitly guide the 3D pose regression from multiple views under a volumetric scheme. [77] presented a volumetric-based network to fuse features from different views with known camera parameters. [**he2020epipolar**] proposed an approach that enables a 2D keypoints estimator to use relative features from neighboring views, following the concept of epipolar constrain. Although achieved reliable performance, volumetric approaches require significant computational resources, leading to bad generalization capabilities.

Some approaches predict 3D pose in a sparse space, and rely on a 2-stage strategy: first estimate 2D poses from an image and then extrapolate it to 3D with a learning-based model[**shuai2022adaptive**, **qiu2019cross**]. In this paper, we mainly focus on this kind of approach, which we refer to as the 2D-to-3D pose lifting task. [**shuai2022adaptive**] proposed a calibration-free approach that can encode geometrical clues only from 3D ground-truth data, resulting in a good adaptive ability to new calibration configurations. Nevertheless, these approaches need well-annotated datasets as supervision, which needs to be collected using specific motion capture (MoCap) systems that is hard to access.

5.3.2 Weakly/Unsupervised 3D Human Pose Estimation

To alleviate the challenges of data-hungry and overfitting, some weakly/unsupervised approaches have been explored to localize 3D pose over monocular setting [**zhou2018monocap**, **pavlakos2018ordinal**, **rhodin2018learning**, **chen2019unsupervised**, **hua2022weakly**, **usman2022metap**, **chen2023multi**]. Majority of them need object-specific information as prior knowledge to guide 2D-3D mapping [**zhou2018monocap**, **pavlakos2018ordinal**, **chen2023multi**], such as bone length, pose templates, etc. [**zhou2018monocap**] used a 3D pose dictionary as a pose prior for monocular 3D pose reconstruction. [**pavlakos2018ordinal**] explored ordinal depth relations among joints without requiring 3D annotations. [**rhodin2018learning**] utilized 2D data

to train their model by incorporating multiple images of a single pose and a few guidance from 3D data as a weak-supervision mode. [**chen2023multi**] proposed an interactive scheme to fully leverage human's articulated information, identifying unreasonable and occluded articulations. [**kundu2020kinematic**] attempted to construct an unsupervised approach without paired supervision. But it still needs prior kinematic information even if it is minimal compared to previous works. Therefore, weakly-supervised methods require either 3D supervision or extra object-specific supervision. A few works proposed solutions for unsupervised 2D-to-3D pose lifting under monocular settings. Hua et al. [**hua2022weakly**] introduced a cross-view weakly supervised architecture based on a U-shaped Graph Convolutional Network (GCN), achieving impressive performance without using ant 3D annotation. However, GCN-based architectures require a predefined spatial graph specifically designed for the human skeleton. [**kocabas2019self**] suggested a 2-branches network to predict the calibration parameters and 3D poses utilizing geometric constrain and 2D ground truth poses. [**li2020geometry**] trained a monocular network with re-projection and cross-view loss, with the prior clues from geometric constraints and temporal sequences. However, unsupervised 3D pose estimation over multi-view settings has not attracted too much attention from researchers, due to the straightforward triangulation [**hartley2003multiple**] alternative under this setting. Nevertheless, triangulation suffers from extrinsic parameters dependency and sensitivity.

There are two works closely related to our approach [**wang20193d**, **chen2019unsupervised**]. Motivated by the geometric consistency across 3D and 2D space, [**wang20193d**] employed several fully-connected layers to implicitly project predicted 3D pose to 2D. Then optimized the entire model with a re-projection loss, supplemented by supervision from both 2D and 3D ground truth when available. Differs from [**wang20193d**], we consider both 3D-2D and cross-view geometric consistency using explicit geometric constraints (camera parameters), eliminating the need for 3D ground truth annotations. With both of geometric consistency in mind, [**chen2019unsupervised**] first randomly projects the 3D prediction to 2D. Then, it applies a symmetrical 2D-to-3D lifting pipeline with the projected 2D pose as input and finally executes another 3D-to-2D projection process. In comparison, our unsupervised pipeline integrates both of 3D-2D and cross-view geometric consistency within a single projection process. Our projection can be divided into 2 stages: 1) Projecting 3D predictions from world space to camera spaces, which inherently ensures consistency across views; 2) Projecting 3D pose in camera space to 2D space, ensuring 3D-2D consistency. Furthermore, another fact that sets us apart from both [**wang20193d**, **chen2019unsupervised**] is our adaptability to scene pose estimation. We demonstrate this through evaluations on a new benchmark dataset, DK-3D.

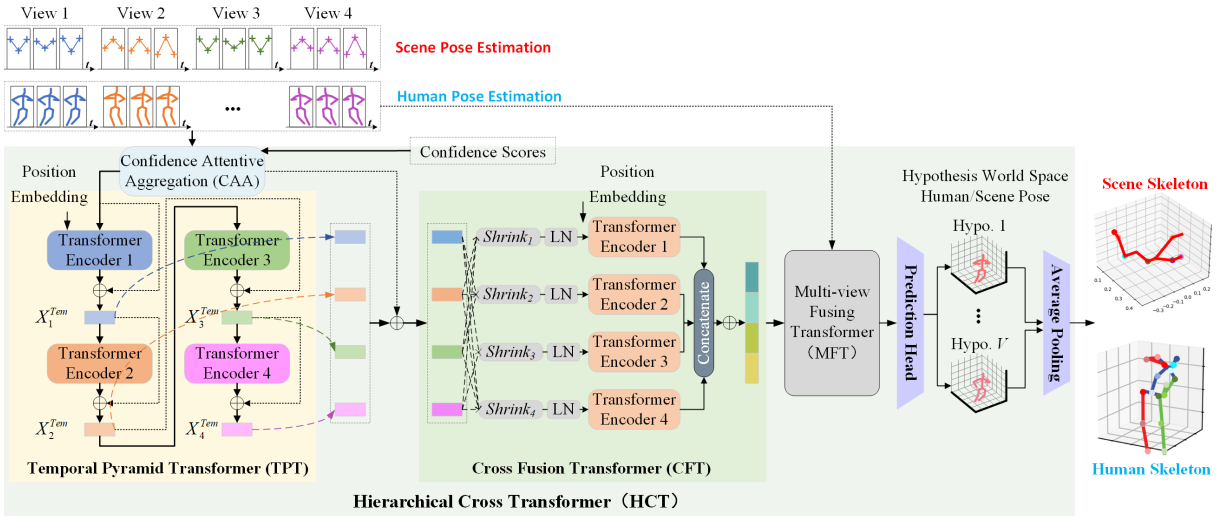


Figure 5.3 – The architecture of the HCT. It comprises four modules in series: the CAA adapted from [**shuai2022adaptive**], the proposed TPT and CFT, and the MFM borrowed from [**shuai2022adaptive**]. The HCT predicts the scene skeleton for "Scene Pose Estimation" when provided with 2D keypoints or the human skeleton for Human Pose Estimation when given 2D pose as input.

5.3.3 Keypoint based Scene Understanding for Robotic Manipulation

Although 6 Degrees of Freedom (DoF) prediction can locate objects in 3D space [**castro2023crt**], it is still limited in several application scenarios. Firstly, it only works under the assumption that the object is rigid. Secondly, it lacks the representation capacity to capture spatio-temporal dependency when applied to specific tasks such as robotic manipulation. To this end, keypoint based pose representation has emerged as a promising alternative to 6-DoF prediction for certain applications, particularly those involving non-rigid objects or tasks requiring a more detailed representation of the object's motion [81]. In keypoint based pose representation, keypoints on the object are identified and tracked over time. These keypoints can be used to construct a more detailed and accurate representation of the object's pose, including non-rigid objects, information about its absolute position, relative position to other objects, and possible trajectories for a specific task. Although some works have been presented [162], most of them use monocular or stereo observations under a supervised setting [**liu2020keypose**].

5.4 Method

This section describes our unsupervised 3D pose estimation approach in a multi-view setting. We describe the proposed unsupervised training pipeline in section 5.4.1. Details of the backbone (HCT) are provided in section 5.4.2.

5.4.1 Proposed Unsupervised Training

Our unsupervised training pipeline aims to optimize a 2D-to-3D pose lifting model by defining a proxy-goal. The goal is to ensure that the 3D keypoints, when lifted and subsequently back-projected to image space, maintain as much coherence as possible with their original 2D inputs. The unsupervised training pipeline works in 4 steps.

Step 1. 2D Pose Detection: Following [**shuai2022adaptive**] strategy, 2D keypoints are extracted from the multi-view videos using an off-the-shell 2D pose estimator, e.g., CPN [**chen2018cascaded**] for Human Pose Estimation case. Let $\mathbf{I}_t$ be the t-th frame of video $\mathbf{Vid}_v$ taken from the camera v, where $\mathbf{I}_t \in \mathbb{R}^{W \times H \times 3}$. For each frame, the 2D keypoints and their corresponding confidence are first extracted by an 2D pose estimator $\mathcal{D}_{2D}\left(\right)$:

$$\{\mathbf{K}_{2D,}\mathbf{C}_{2D}\} = \mathcal{D}_{2D}\left(\mathbf{I}_t\right), \tag{5.1}$$

where $\mathbf{K}_{2D} = \{(x,y)_i\}_{i=1}^{J}$ is the set of the 2D coordinates of each keypoint, (J keypoints in total), and $\mathbf{C}_{2D} = \{(c)_i\}_{i=1}^{J}$ refers to the confidence score of each keypoint. $(\mathbf{K}_{2D})_t^v$ and $(\mathbf{C}_{2D})_t^v$ denotes the keypoints and their confidence value for camera/view v and frame t. There are V cameras/views, and each video has T frames.

Step 2. Lifting image space 2D poses to world space 3D poses: This step corresponds to the “Inference" part of Figure 5.2. Our proposed HCT backbone (as shown in Figure 5.3) is used to lift 2D poses to 3D poses proposals, which are later pooled as one single world-space 3D pose. The HCT 2D-to-3D lifting is formally described by Eq. (5.2).

$$\widehat{\mathbf{K}}_{3D} = \mathcal{HCT}\left(\{(\mathbf{K_{2D}}, \mathbf{C_{2D}})_t^v\}_{t=1}^{T}\right), \tag{5.2}$$

where $\widehat{\mathbf{K}}_{3D} = [x_i, y_i, z_i]_{i=1}^{J}$ is the 3D coordinates of all J keypoints for the predicted 3D pose of middle frame in the world space coordinate system.

Step 3. Projecting 3D prediction back to 2D image space:

This step corresponds to the “Re-projection" part of Figure 5.2. The predicted 3D pose is

projected back to the different 2D views using the cameras' extrinsic and intrinsic parameters.

$$(\widehat{\mathbf{K}}_{2d})'_v = K^v[R^v \mid R^v T^v] \begin{bmatrix} \widehat{\mathbf{K}}_{3d}^T \\ 1 \end{bmatrix} \tag{5.3}$$

$$K^v = \begin{bmatrix} f_x^v & 0 & c_x^v \\ 0 & f_y^v & c_y^v \\ 0 & 0 & 1 \end{bmatrix}, T^v = \begin{bmatrix} T_x^v \\ T_y^v \\ T_z^v \end{bmatrix} \tag{5.4}$$

where $(\widehat{\mathbf{K}}_{2D})_v = [u_i, v_i]_{i=1}^J$ is the result of eliminating the last term of a homogenized matrix $(\widehat{\mathbf{K}}_{2d})'_v$. K^v indicates the intrinsic camera parameters for v-th camera (e.g., focal length f_x^v and f_y^v, principal point c_x^v and c_y^v). R^v and T^v represent the v-th camera's extrinsic parameters for rotation and translation, respectively. For simplicity, we omit camera distortion.

Step 4. Computing the loss function in the 2D image space: The training loss is computed in the 2D space as the L_2 loss between the 2D input poses and the predicted 3D poses re-projected to 2D:

$$\mathcal{L}_{unsup}(\widehat{\mathbf{K}}_{3D}) = \frac{1}{VJ} \sum_{v=1}^{V} \sum_{i=1}^{J} \left\| (\widehat{\mathbf{K}}_{2D})_{v,j} - (\mathbf{K}_{2D})_{v,j}^{\lceil \frac{T}{2} \rceil} \right\|_2 \tag{5.5}$$

Following [**shuai2022adaptive**], our backbone predicts the 3D pose of the central frame $\lceil \frac{T}{2} \rceil$ from the temporal sequence of T input 2D poses $\mathbf{K}_{2D}$).

5.4.2 Backbone Details

Some of the existing works address 3D Pose Estimation using human-specific prior knowledge [**zhou2018monocap**, **pavlakos2018ordinal**] or encoding mechanism [**shuai2022adaptive**], limiting generalization ability. To this end, we designed a Hierarchical Cross Transformer (HCT) without any human-specific knowledge.

Fig 5.3 illustrate the architecture of the HCT. Given multi-view 2D pose sequences and corresponding confidence scores estimated by an off-the-shelf 2D pose detector, our HCT predicts the 3D pose in world space for the middle frame of the input sequence. The HCT mainly comprises four modules arranged in series: the Confidence Attentive Aggregation (CAA) adapted from [**shuai2022adaptive**], the Temporal Pyramid Transformer (TPT), the Cross Fusion Transformer (CFT), and the Multi-view Fusing Transformer (MFT) which is adapted from [**shuai2022adaptive**]. Before the TPT, we employ a CAA to mitigate the impact of unreliable 2D poses on the final result, without separately processing different parts of the human body as done in [**shuai2022adaptive**].

The TPT encodes temporal correlations of the input as several hierarchical features in a pyramid transformer structure. Subsequently, these hierarchical features and the input of the TPT are incorporated in the CFT to aggregate spatial features along both joints and views axis. Then, the MFT is applied to further fuse features from multiple views. Finally, the prediction head produces the output.

Temporal Pyramid Transformer

The Pyramid Transformer has been used in 3D human pose estimation by [99] to generate multiple hypotheses for avoiding the sub-optimal problem. Following the similar structure, we build our TPT with M cascaded transformer encoder. The difference between TPT and [99] is that TPT aims to extract different levels of semantic information along the temporal axis, while [99] produces several plausible 3D pose hypotheses. Additionally, the TPT transformer encoder consists of only one self-attention layer, whereas [99] includes four. The outputs of the TPT are M features $[X_1^{Tem}, \cdots, X_M^{Tem}]$ generated by each of M encoders, where $X_m^{Tem} \in \mathbb{R}^{V \times T \times C}$ indicates the feature yielded by m-th encoder. Each encoder in the TPT is applied as a residual connection to facilitate gradient propagation. A learnable embedding jointly encodes the spatial information for views and joints.

Cross Fusion Transformer

Given M hierarchical features $[X_1^{Tem}, \cdots, X_M^{Tem}]$ from the TPT, the CFT extracts higher-level features by enabling interactions among these hierarchical features. However, hierarchical features mainly consist of spatio-temporal information and lack the original representation for cross-view correlations. Therefore, the input of TPT is combined with hierarchical features to complement the low level cross-view correlations in the original input:

$$X_m^{'} = X_0 + X_m^{Tem} \tag{5.6}$$

where $X_0 \in \mathbb{R}^{V \times T \times C}$ indicates the input of TPT, V, T and C denote the number of views, the sequence length of the input 2D pose and the channel dimension, respectively. The merged feature $X_m^{'}$ is encoded by following process to meet the input requirements of the CFT:

$$\tilde{X}_m = \mathrm{LN}\left(\mathrm{Shrink_m}\left(\mathrm{CAT}\left(X_1^{'}, ..., X_M^{'}\right)\right)\right) + E_m^V \tag{5.7}$$

We note CAT(·) as concatenate operation along the last dimension, LN(·) a LayerNorm operation, $\mathrm{Shrink}(\cdot)_m$ and $E_m^V \in \mathbb{R}^{V\times T\times(C\times 3)}$ a multi-layer perceptron (MLP) and a learnable position embedding for m-th Transformer Encoder of CFT. After that, we divide $\tilde{X}_m \in \mathbb{R}^{V\times T\times(C\times 3)}$ into $\tilde{X}_m^q \in \mathbb{R}^{V\times T\times C}$, $\tilde{X}_m^k \in \mathbb{R}^{V\times T\times C}$, and $\tilde{X}_m^v \in \mathbb{R}^{V\times T\times C}$ along the last dimension to serve as the query, key, value for the m-th Transformer Encoder in CFT, respectively:

$$Z_m = \mathcal{T}_m(\tilde{X}_m^q, \tilde{X}_m^k, \tilde{X}_m^v) \tag{5.8}$$

$$\mathcal{T}_m(\cdot) = \mathrm{MHSA}_l(\cdot) \tag{5.9}$$

where $Z_m \in \mathbb{R}^{V\times T\times C}$ is the output of m-th Transformer Encoder $\mathcal{T}_m(\cdot)$ of CFT, $\mathrm{MHSA}(\cdot)$ denotes multi-head self-attention. Then, we concatenate $\{Z_m\}_{m=1}^M$ along the last dimension, following LayerNorm an MLP to generate the final output of CFT:

$$\tilde{Z} = \mathrm{MLP}(\mathrm{LN}(\mathrm{CAT}(Z_1, ..., Z_m))) \tag{5.10}$$

where $\tilde{Z} \in \mathbb{R}^{V\times T\times(C\times 2)}$ and M indicates the output of the CFT and the number of Transformer Encoder in the CFT, respectively. Subsequently, $\tilde{Z}$ is fed into MFT for further processing. Details of the MFT can be found in [**shuai2022adaptive**].

Prediction Head

A multi-hypothesis prediction head is applied to generate the final 3D pose in world space. Given $\tilde{Z}' \in \mathbb{R}^{V\times T\times C}$ as the output of the MFT, the prediction head first encodes $\tilde{Z}'$ to multiple 3D pose hypothesis $H \in \mathbb{R}^{V\times J\times 3}$:

$$\tilde{Z}'' = \mathrm{Reshape}(\mathrm{Clip}(\tilde{Z}')) \tag{5.11}$$

where $\mathrm{Clip}(\cdot)$ denotes clipping the middle frame of the feature from $\tilde{Z}'$ along the second dimension, and $\mathrm{Reshape}(\cdot)$ denotes dimension transposition. Subsequently, $\tilde{Z}'' \in \mathbb{R}^{V\times C\times 1}$ is fed into a MLP to produce multi-hypothesis 3D poses $H \in \mathbb{R}^{V\times J\times 3}$:

$$H = \mathrm{MLP}_o(\tilde{Z}'') \tag{5.12}$$

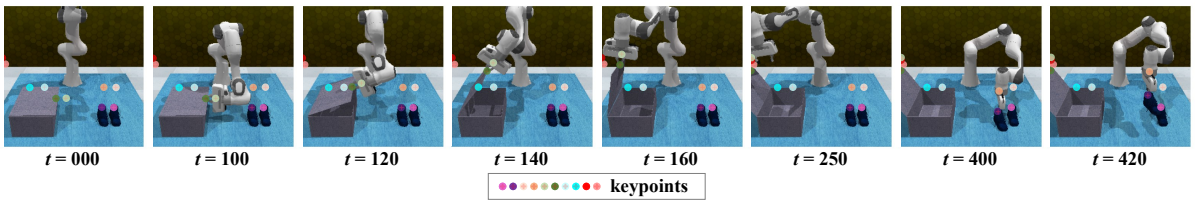


Figure 5.4 – Visualization of several frames depicting the execution of the “put_shoes_in_box" task from our DK-3D dataset, chronologically arranged. A subset of frames from the complete execution sequence is presented. t indicates the index of the frame in the sequence. Each frame showcases colored points, representing 3D keypoints capturing crucial end-effector poses. These keypoints serve as pivotal positions that the robot must successively reach to accomplish the task. Specifically, these keypoints correspond to critical moments requiring the robot to either adjust its end-effector's opening state (grasping or releasing an object) or halt the end-effector's motion, often encountered during pre-grasp poses or transitioning between task phases. The definition of these keypoints is similar to Gervet et al. [**gervet2023act3d**].

Eventually, an average pooling is leveraged to generate the final prediction:

$$\widehat{\mathbf{K}}_{3D} = \mathrm{Avg}(H) \tag{5.13}$$

where $\widehat{\mathbf{K}}_{3D} \in \mathbb{R}^{J\times 3\times 1}$ indicates the predicted 3D pose in world space, $\mathrm{Avg}(\cdot)$ denotes average pooling along the first dimension, and J is the number of skeleton joints.

5.5 Experiment

In this section, we first introduce all the benchmark datasets utilized. Subsequently, we evaluate the performance of our unsupervised pipeline, along with the proposed backbone and its components. Finally, we assess our method's adaptation capability for other scenarios with DK-3D.

5.5.1 Datasets

Human3.6M Dataset [76] provides 3.6 million synchronized images with accurate 3D body joints location. There are 15 activity scenarios in total such as “Walking”, “Eating” and “Sitting”. Each action is performed by 7 professional actors in a laboratory environment. Specifically, subjects S1, S5, S6, S7, and S8 are used for model training, and S9 and S11 for testing. A sophisticated Vicon motion capture system is used to acquire full-body 3D ground truth annota-

tions and well-calibrated intrinsic and extrinsic parameters. Both Mean-Per-Joint-Position-Error (MPJPE) and the Procrustes-analysis-MPJPE (P-MPJPE), in millimeters, are used as evaluation metrics, which respectively measure the mean 3D error before and after alignment to the ground truth in terms of translation, rotation, and scale. This is referred to as protocol 1 (MPJPE) and protocol 2 (P-MPJPE) in [**martinez2017simple**].

HumanEva-I Dataset [153] is a considerably smaller dataset with fewer subjects and actions compared to Human3.6M. Following the approach outlined in [**li2022exploiting**], our model is trained on all subjects (S1, S2, S3) and for all actions (Walk, Jog, Box).

Proposed Dataset for Scene Pose Estimation: We used the RLBench [81] robot learning environment to generate a dataset for Scene Pose Estimation dedicated to robotic manipulation. We named our dataset “Dynamic-Keypoints-3D" (DK-3D). We present Figure 5.4 as an example of the dataset, consisting of 3D keypoints corresponding to a robotic manipulation task that the robot is expected to execute. The 3D keypoints define strategic positions in the scene that the robot should reach to grasp, manipulate or place the corresponding objects while executing the target task. The 3D keypoints are defined relative to the objects of interest in the scene and are constrained to follow the movement of a parent object.

To generate the dataset, we repurposed the automatic demonstration generator integrated into RLBench, which relies on sparse sets of strategic keypoints manually defined by RLBench developers. We generated data on four tasks, illustrated in Figure 5.5, that vary in number of keypoints from 1 to 12 to evaluate the impact of keypoints sparsity on training performances. For each task, we generated 300 demonstrations, where each of them is a sequence of frames containing 2D keypoints in the four views available in the RLBench environment, along with associated 3D keypoints. We also saved images associated with each frame for visualization. The demonstrations exhibit variability in their initial scene configurations, with random placement of objects influencing the distribution of keypoints relative to the objects within the scene. It is important to mention that, in the context of scene pose estimation, we opt for a simplified approach that employs the re-projection of 3D ground truth data to 2D views of different cameras as the input for our model. To simulate the effects of errors arising from the 2D pose detector, we added Gaussian noise $\mathcal{N}(0, \sigma)$ to the 2D ground truth keypoints. Here, σ represents the average L1 error between ground truth and CPN [**chen2018cascaded**] predicted 2D keypoints on the Human3.6M dataset. The evaluation metric on our use case is the mean square error (mse) between ground-truth and predicted keypoints.

The dataset and the tools used to generate it were released, enabling investigation of multi-view 3D-Keypoint estimation in the context of scene pose estimation. One major challenge of

Table 5.1 – Relative Prediction results on Human3.6M. All of our models are trained with 7 frames unless otherwise stated, where t is the frame number for the input sequence. CPN means using [**chen2018cascaded**] to extract 2D keypoints. GT means using 2D ground truth keypoints. "-" means no information about the 2D detector used. $\star$: indicates more than 1 view augmentation as implemented in [**shuai2022adaptive**]. $\triangle$: indicates training with 27 frames. It is noteworthy that the accuracy of 2D keypoints will influence 3D estimation performance heavily. The best result in bold, second best underlined. The short names of the activity scenarios, such as Direction (Dir.), Discussion (Disc), etc., are represented at the top of each column. All results here are MPJPE unless specifically indicated otherwise.

	Dir.	Disc.	Eat.	Greet	Phone	Photo	Pose	Purch.	Sit.	SitD.	Smoke	Wait.	WalkD.	Walk	WalkT.	MPJPE	P-MPJPE
Supervised Monocular Methods																	
Liu et al.[**Liu_2020_CVPR**][CVPR'20](CPN,*t*=243)	41.8	44.8	41.1	44.9	47.4	54.1	43.4	42.2	56.2	63.6	45.3	43.5	45.3	31.3	32.2	45.1	35.6
Wang et al.[**wang_motion_2020**][ECCV'20](CPN,*t*=96)	40.2	42.5	42.6	41.1	46.7	56.7	41.4	42.3	56.2	60.4	46.3	42.2	46.2	31.7	31.0	44.5	34.5
Cheng et al.[**Cheng_2019_ICCV**](CPN,*t*=96)	38.3	41.3	46.1	40.1	41.6	51.9	41.8	40.9	51.5	58.4	42.2	44.6	41.7	33.7	30.1	42.9	32.8
Zheng et al.[**zheng20213d**][ICCV'21](CPN,*t*=81)	41.5	44.8	39.8	42.5	46.5	51.6	42.1	42.0	53.3	60.7	45.5	43.3	46.1	31.8	32.2	44.3	34.6
Wang et al.[**wang2022best**][CVPR'22](-,*t*=1)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	54.7	38.4
Wu et al. [**wu2021limb**][TIP'21](CPN,*t*=135)	39.4	43.1	38.9	39.5	43.5	47.4	35.9	39.0	49.9	56.4	42.0	39.0	42.8	31.7	32.5	**41.4**	**31.8**
Li et al.[**li2022exploiting**][TMM'22](CPN,*t*=351)	41.4	43.3	40.2	42.3	45.6	52.3	41.8	40.5	55.9	60.6	44.2	43.0	44.2	30.0	30.2	43.7	35.2
Xu et al. [**xu2021monocular**][TPAMI'21](SHN,*t*=1)	47.1	52.8	54.2	54.9	63.8	72.5	51.7	54.3	70.9	85.0	58.7	54.9	59.7	43.8	47.1	58.1	43.8
Weakly/Unsupervised Monocular Methods																	
Mitra et al.[**mitra2020multiview**][CVPR'20](GT, -)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	94.3	90.8
Drover et al.[**drover2018can**][ECCV'18](SHN,*t*=1)	58.4	59.4	58.7	64.5	59.0	60.9	57.0	61.6	85.8	60.4	64.7	57.4	63.0	65.5	62.1	62.3	64.6
Li et al.[**li2020geometry**] [AAAI'20](-,*t*=243,$\star$)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**56.1**	**43.2**
Wandt et al.[**wandt2021canonpose**][CVPR'21] (CPN, *t*=243)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	74.3	53.0
Sosa et al.[**sosa2023self**] (-,*t*=1)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	96.7
Chen et al.[**chen2019unsupervised**] [CVPR'19] (SHN,*t*=-)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	68.0	-
Wang et al.[**wang20193d**] [TPAMI'20] (SHN,*t*=-)	50.0	60.0	54.7	56.6	65.7	52.7	54.8	85.9	118.0	62.5	79.6	59.6	41.5	65.2	48.5	63.7	-
Supervised Multi-View Methods with Camera Parameters																	
Shuai et al.[**shuai2022adaptive**][TPAMI'22](CPN.*t*=7, $\star$)	23.7	26.5	24.1	25.4	28.2	29.7	23.5	25.7	29.5	37.1	26.9	24.4	27.9	23.9	23.5	**26.7**	**21.6**
Ours(CPN,*t*=7, $\star$**)**	24.0	26.3	24.0	25.2	28.6	29.0	24.0	26.1	30.4	37.9	27.3	24.9	27.7	24.2	23.9	26.9	21.9
Supervised Multi-View Methods without Camera Parameters																	
Batrol et al.[**bartol2022generalizable**][CVPR'22](CPN,*t*=7)	27.5	28.4	29.3	27.5	30.1	28.1	27.9	30.8	32.9	32.5	30.8	29.4	28.5	30.5	30.1	29.1	-
Shuai et al.[**shuai2022adaptive**][TPAMI'22](CPN,*t*=27,$\star$,$\triangle$)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**26.1**	**20.6**
Shuai et al.[**shuai2022adaptive**][TPAMI'22](CPN, *t*=7,$\star$)	24.6	25.4	24.8	24.6	28.7	29.1	23.9	25.6	31.4	36.2	26.6	24.7	28.9	23.7	23.6	26.9	21.2
Ours(CPN, *t*=27, $\star$)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	27.8	22.4
Ours(CPN, *t*=7, $\star$)	24.3	26.5	24.4	25.2	29.1	29.8	24.5	25.9	31.9	39.0	27.6	25.1	28.6	23.9	23.7	27.3	22.4
Unsupervised Multi-View Methods with Camera Parameters																	
Iqbal et al.[**iqbal2020weakly**] [CVPR'20] (-, *t*=1)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	50.2	54.5
Xu et al.[**xu2021invariant**] [AAAI'21] (GT, *t*=1)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	77.2	51.9
Hua et al.[**hua2022weakly**][TMM'23](CPN, *t*=1)	25.6	27.7	25.3	24.5	29.1	29.5	23.5	25.7	31.2	37.3	28.9	24.9	28.8	24.6	24.8	27.4	-
Unsup Shuai et al.[shuai2022adaptive](CPN, *t*=7)	47.8	79.6	40.6	43.2	48.6	88.1	45.7	39.1	49.1	54.5	60.2	48.9	87.1	49.0	44.6	55.1	36.3
Ours (CPN, *t*=7)	36.3	37.7	36.8	34.9	41.8	46.3	37.9	37.0	46.7	53.5	44.2	36.5	40.6	41.7	38.3	40.0	27.6
Ours(GT, *t*=7)	22.4	23.7	22.1	22.3	25.7	30.7	23.5	22.6	27.3	28.4	26.2	22.6	25.6	26.8	23.7	**24.9**	**16.4**
Triangulation																	
CPN, *t*=1	29.7	30.7	29.4	30.0	34.6	33.0	27.4	30.7	36.8	43.6	33.5	30.0	30.0	28.9	28.4	32.0	27.3

the proposed dataset is that part of the keypoints can reach the blind spot of some cameras during task execution, precluding the use of a naive triangulation approach. This scenario is well illustrated in Figure 5.7 in the "shoulder_left" view, where several out-of-view keypoints are croped to the right side of the image.

5.5.2 Human Pose Estimation Results

Experiment Details on Human3.6M: Our models were implemented with PyTorch 1.8.0. They were all trained using a batch size of 720, a learning rate of $1e^{-3}$, and a dropout rate of 0.1. For supervised pipeline, our models are trained for 60 epochs with a learning decay set to 0.95.

Table 5.2 – Comparison of MPJPE, Parameter Amount, and Computational Complexity on Human3.6M dataset. The best result is in bold and the second best is underlined.

Method	T	2D	MPJPE	Params (M)	FLOPs (M)
He [**he2020epipolar**]	1	IMG	**26.9**	69.0	204000.0
Qiu [**qiu2019cross**]	1	IMG	31.2	2100.0	-
Remeli [**remelli2020lightweight**]	1	IMG	30.2	251.0	-
Li [99]	27	CPN	45.9	18.92	1030.0
Gorden [**gordon2022flex**]	27	CPN	31.7	70.6	4270.0
Shuai [**shuai2022adaptive**]	7	CPN	**26.9**	<u>10.1</u>	<u>459.6</u>
Hua [**hua2022weakly**]	1	CPN	27.4	20.0	-
Ours (Supervised)	7	CPN	<u>27.3</u>	**9.6**	**412.6**

Table 5.3 – Quantitative comparisions on HumanEva-I Dataset under MPJPE (Protocol #1) and P-MPJPE (Protocal #2). Formally, let t denote the sequence length during both training and testing. GT/GT indicates training and testing with ground truth 2D pose, while Detectron/Detectron signifies training and testing using the Detectron [**wu2019detectron2**]. $\mathcal{N}(0, \sigma)$ indicates Gaussian noise with a standard deviation of σ, which is added to the intermediate triangulation result, as specified in [**hua2022weakly**]. We tested Shuai et al.'s model [**shuai2022adaptive**] on HumanEval-I [153]. Note that the results of our approach and those for [**shuai2022adaptive**] are both based on the supervised pipeline. All results here are P-MPJPE unless specifically indicated otherwise. The best result is presented in bold.

Method	Walk			Jog			Box			MPJPE	P-MPJPE
	S1	S2	S3	S1	S2	S3	S1	S2	S3		
Weakly/un-supervised methods											
Bouazizi et al. [**bouazizi2021self**], t=27	59.2	60.3	52.2	38.2	61.7	81.3	-	-	-	-	58.9
Pavllo et al. [**Pavllo_2019_CVPR**], t=243	**13.9**	**10.2**	46.6	**20.9**	**13.1**	**13.8**	**23.8**	33.7	**32.0**	-	**23.1**
Ours GT/GT, t=7	24.2	24.2	**24.2**	22.9	22.9	22.9	32.4	**32.4**	32.4	43.6	26.5
Triangulation-based methods											
Hua et al. [**hua2022weakly**] GT/GT, t=1	**2.3**	**2.2**	**2.9**	**2.3**	**2.4**	**3.8**	**2.5**	**1.3**	**7.0**	-	**3.0**
Hua et al. GT/GT+$\mathcal{N}(0, 5)$, t=1	6.1	6.1	6.4	6.1	6.2	7.1	6.4	5.4	10.5	-	6.7
Hua et al. GT/GT+$\mathcal{N}(0, 10)$, t=1	11.0	11.1	11.4	11.1	11.1	11.9	11.3	10.5	15.0	-	11.6
Hua et al. GT/GT+$\mathcal{N}(0, 15)$, t=1	16.2	16.2	16.4	16.2	16.3	16.9	16.4	15.6	19.7	-	16.7
Hua et al. GT/GT+$\mathcal{N}(0, 20)$, t=1	21.3	21.3	21.6	21.3	21.4	22.0	21.4	21.0	24.4	-	21.7
Supervised methods											
Shuai et al. Detectron/Detectron, t=7	30.5	31.0	29.5	28.6	27.8	28.6	44.6	42.8	42.8	38.5	34.0
Ours GT/GT, t=7	**15.8**	**15.9**	**16.4**	**15.8**	**15.5**	**16.3**	**24.8**	**26.1**	**26.7**	**23.1**	**19.3**
Ours Detectron/Detectron, t=7	24.3	24.4	24.9	19.3	19.3	19.6	33.3	32.3	32.7	29.3	25.6

Our unsupervised model is trained with a learning decay of 0.99 for 300 epochs. During training phase, we use MPJPE as supervised loss and Mean Squared Error (MSE) as unsupervised loss to optimize the models.

Experiment Details on HumanEva-I: Generally, the same training configuration as used for Human3.6M was adhered to. However, in contrast to Human3.6M, a batch size of 360 was employed, and training was conducted for 1000 epochs on HumanEva-I, with a receptive field of 7 frames.

Quantitative and Qualitative Evaluation: Table 5.1 reports the quantitative results of various models on Human3.6m dataset. All the benchmark methods are categorized into five groups: supervised monocular methods, weakly/un-supervised monocular methods, supervised multi-view methods with/without camera parameters, and unsupervised multi-view methods with camera parameters. Specifically, 'with camera parameters' means that calibrated extrinsic and intrinsic parameters are accessible during training.

From Table 5.1:

i.) In terms of Supervised Multi-View methods with/without Camera Parameters, our methods surpass most state-of-the-art methods while are slightly worse than [**shuai2022adaptive**]. We assume that this is because our method does not include the human-specific designs found in [**shuai2022adaptive**]. Such features, while potentially improving performance, might hinder generalization across object categories.

ii.) Our method outperforms all state-of-the-art unsupervised multi-view methods with camera parameters when using ground truth 2D poses (GT) as input. When using CPN as a 2D pose detector, our method is inferior to Hua et al.[**hua2022weakly**] but superior to other approaches. Considering Hua et al.'s requirement of camera parameters during both training and testing phases to provide solid prior knowledge, our method's flexibility and practicality are highlighted by not requiring camera parameters during inference.

iii.) Our method is slightly inferior to triangulation, which heavily relies on camera parameters, similar to Hua et al.[**hua2022weakly**]. This reliance makes it sensitive to calibration parameters: if the camera parameters change, triangulation will fail completely. With only an 8mm increase in error, our unsupervised method, which does not require camera parameters during inference, is more flexible and practical compared to triangulation.

iv.) Furthermore, our model demonstrates robustness to variations in multi-view camera setups, as it is trained and tested on different Human3.6M subjects, corresponding to different extrinsic parameters. In other words, there's no need to calibrate again if the cameras position is perturbed. Further limitations of triangulation will be highlighted in next section.

v.) The qualitative results of our supervised and unsupervised pipeline are presented in Figure 5.6. Overall, our method estimates high-quality 3D human poses in both supervised and unsupervised processes. However, they show biases in high degree-of-freedom joints like hands, feet, and head. In the last row, the right arm prediction in view 3 under the unsupervised process deviates notably from the ground truth, and significant deviations are seen in head predictions in views 3 and 4. In contrast, the supervised process predictions are more accurate.

Table 5.2 compares MPJPE, parameters amount and computational complexity with several state-of-the-art methods on Human3.6M dataset. Our model has approximately 5% fewer parameters and about 10.2% less computational complexity compared to the state-of-the-art method of Shuai et al. [**shuai2022adaptive**], yet it only falls behind by approximately 1.5% in terms of MPJPE. This is due to that our Temporal encoding module, TPT, is both lighter and more computationally efficient than the one presented in [**shuai2022adaptive**]. Additionally, the human-specific calculations in [**shuai2022adaptive**] also introduce redundant parameters and increase computational complexity.

Table 5.3 is the quantitative comparison on HumanEva-I [153] dataset. Given that existing off-the-shelf 2D pose estimation models, such as Detectron [**wu2019detectron2**], CPN [**chen2018cascaded**] predict a 17-joint skeleton for 2D poses, while the 3D ground truth of HumanEva-I consists of a 15-joint skeleton, a significant disparity arises between the 2D input and the 3D target. Consequently, applying our unsupervised pipeline to HumanEva-I becomes challenging, as our pipeline depends on the geometrical consistency between 3D space and 2D space. So, most of the weakly/un-supervised methods can only take the ground truth 2D pose as their input. Table 5.3 is analyzed from three perspectives:

i.) Compared with weakly/unsupervised methods, our approach trained in unsupervised pipeline achieves competitive results compared to Pavllo et al. [**Pavllo_2019_CVPR**]. Pavllo et al. [**Pavllo_2019_CVPR**] incorporates human bone length as prior information for training, whereas our method performs without reliance on any such prior information within the dataset. While this type of prior can effectively enhance the model's performance, it may hinder its ability to generalize to other scenarios.

ii.) Hua et al. [**hua2022weakly**] involves a two steps: 1) triangulation to obtain a 'coarse' 3D pose, and 2) refining the 3D pose using an auto-encoder architecture. Our method is slightly inferior to Hua et al.[**hua2022weakly**] because this kind of triangulation-based approach relies on camera parameters in both training and testing phases to establish solid geometric priors, while ours does not require camera parameters during testing. Additionally, while triangulation yields nearly perfect results with ground-truth 2D pose, this method proves highly sensitive

to errors in the 2D inputs. To address this, Hua et al. introduced various levels of noise after triangulation to simulate the effects of erroneous 2D inputs.

iii.) In supervised phase, we need to note that the model proposed by Shuai et al. [**shuai2022adaptive**] incorporates a human-specific module designed based on the Human3.6m human skeleton, it is challenging to apply HumanEva-I's 15-joint ground truth 2D pose as input for testing the model's performance. As a result, we are not able to report Shuai et al.'s results when trained with ground truth 2D pose in HumanEva-I. In contrast, our method does not include a human-specific module, allowing it to seamlessly accept a 15-joint ground truth 2D pose as input. In terms of the condition of "Detectron/Detectron", our method surpassed the baseline [**shuai2022adaptive**] by 9.2 mm and 8.4 in terms of MPJPE and P-MPJPE, respectively. The possible reason is that our HCT models spatio-temporal feature interactions at different levels. In contrast, Shuai et al.'s method focuses on multi-view feature fusion and uses cascaded Transformer Blocks for single-level spatio-temporal modeling, making it difficult to balance the contributions of different feature levels to the final prediction.

Ablation evaluation for proposed modules in HCT: Table 5.4 studies the effectiveness of our proposed modules "TPT" and "CFT". "MFT" is adapted from [**shuai2022adaptive**]. We trained all the models with 4 views but tested them with different number of views. When tested with 4 views, our model achieves a MPJPE of 34.7 mm in the absence of the TPT module. By integrating the TPT module with 3 layers of encoders, our model has approximately 22.4% MPJPE reduction. This improvement underscores the importance of multi-level spatio-temporal context for 3D pose estimation.

When tested with 4 views, our model equipped with the TPT module (4 layers of encoders) achieves an MPJPE of 27.5 mm without the CFT module. With both the 4-layer TPT and the CFT modules, the MPJPE is reduced to 26.9 mm, demonstrating the effectiveness of the CFT module.

To demonstrate the effectiveness of MFT, we evaluated our model's performance with and without MFT, using a 3-layer TPT backbone and CFT. When tested with 4 views, our model achieves an MPJPE of 26.9 mm with MFT and 36.9 mm without MFT. This demonstrates that explicit multi-view feature fusion is crucial for accurate 3D pose estimation. Furthermore, we observe an obvious degradation when the number of testing views decreases. This indicates that more complementary information from multiple views leads to better performance of the proposed method.

Evaluating the impact of varying view numbers on model performance: Table 5.5 studies the robustness of our approaches while adjusting the number of views during both training

Table 5.4 – Ablation analysis on Human3.6M for the proposed modules within our **HCT**, l indicating the number of Transformer Encoders in the proposed **TPT**. All models are trained using 2D poses predicted by CPN from 4 views as input, each with a sequence length of 7. The best result is presented in bold.

Method	TPT	CFT	MFT	MPJPE				P-MPJPE				Params (M)
				Number of views during testing								
				1	2	3	4	1	2	3	4	
Ours	✗	✓	✓	78.7	55.5	37.6	34.7	58.3	37.9	31.1	28.7	8.1
Ours(l=4)	✓	✗	✓	56.3	46.6	30.4	27.5	44.3	30.1	25.1	22.6	**6.3**
Ours(l=4)	✓	✓	✓	55.8	**32.7**	30.9	**26.9**	44.1	**28.2**	**23.3**	**21.9**	10.3
Ours(l=3)	✓	✓	✗	56.0	54.2	40.2	36.9	42.6	36.7	33.8	31.8	9.1
Ours(l=3)	✓	✓	✓	53.3	45.6	**29.7**	**26.9**	**41.7**	28.6	24.0	22.1	9.6

and testing. Similar to [**shuai2022adaptive**], our model's MPJPE consistently increases as the number of views decreases under a supervised setting, indicating that more views provide more complementary information for the model. However, as expected, our unsupervised approach is affected by reducing the number of views because the supervision signal solely relies on the 2D consistency averaged over multiple views. Moreover, our method is sensitive to the disparity between the number of training and testing views, as particularly evidenced when comparing {training with 4 views, testing with 1 view} scenario to {training with 2 views, testing with 1 view} scenario. This phenomenon is exacerbated by the unsupervised pipeline. We suspect this is because our method cannot leverage human-specific priors to correct the erroneous fusion of multi-view features.

Ablation evaluation for the scale of the TPT: An ablation study was conducted on Human3.6M to assess the impact of varying the number of Transformer Encoder layers in TPT on both the ultimate performance and parameter volume. As the number of transformer encoder layers in TPT increases, our model scales up, as evidenced by the growth in parameter volume. Table 5.6 demonstrates that increasing the number of transformer encoder layers benefits the model's performance. This suggests that incorporating more features from different levels can provide more semantic information for 3D pose estimation.

Ablation evaluation for feature pooling strategies: In our unsupervised pipeline, multiple hypotheses of 3D poses are generated by the model. These poses are then pooled using a specific strategy, as illustrated in Figure 5.2. The results of different pooling strategies, including Max, Average, Add, and Concat, are presented in Table 5.7. The results indicate that the Average pooling achieves the best performance. The poor performance of "Max" may be due to gradients being propagated based on the maximum value's position during back-propagation, leading to

Table 5.5 – Investigating the impact of view count on Human3.6M performance with MPJPE, with fixed training and testing length of 7. 'Unsup Shuai et al.' denotes the incorporation of Shuai et al.'s backbone into our unsupervised pipeline. FLOPs computed using input with 4 views and length 7.

Training Views	**2**		**3**			**4**				**FLOPs (M)**
Testing Views	**1**	**2**	**1**	**2**	**3**	**1**	**2**	**3**	**4**	
Shuai et al.(Sup)	52.5	46.8	**49.8**	46.0	**29.5**	**46.5**	43.6	**28.7**	**26.7**	459.6
Ours (Sup)	**51.9**	**32.5**	50.7	**32.5**	30.6	55.8	**32.7**	30.9	26.9	**412.6**
Shuai et al. (Unsup)	**174.3**	85.8	**146.7**	**96.8**	70.5	**117.3**	**80.6**	**65.2**	55.1	459.6
Ours (Unsup)	224.6	**78.3**	197.4	212.6	**52.4**	222.1	164.6	194.7	**40.0**	**412.6**
Triangulation	62.3		34.5			32.0				-

loss of correlation between multiple hypothesis predictions. The "Add" method closely matches average pooling, indicating effective utilization of prediction correlations. The "Concat" performs similarly but slightly increases parameters and computation, suggesting challenges in modeling prediction correlations and optimization.

Table 5.6 – An ablation study on Human3.6M was conducted to assess the impact of varying the number of Transformer Encoder layers in TPT on both the ultimate performance and parameter volume. l denotes the number of Transformer Encoder in the TPT.

Method	**TPT**	**MPJPE**	**P-MPJPE**	**Params(M)**
Ours(CPN, l=0)	✗	34.7	28.7	**8.1**
Ours(CPN, l=2)	✓	32.1	24.9	9.0
Ours(CPN, l=3)	✓	26.9	22.1	9.6
Ours(CPN, l=4)	✓	**26.9**	**21.9**	10.3

5.5.3 Scene Pose Estimation

Experiment Details: On the DK-3D dataset, all models are trained for 60 epochs using batch size, learning rate, learning rate decay, and dropout rate, respectively set to 720, $1e^{-3}$, 0.99, and 0.1.

Quantitative and Qualitative evaluation: Table 5.8 details the results on our DK-3D dataset. The DK-3D setup first highlights a major drawback of the triangulation approach. Tri-

Table 5.7 – Pooling strategies comparison to aggregate generated 3D poses: We compared pooling strategies using our unsupervised model in the table below. Specifically, we introduced an additional 2D convolutional layer to aggregate the generated 3D poses during the "Concat" strategy.

	Max	**Average**	**Add**	**Concat**
MPJPE	53.4	**40.0**	40.8	40.8
P-MPJPE	34.6	**27.6**	28.0	28.3

Table 5.8 – 3D Keypoints Prediction Error on DK-3D dataset(error unit: mm).

	Pack.	**Money.**	**Fridge.**	**Reach.**
Ours				
Supervised	3.07	2.00	8.12	7.64
Unsupervised	7.64	3.70	8.25	6.67
Triangulation				
	24.6	4.70	141.20	8.71

angulation completely fails on tasks "put_shoes_in_box" and "close_fridge", namely "Pack" and "Fridge" in Table 5.8. As illustrated in Figure 5.5, these tasks include keypoints that reach some of the camera's blind spots during the manipulations, while for tasks "reach_target" and "take_money_out_safe", namely "Reach" and "Money", the target objects are always centered in all the cameras. Triangulation is a geometric approach to determine 3D points from their actual 2D positions onto multiple cameras. Therefore, as a "fixed approach", triangulation can not learn exception cases, such as blind spots. On the contrary, learnable approaches can leverage knowledge acquired from a dataset of examples to alleviate such exceptional cases. Both supervised and unsupervised approaches got similar results on the DK-3D dataset regarding the unit considered in Table 5.8. Interestingly, the two tasks that contain extremely sparse keypoints, "reach_target" and "close_fridge", got worst results than the two other tasks with supervised approach, which suggests that our model leverages inter-points information, which we represented as scene skeleton in Figure 5.1.

5.6 Conclusion

In this chapter, we examined both existing and proposed supervised and unsupervised methods for 3D human pose estimation. Table **??** summarizes the Mean Per Joint Position Error

(MPJPE) and Procrustes-aligned MPJPE (P-MPJPE) for different methods on the Human3.6M dataset and our dataset (with 16 cameras). While unsupervised methods yield decent P-MPJPE results, their MPJPE results significantly lag behind those of supervised methods, indicating that while relative joint positions are well-predicted, absolute positions are not.

For ergonomics assessment, where relative joint positions are the crutial information, unsupervised methods may suffice. However, for applications like action recognition or modeling human joint trajectories, which require accurate tracking of absolute joint positions over time in the world coordinate system, supervised methods are more suitable due to their superior performance in predicting absolute positions.

Supervised models, however, struggle with generalization to novel multi-view configurations, particularly in scenarios with occlusions. These models must be retrained for each new multi-view setup the maintain performance levels shown in Table **??**, and require access to 3D ground truth poses during training.

For limited datasets featuring a small number of subjects, calibrating cameras and triangulating 3D poses with the method proposed in Chapter 4 seems to be the most effective approach. For larger datasets, a hybrid approach where a model is trained on a subset of annotated data (again, using method proposed in Chapter 4) and then deployed on the remaining data would be more effective. Similarly, unsupervised methods with access to camera calibration parameters can be trained on a subset of data with calibrated cameras, before being applied to the entire dataset without requiring calibration parameters.Compared to the proposed triangulation based approach (Chapter 4), our unsupervised method with access to the calibration parameters of the cameras doesn't require access to the bone length of the subjects, which is a significant advantage.

Compared to basic triangulation (which does not leverage the bone length of the subjects), our unsupervised method with camera calibration provides slightly better results. However, this improvement may not always justify the increased complexity.

Notably, our unsupervised method for uncalibrated cameras, despite its current limitations, shows the most promise for real-life applications where cameras are not fixed, calibration parameters change over time, and 3D ground truth poses are unavailable.

In scenarios with limited occlusions, using 3D cameras like stereo cameras or depth sensors for human pose estimation is a viable alternative. This approach is particularly appealing for real-life applications where deploying multi-view setups is challenging and cameras are not fixed. However, it is less robust to occlusions and typically more expensive than multi-view setups with a moderate number of cameras.

Overall, the choice of method depends on the specific requirements of the target application, including the importance of relative versus absolute joint positions, the ability to handle novel multi-view configurations, and the feasibility of obtaining calibration parameters and 3D ground truth data.

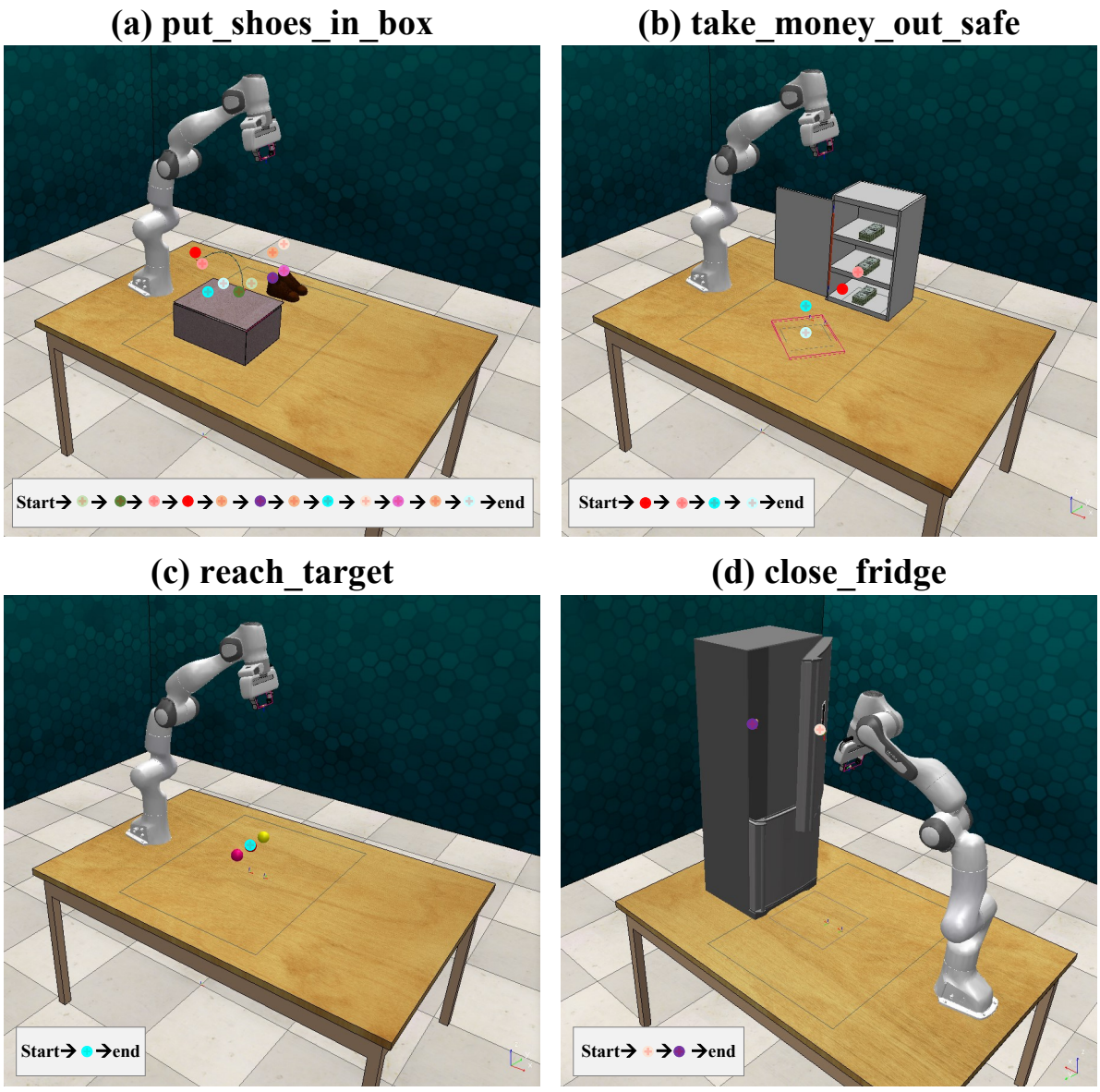


Figure 5.5 – Visualization of the 4 RLBench tasks chosen for our DK-3D dataset. (a) “put_shoes_in_box" contains 12 keypoints. (b) “take_money_out_safe contains 4 keypoints. (c) reach_target contains 1 keypoint. (d) close_fridge contains 2 keypoints.

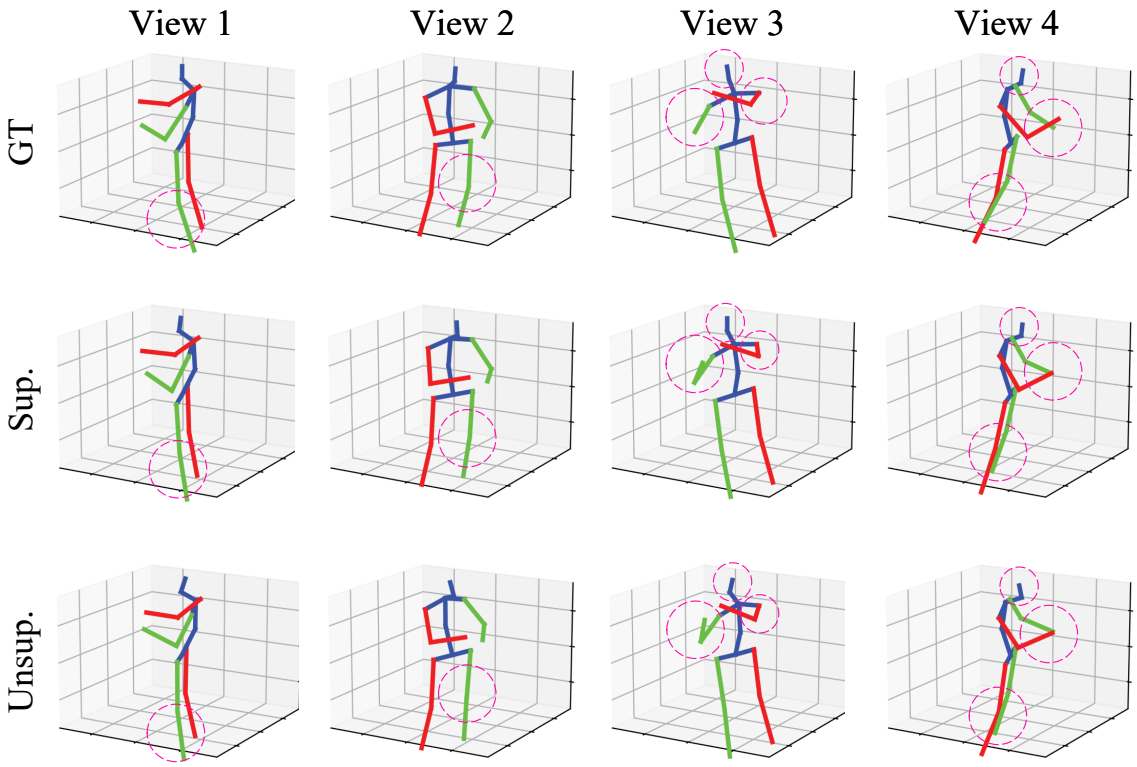


Figure 5.6 – Qualitative results of the prediction of our model in supervised and unsupervised training on Human3.6m. Noticeable prediction errors are enlarged.

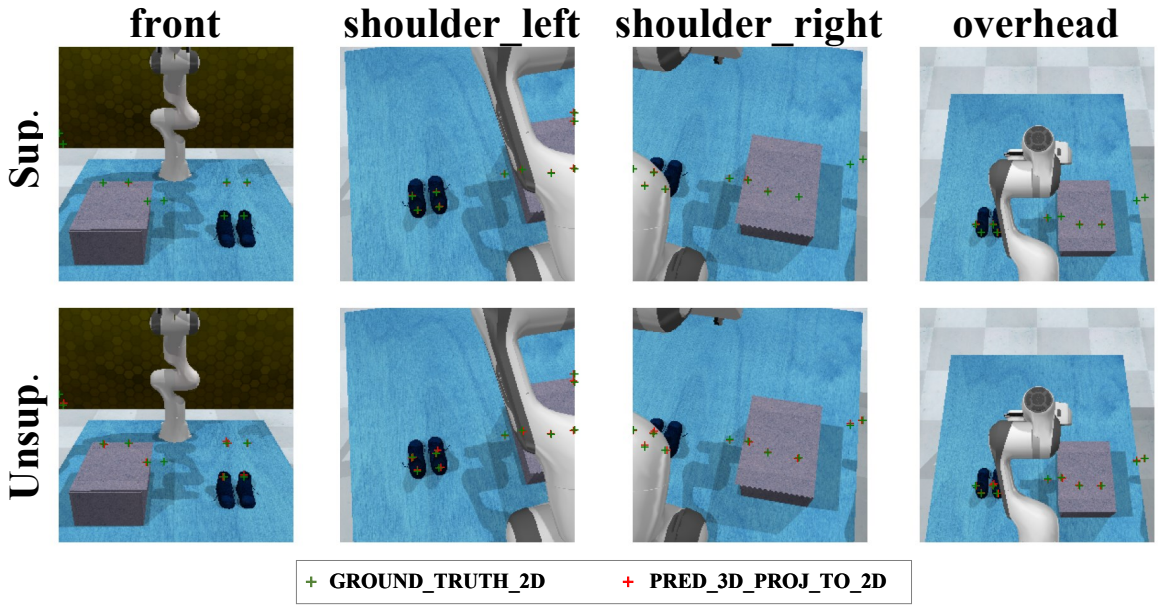


Figure 5.7 – Qualitative results of the prediction of our model in supervised and unsupervised training on DK-3D dataset.

CHAPTER 6

ACTION RECOGNITION FROM HUMAN POSE

PART III

Learning a task from demonstrations

Overview

Currently, most of the existing robots are manually programmed by human experts to perform a specific task, in a specific environment. Each change in the task or the environment requires a new programming phase, which is time-consuming and expensive. Robotic Manipulation Learning, i.e., the ability for a robot to autonomously learn various manipulation skills from demonstrations or interaction experiences, is a crucial approach for an autonomous system. Two paradigms are well-studied for developing this approach: reinforcement learning (RL) and behavior cloning (BC). Behavior cloning attempts to learn a mapping from observations to actions, thus mimicking the expert's behavior. Behavior cloning policies are thus trained on datasets of observation-action pairs, usually collected by teleoperating the robot. With reinforcement learning, the robot learns a policy by interacting with the environment and receiving rewards. The policy takes a series of decisions based on environmental observations, and is rewarded based on the outcome of the decisions. Although impressive progress has been achieved by reinforcement learning, it still requires a large number of interactions with the environment, which can not be realistically achieved in many real-world scenarios, unless a simulation of the environment is available. Moreover, as illustrated in Chapter 3, the reward function definition is a challenging task, and can be infeasible in some cases. Behavior cloning on the other hand does not require any reward function nor interaction with the environment. However, BC has some inherent limitations that will be discussed in Chapter 7, where we further propose a solution to mitigate part of these limitations. Moreover, Behavior Cloning requires a dataset of expert demonstrations, which are usually collected by teleoperating the robot. Such teleoperation can be time-consuming and tedious, especially for complex tasks. In particular, Mandlekar et al. [117] showed that the expertise of the demonstrator in teleoperation can significantly impact the performance of the learned policy. In Chapters we'll investiguate how to collect training data for BC from videos of human demonstrations, which eliminates the need for teleoperation. In Chapter , we'll show how to train a BC model on the collected data, and deploy it on a robot.

CHAPTER 7

A KEYPOINTS-BASED LEARNING PARADIGM FOR VISUAL ROBOTIC MANIPULATION

Overview

Contents

Part of this chapter is based on the research paper [142].

7.1 Introduction

7.2 Literature Review

In this section, we will revise primary knowledge from three aspects of Visual Robotic Manipulation: Reinforcement Learning (RL), behaviour cloning (BC) and keypoint-based manipulation learning.

7.2.1 Reinforcement Learning (RL) for manipulation tasks.

Online Reinforcement Learning highly relies on iterative interactions between agent and environment, which is impractical in real-life scenarios due to expensive or dangerous experience collection. The success of Deep Neural Network (DNNs) has promoted the advent of offline Reinforcement Learning for large-scale data representation. Kumar *et al.* [94] proposed a practical algorithm to reduce error accumulation when training from offline data. Ebert *et al.* [35] presented a model-based reinforcement learning method centered around prediction of raw sensory observation by taking use of prediction in the context of robotic manipulation. However, both online and offline approaches focus on learning a direct mapping from environment states to robot actions, which means (1) they are not purpose-aware and (2) they have to learn primitive skills that roboticists can easily solve using off-the-shelf solutions. Primitive based Reinforcement learning is an augmentation for standard Reinforcement Learning with a predefined library of behavior primitives learnt from data, which is more robust and reusable for alleviating the generalization challenge. Strudel *et al.* [158] designed a sample efficient pipeline to learn robust RL policies confined with primitive skills. Nasiriany *et al.*[120] introduced a Deep Reinforcement Learning (DRL) framework which utilized predefined hierarchical primitive skills, to narrow the exploration space of DRL. Dalal *et al.* [28] manually coded primitives with arguments that are learned by RL policy, leading to a better adaptability for the considered tasks.

7.2.2 Behavior cloning for manipulation tasks learning.

Benefit from the promising development of Deep Neural Network (DNN) over the past decade, the effectiveness of data-driven learning methods have been proved over many visual sensory applications[160]. Behavior cloning is a straightforward imitation learning method that utilizes the representation capability of DNNs to map the expert's actions to the agent's ob-

servations using abundant data. It does not require the agent to interact with the environment during training, making it a simple approach. However, behavior cloning suffers from several limitations, such as not being purpose-aware, having to learn primitive skills that can be easily solved using off-the-shelf solutions, and well-known error accumulation during task execution. To overcome these issues, researchers have developed an alternative approach called Hierarchical Behavior cloning (HBC), which involves a high-level model that learns intermediate goals and a low-level model that predicts sequences of actions to reach sub-goals, as demonstrated in studies such as [193, 32, 192]. HBC uses an end-to-end framework that significantly reduces the error accumulation and enhances purpose-awareness. Nevertheless, HBC still needs to learn some primitive skills that roboticists can easily solve using off-the-shelf solutions.

7.2.3 Keypoint-based manipulation learning

As a task-relevant context on object, keypoints of object surface come with a significant importance for Visual Robotic Manipulation (VRM). With the progressive development of object keypoint discovery approaches [160, 116, 93, 162], many manipulation learning methods have been proposed to leverage keypoints-based context for geometry and semantic representations. But most of these methods leverage keypoints as a bridge for facilitating the transfer of manipulation skills between demonstrators and imitators. Gao *et al.*[43] decomposition robotic manipulation task into keypoints constrain representation and control policies selection, which allows the successful reproductions across various tasks. Yang *et al.* [194] utilized keypoints detector to represent the similarity between a human demonstration and robot execution, then maximized it by Bayesian optimization. Wada *et al.* [175] represent 6D pose of objects in voxels to improve multi-object reasoning in cluttered scenes. Kulkarni *et al.* [93] proposed a unsupervised keypoints discovery network, then apply the learned object keypoints as state input that related to policy explorations over Reinforcement Learning (RL) settings.

7.3 Background on Behavior Cloning

7.3.1 Behavior Cloning for Visual Robotic Manipulation

Behavior Cloning (BC) refers to a supervised learning approaches used to learn sensorimotor policies from offline data. BC only requires pairs of sensory observations, e.g., images, associated to expert actions. Given access to an expert agent, we can build a BC dataset, $D = \{(o_i, a_i)\}_{i=1}^{N}$, where o_i are sensory observations, and $a_i = \pi^* (o_i)$ are the respective actions taken

by the expert π^*. When using BC for robotic manipulation, actions are usually described as the Cartesian-position of next gripper Tool Center Point (TCP) and Quaternion-orientation, in the scene, also associated with the gripper state, e.g. opening amount, $a_i = (x_{tcp}, y_{tcp}, z_{tcp}, q_{1_{tcp}}, q_{2_{tcp}}, q_{3_{tcp}}, q_{4_{tcp}},$ Typical solutions to acquire such expert actions consist of robot teleoperating by human, robot behavior hard-coding with omniscient knowledge in simulation, expert RL policy training using hard-coded reward function as well as huge amount of simulation trials.

When considering visual robotic manipulation, the observations are limited to raw images of the environment $o_i = \{I_i^v\}_{v=1}^V$, where I_i^v represents the image from view v in a multi-view configuration. The goal of a BC algorithm is to learn a policy π, parameterised by θ, that produce similar actions to the expert π^* when provided with the same observation o_i. Optimal parameters θ^* are found by minimizing the BC loss l :

$$\theta^* = \arg\min_\theta \sum_i l\left(\pi\left(o_i; \theta\right), a_i\right)$$

At inference phase, the trained policy is used to sequentially predict actions from observations to execute the target task.

7.3.2 Limitations

BC is used to solve a sequential decision problem, where future observations depend on previous actions. At training phase, it violates the i.i.d. assumption made in statistical learning. Moreover, at inference phase, errors of the action predictions accumulate along the task execution, which leads to a distributional shift between training and inference observations, which subsequently results in out-of-distribution prediction.

To alleviate this problem, hierarchical algorithms has been proposed to decompose trajectories of manipulation task into sub-trajectories that mostly consisted in existing off-the-shelf robotics primitives, e.g. "reaching" and "grasping". However, both traditional BC and RL still have to learn by themselves to accomplish the target tasks.

7.4 Predicting a sequence of keypoints for visual robotic manipulation

Instead of considering (observation, action) tuples to train our model, we format the dataset as a set of trajectories, $D = \{T_i\}_{i=1}^N$, in which each trajectory can be represented as $T =$

$\{(o_t, a_t)\}_{t=1}^{T}$.

We can then further decompose a task as sequence, $T' = \{(o_m, wp_m)\}_{m=1}^{M}, M << T$ made of a few waypoints wp_m that can be sequentially reached using off-the-shelf robotics primitives. Fig. 3.1 illustrates the observation associated to 4 waypoints which define a "cube lifting" task.

Given these statements, we designed a model that can directly predict such set of waypoints from the first observation of the scene. In this scenario, the training dataset D can be formatted as a set of tasks $T_i = (o_0^i, \{wp_0^i, ..., wp_M^i\})$, where the few waypoints needed to accomplish the task must be directly predicted from o_0. This approach benefits from several advantages. First, the sequential decision making problem solved by traditional BC or RL approaches is changed to a one-step prediction problem, where the whole trajectory is predicted from initial observation of the scene, which eliminates both accumulation error during inference and the violation of the i.i.d. assumption during the training phase. Secondly, the trained model focuses on understanding the high level structure, i.e. the purpose of the task, while sub-trajectories prediction is remained for existing and more robust robotics primitives.

This section details the framework proposed to solve the one-step trajectory prediction problem mentioned above.

7.4.1 Overall framework description

Fig.5.2 describes the whole proposed framework. The framework is made of two main components: (1)The deep learning model, predicting waypoints from a multi-view images of the initial observation, denoted as Multi-View Fusion Transformer (MFT) in Fig.5.2; (2)A path planner, that can drive the robot to the predicted waypoints.

The waypoints prediction framework can be divided in two parts: (1)The image processing part aims to inherently learn to locate the waypoints in the 2D images; (2)The 2D-to-8D projection part that learns to map the muli-view 2D information extracted by the the image processing part to actual 8D robot poses $(x_{tcp}, y_{tcp}, z_{tcp}, q_{1_{tcp}}, q_{2_{tcp}}, q_{3_{tcp}}, q_{4_{tcp}}, Grip.State)$ that can be understood by the robotics path planner. The whole framework is trained end-to-end from ground truth 8D waypoints recovered from expert demonstrations. The method used to choose and extract the waypoints from the demonstrations is detailed in section 7.5.1.

7.4.2 Framework details

The architectural choices for the framework parts were guided by the actual functions they had to satisfy. The aim of the image embedding module is to extract pertinent characteristics

from various individual views. These features are later fused by the 2D-to-8D projection model to recover the final 8D waypoints. Furthermore, we can state that the image embedding module aims to extract information about the 8D waypoints projected in various 2D views. The 2D-to-8D projection module can be interpreted as a sophisticated learned triangulation method that projects 2D spatial information into the 3D scene space.

Image Embedding

We took inspiration from a cutting-edge deep learning architecture for 2D human pose estimation, called HRNet [160], to develop our image embedding module. HRNet has the capability of generating feature maps with high spatial resolution as well as rich semantic information. This combination is critical for learning manipulation tasks, particularly those involving small objects, as it provides a comprehensive understanding of the objects in the scene. In this work, we employ the HRNet-W32 [160] architecture as the feature extraction backbone, which will consequently generate M-channels feature maps for each view in 2D space.

MFT

Our Multi-view Fusion Transformer (MFT) is a simple Transformer-based network which aims to encode the latent waypoints features in 2D space, to 8D waypoints in 3D space, as shown in 7.1. Firstly, the latent feature map of each view is flattened to M 1-D tensor, which are then fed into a multi-layer perceptron (MLP) to obtain implicit features for 8D waypoints. The shape of these features is $M \times 16$, where M is then number of waypoints, and 16 is the latent dimension of the features. After the pre-embedding step, the generated features from each view are processed using our proposed MFT module, which includes learnable spatial embedding. Subsequently, four implicit vectors are produced after passing through a residual MLP layer, followed by four transformer blocks in parallel, with a cross-connection configuration[99]. The residual connection is also utilized in this block to moderate gradient vanishing. Finally, the resulting outputs are concatenated and passed through MLP layers to generate the final output.

Training details

The whole framework is trained end-to-end to minimize the following supervised loss.

$$L = \sum_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \left(w_m^i - w_m^{i^*} \right)^2$$

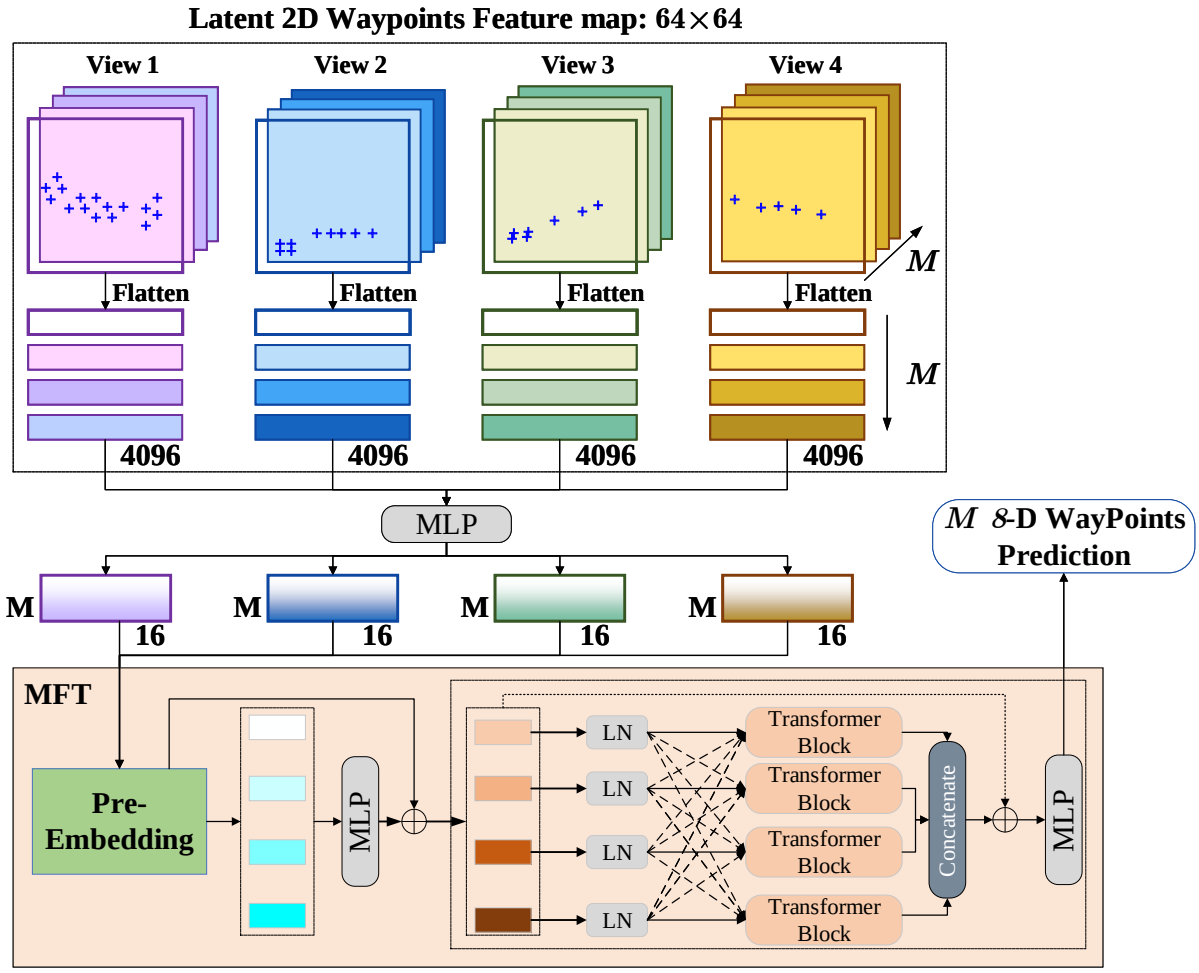


Figure 7.1 – Framework of our MFT module. A high-level implicit representation with shape of $M \times 16$ will be obtained from the latent 2D waypoints feature map through a flatten operation and MLP layers, prior to its transmission to MFT. Subsequently, these implicit representations will be mapped into vectors with same shape of input through a pre-embedding and a residual MLP layer. Taking generated multi-view vectors as input, our 4 branches cross-transformer will enable the interaction among multi-views features. Besides, a residual connection is also used to alleviate the gradient vanishing.

, where $\{w_m^i\}_{m=1}^M = \pi\left(o_0^i; \theta\right)$. π represents the framework that predicts M waypoints from the initial observation of the scene o_0; w_m^i and $w_m^{i^*}$ represent respectively the m-th predicted and ground-truth waypoints for the i-th trajectory in the training dataset.

Path Planning

As we conducted our experiments in RLBench [81] simulation, we used the integrated 'ABS_EE_POSE_PLAN_WORLD_FRAME' path planer, which can drive the robot TCP to a specified absolute position and with the specified orientation in the world frame. This path planner relies on the Open Motion Planning Library [159] combined with a feedback-control loop. The gripper control is also integrated to the path planer and is discretized to either 'open_gripper' or 'close_gripper'. The planning always operates in two steps, (1) driving the robot TCP to the desired position, and (2) updating the gripper state.

7.5 RESULTS

7.5.1 Simulation

We conducted our experiments using RLBench [81] simulation because (1) it provides a multi-camera working environment, (2) it gives access to a wide variety of tasks, and most importantly, (3) within the simulation, an automatic task demonstration generator is integrated. This generator utilizes sets of pre-defined waypoints for each available task to create custom waypoint-based datasets that can be used for further analysis and training. As a result, we propose a tool that can generate datasets specifically tailored to the novel paradigm proposed in this work using RLBench. The generated datasets comprise demonstrations of tasks, consisting of initial observations of the scene, and their corresponding "ground truth" sequences of 8D waypoints necessary for successfully completing the respective tasks. As illustrated in Table 7.1, the automatic demonstration generator provides almost two orders of magnitude more steps than the number of waypoints that are actually needed to accomplish the task.

The state of the simulation is also saved for each generated demonstration, so that the tasks can be later replayed with the exact same scene configuration. By leveraging this feature, an evaluation of the trained framework can be executed in three steps. (1) A part of the generated dataset is kept apart for evaluation during the framework training. (2) Once the training is done, the evaluation data are inferred by the trained model, which will predict waypoint sequences corresponding to the evaluation examples. (3) Simulation is iteratively loaded with

	Trash.	Reach	Grill	PickLift	Unplug	MoneyOut
M	4	1	5	4	3	4
T_{max}	564	83	399	460	343	306
T_{min}	117	22	108	103	97	162

Table 7.1 – Number of waypoints (M) vs min and max total number of steps (respectively T_{max} and T_{min}) when using RLbench demonstration generator. The tasks Trash., Reach, Grill, PickLift, Unplug and MoneyOut correspond to RLBench tasks considered in our experiments, respectively 'put_rubbish_in_bin', 'reach_target', 'meat_off_grill', 'pick_and_lift', 'unplug_charger' and 'take_money_out_safe'.

scene configurations that correspond to the evaluation data. The predicted waypoints generated from these configurations are then utilized by the path planner in an attempt to accomplish the assigned tasks. A success rate can be computed over the evaluation data by reporting the number of successful examples over the total number of examples.

7.5.2 Is our approach competitive with existing solutions for visual-robotic-manipulation learning ?

To compare our approach to existing robotic manipulation learning solutions, we referred to James *et al.* [80], who ran two sets of experiments in their work. In the first set of experiments, they compared common imitation learning and reinforcement learning methods , including ARM [79], BC, SAC+AE [198], DAC[89], SQIL [135] and DRQ [88], and their baseline, C2F-ARM, [80] , on a set of tasks that can be solved from the front camera only. In a second set of experiments, they evaluated ARM [79] and C2F-ARM [80] on tasks that require more than one camera to be achievable.

ARM, C2F-ARM, SAC+AE, DAC, SQIL and DrQ are RL-based approaches, which means they require thousands of interactions with the environment to learn a task. James *et al.* [80] provided all baselines, including RL ones, with 100 demonstrations for each task, except for their C2F-ARM baseline, which was provided with 10 demonstrations. However, they used a data augmentation method to extend their data. Since we didn't use data augmentation strategies in our case, we provide our model with 300 demonstrations. However, in section **??**, we discuss how our paradigm allows considerably more efficient data collection than previous imitation learning paradigms.

As a proof of concept, we selected, (i) two tasks in the first set of experiments, to validate the performances of our framework compared to a wide range of robot manipulation learning

methods, and (ii) two tasks in the second set of experiments to validate that our approach can compete with existing multi-view approaches.

We point out that in both sets, the authors had access to both RGB and depth information from the considered cameras, while our method learns to recover depth information from RGB images only, using our multi-view fusion module. For this reason, our framework always considers the four cameras placed around the scene in the simulation, namely 'front', 'overhead', 'over-shoulder-right' and 'over-shoulder-left' cameras, but only accesses RGB information.

Note that in section 7.5.4, we discuss two additional tasks that we tried but actually partially failed to train.

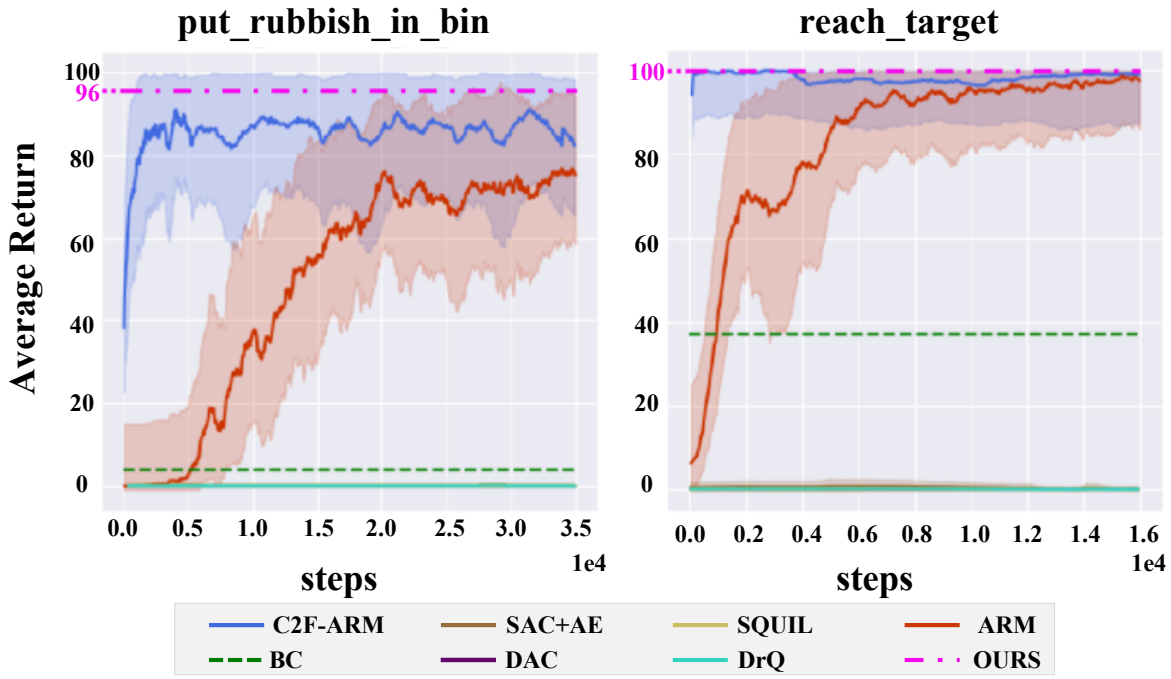


Figure 7.2 – Our training results compared with [80] learning curves on monocular setting.

Fig. 7.2 shows the results for the first set of experiments. It highlights that our method (1) outperforms traditional BC by a large margin and (2) slightly outperforms mean results of the best state-of-the-art RL based approach, C2F-ARM, on tasks where the manipulated objects can be seen from all view points on the initial observation of the scene. Other RL approaches were already outperformed by C2F-ARM, and by extension we outperform them as well.

Fig. 7.3 shows the results for the second set of experiments. It highlights that our method can still compete with C2F-ARM when they also access multiple views.

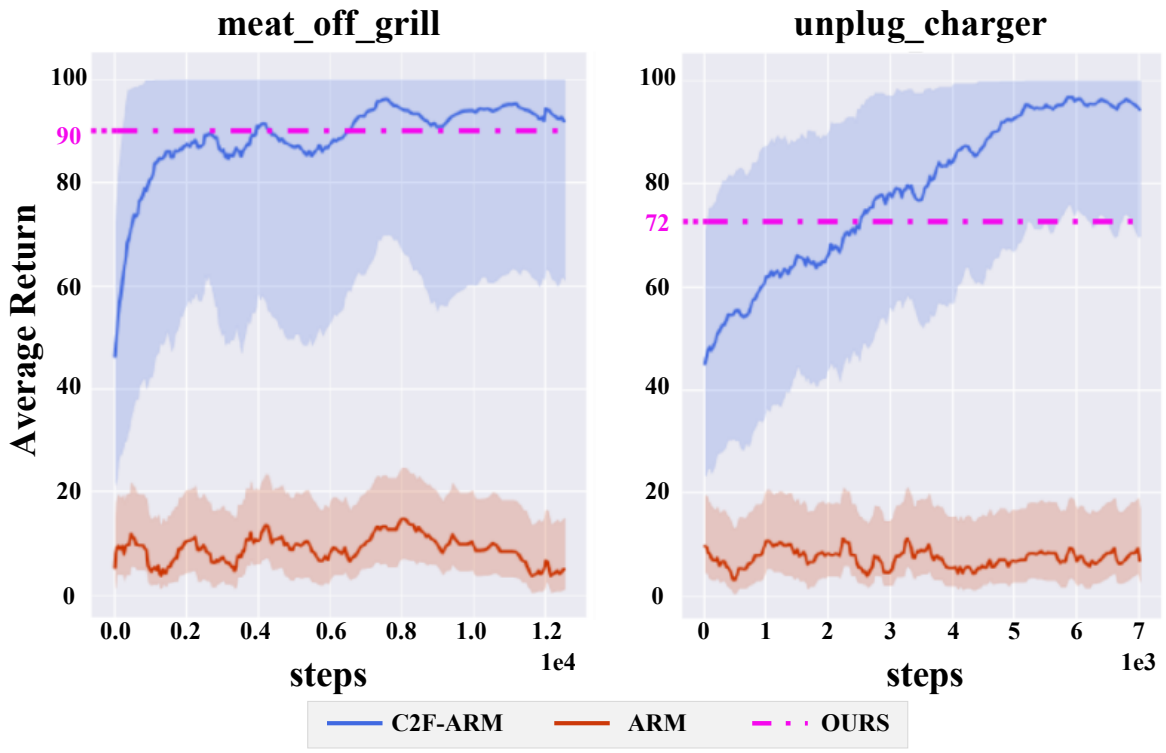


Figure 7.3 – Our training results compared with [80] learning curves on multi-view setting.

7.5.3 Does the one-shot waypoints prediction help reduce error accumulation during task execution?

Section 7.5.2 reports evaluation results of various baselines, but the differences among them can be attributed to several factors. Firstly, C2F-ARM, our model, and other existing baselines employ different networks for observation representation. Secondly, while our model predicts waypoints, other approaches predict the gripper pose of the next step. Finally, our model predicts the entire trajectory configuration at once, whereas other approaches rely on sequential decision-making, which can result in prediction errors and observation distribution shifts.

To isolate the impact of error accumulation and associated observation distribution shift on the same tasks studied in section 7.5.2, we trained a behavior cloning model to predict the next waypoint sequentially from low-dim observations. This model has access to the 6D poses of each object in the scene, enabling it to infer ground-truth object positions directly without relying on a vision model to process images of the scene. While the model predicts waypoints like our approach, it can still be affected by prediction errors and observation distribution shifts

due to its sequential predictions.

The results of the waypoint-based behavior cloning trained from Ground Truth 6D objects poses and our framework are compared in Table 7.2. This table displays both the training loss and success rate outcomes. To gain a better understanding of how the training results impact the success rate on model deployment, we separated the training loss into three losses: The "Train Pose" loss, which concentrates on the first three dimensions of the 8D waypoints, corresponding to the position of the robot TCP. The "Train Quat" loss focuses on the orientation of the TCP, while the "Train Grip" loss is concerned with the gripper state.

Compared to our approach, the behavior cloning model trained from object poses has significantly lower "Train Pose" and "Train Quat" losses. Without any observation distribution shift between training and inference, or prediction error accumulation, the behavior cloning model should be much more accurate than our framework, and, by extension, should achieve a better success rate. However, our framework can produce similar or superior success rates. In shorter, despite the behavior cloning approach shows better prediction performances during training, the final success rate is affected by error accumulation during task execution, which is not the case with our framework. The exception is the 'pick_and_lift' task, which our model failed to grasp. For the 'put_rubbish_in_bin' task, which completely fails, we noticed that the behavior cloning model was trapped in the neighborhood of the first waypoint, while never actually reaching this waypoint. This was an out-of-distribution case for the trained model

7.5.4 Failure cases

This section highlights the current limitations of our framework. Fig.7.4 points out mitigated results on take_money_out task, while Fig.7.5 highlights that typical failure case for our framework is caused by critical occlusions on the initial observation of the scene.

Table 7.2 – Evaluation of the distribution shift between training and inference.

	Trash.	Grill.	Unplug	PickLift	MoneyOut
Behavior Cloning From 3D Ground Truth					
Train Pose	$3e10^{-5}$	$3e10^{-4}$	$3e10^{-5}$	$3.5e10^{-4}$	$1e10^{-5}$
Train Quat.	$3e10^{-4}$	$4.6e10^{-3}$	$5e10^{-5}$	$5.2e10^{-2}$	$4e10^{-5}$
Train Grip.	$6.3e10^{-3}$	$1.4e10^{-3}$	$2.8e10^{-3}$	$5e10^{-6}$	$1.5e10^{-2}$
Success rate	0	100	70	0.8	0.8
Ours					
Train Pose	$1.6e10^{-2}$	$9.5e10^{-3}$	$8.6e10^{-3}$	$5.7e10^{-2}$	$1.3e10^{-2}$
Train Quat.	$2.3e10^{-2}$	$2.2e10^{-2}$	$2.4e10^{-2}$	0.53	$4.3e10^{-2}$
Train Grip.	$1.7e10^{-3}$	$2.7e10^{-3}$	$1.3e10^{-3}$	$2.2e10^{-3}$	$3.9e10^{-3}$
Success Rate	96	90	72	–	46

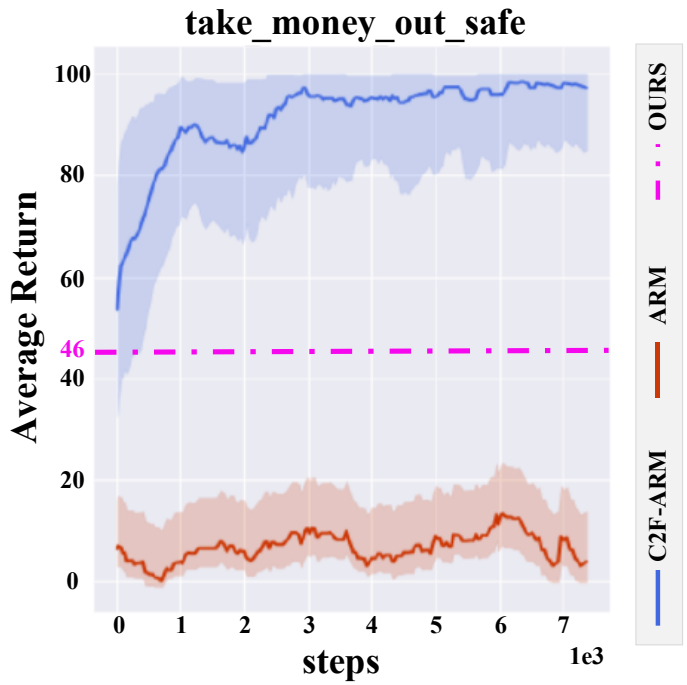


Figure 7.4 – Moderate results compared with [80] learning curves.

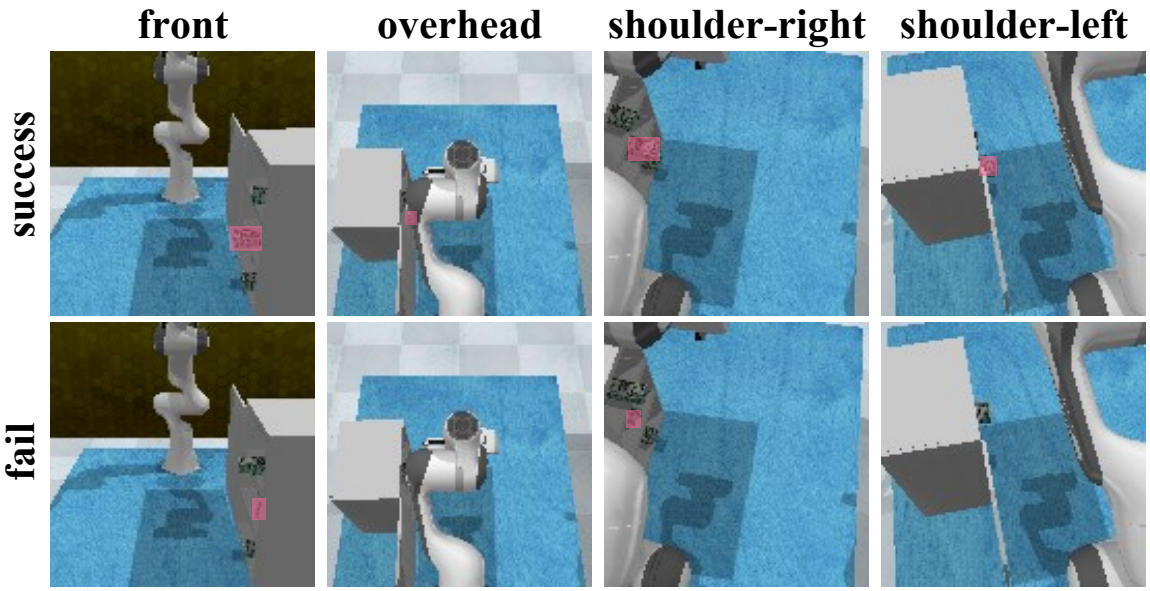


Figure 7.5 – Visualization of initial observation of the scene in failure vs in success cases on take_money_out task.

CHAPTER 8

LEARNING A POLICY AGNOSTIC TO EMBODIMENT, VIEWPOINT, AND BACKGROUND ENVIRONMENT

Overview

Contents

		Train Pos. Err. (mm)	Unseen Ex Pos. Err. (mm)	Unseen View Pos. Err. (mm)	Unseen Ex&View Pos. Err. (mm)	Train View Suc. Rate (%)	Unseen View Suc. Rate
train view is clean (front view), unseen view contains self-occlusions with the robot arm (overhead view)							
Keypoint	VIP	15.3	62.7	300.6	289.0	42	0
	4D Pt.Cl.	2.1	22.0	171.3	197.4	94	10
Frame	VIP	5.8	10.1	80.0	80.0	2	0
	4D Pt.Cl. (h=16, a=8)	7.6	16.3	136.0	150.0	100	6
train view contains self-occlusions with the robot arm (overhead view), unseen view is clean (front view)							
Keypoint	VIP	6.6	61.5	250.0	231.4	32	0
	4D Pt.Cl.	11.0	30.7	46.1	51.6	86	2
Frame	VIP	2.5	6.2	128	129	6	0
	4D Pt.Cl. (h=16, a=8)	4.6	14.8	25.9	31.2	96	14

Table 8.1 – Simulation, world frame

8.1 Introduction

8.2 Related work

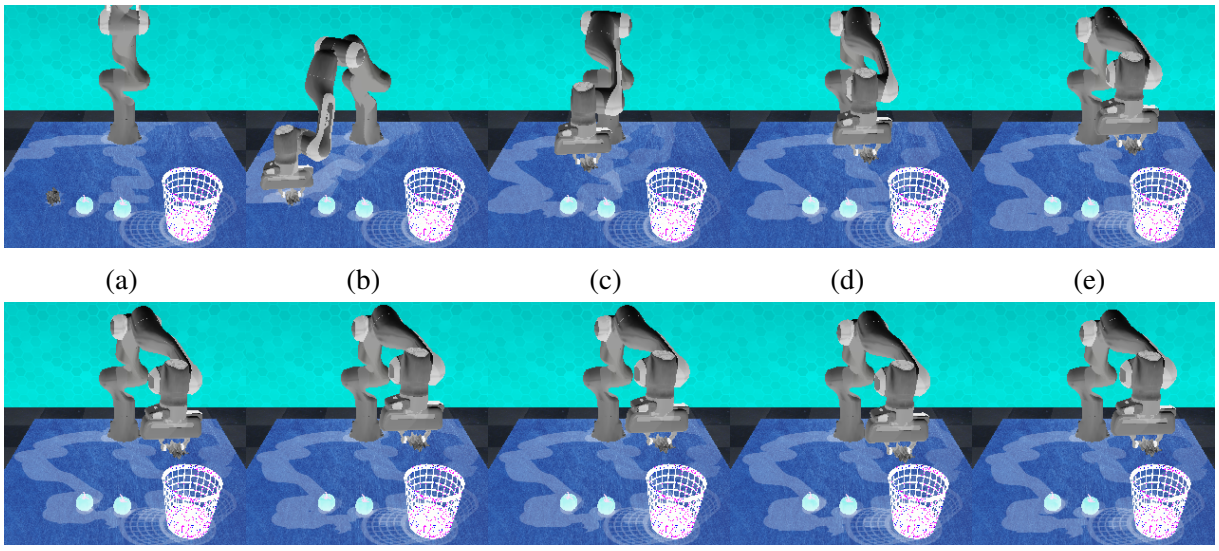
8.3 Datasets

8.3.1 Simulated data

8.3.2 Real data

8.4 Evaluating the viewpoint invariance of various visual representations on simulated data

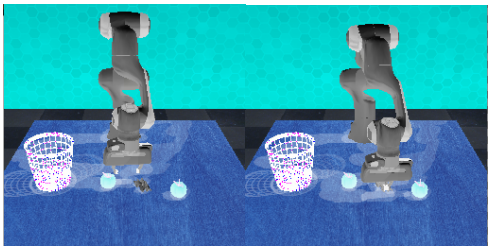
8.5 From real human demos to robot simulated deployment



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

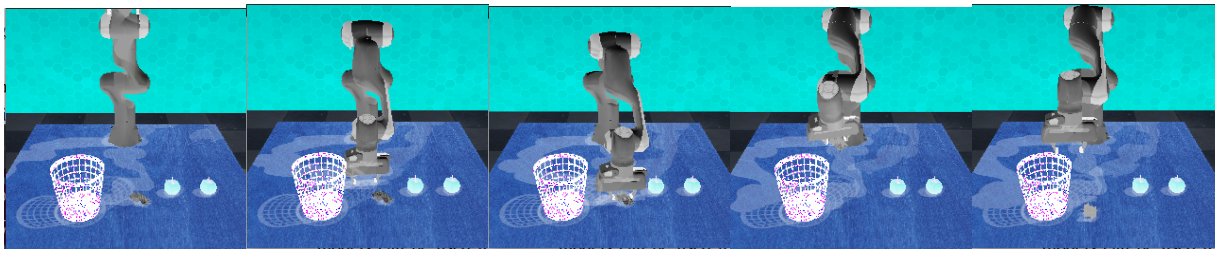
(f) (g) (h) (i) (j)

Figure 8.1



(a) (b)

Figure 8.2



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

Figure 8.3

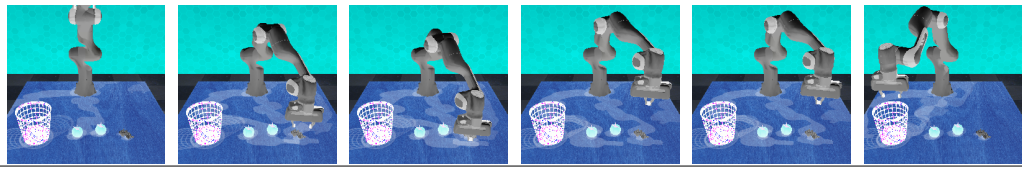
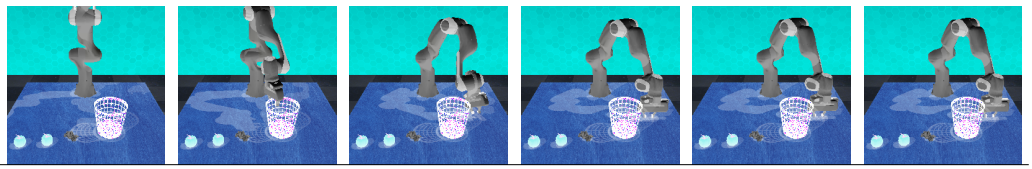
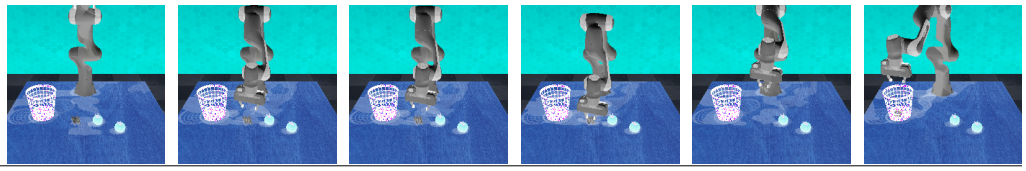
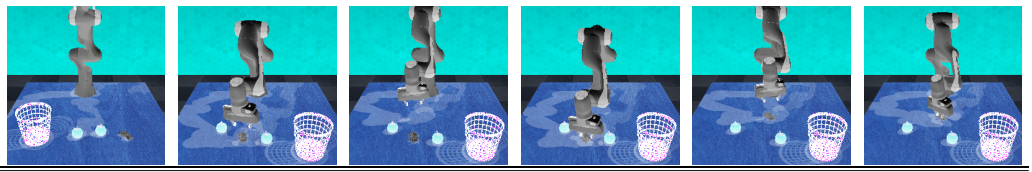
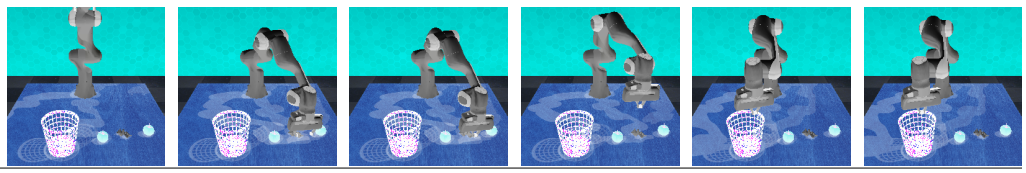
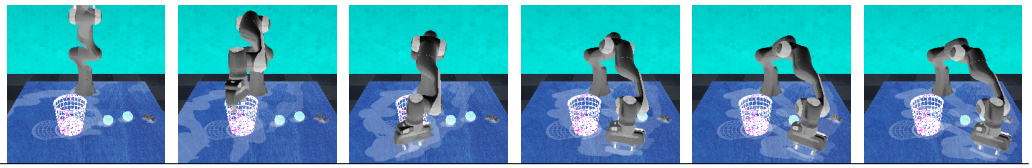
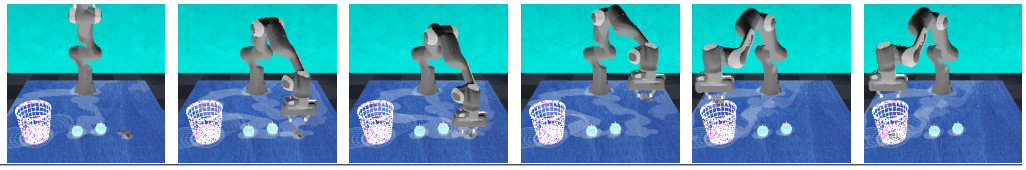
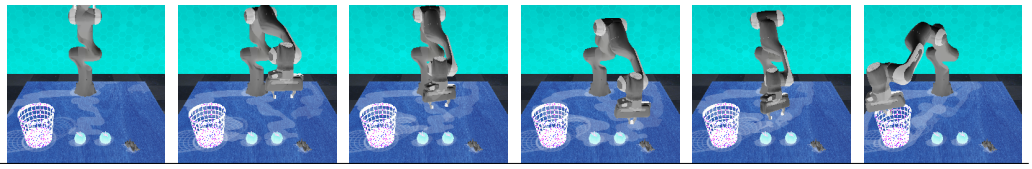
		Keypoint
VIP	Train View	
	Novel View	
4D Pt.Cl.	Train View	
	Novel View	
		Frame
VIP	Train View	
	Novel View	
4D Pt.Cl.	Train View	
	Novel View	

Table 8.2 – Train front, deploy overhead. From left to right, tme is increaing.

		Train Pos. Err. (mm)	Unseen Ex Pos. Err. (mm)	Unseen View Pos. Err. (mm)	Unseen Ex&View Pos. Err. (mm)	Train View Suc. Rate (%)	Unseen View Suc. Rate
train view is clean (front view), unseen view contains self-occlusions with the robot arm (overhead view)							
Keypoint	VIP						
	4D Pt.Cl.	-	-	-	-	-	-
Frame	VIP						
	4D Pt.Cl.						
train view contains self-occlusions with the robot arm (overhead view), unseen view is clean (front view)							
Keypoint	VIP						
	4D Pt.Cl.					-	
Frame	VIP						
	4D Pt.Cl.						

Table 8.3 – Simulation, tool frame

		Train Pos. Err. (mm)	Hum. Unseen Ex Pos. Err. (mm)	Simulated Ex Pos. Err. (mm)	Simulation Suc. Rate
Keypoint	VIP-original				
	VIP-no-Hum.				
	VIP-original+no-Hum.				
	4D Pt.Cl.				
Frame	VIP				
	VIP-no-Hum.				
	VIP-original+no-Hum.				
	4D Pt.Cl.				

Table 8.4 – Training on human demonstrations, moving camera; deploying on simulation

		Train Pos. Err. (mm)	Train Obst. Av. Pos. Err. (mm)	Eval Pos. Err. (mm)	Eval Obst. Av Pos. Err. (mm)	Suc. Rate Front	Suc. Rate Overhead
Basic Keypoints	VIP-original						
	VIP-no-Hum.						
	VIP-original+no-Hum.						
	4D Pt.Cl.	18.9	19.7	219.3	196.1	0	0
Two-Step Keypoints	VIP-original						
	VIP-no-Hum.						
	VIP-original+no-Hum.						
	4D Pt.Cl.	13.8	18.0	128.3	171.0	14	0
Frame	VIP						
	VIP-no-Hum.						
	VIP-original+no-Hum.						
	4D Pt.Cl.						

Table 8.5 – Training on human demonstrations, moving camera; deploying on simulation

		Train Pos. Err. (mm)	Hum. Unseen Ex Pos. Err. (mm)	Simulated Ex Pos. Err. (mm)	Simulation Suc. Rate
Keypoint	VIP-original				
	VIP-no-Hum.				
	VIP-original+no-Hum.				
	4D Pt.Cl.				
Frame	VIP				
	VIP-no-Hum.				
	VIP-original+no-Hum.				
	4D Pt.Cl.				

Table 8.6 – Training on human demonstrations, fixed camera; deploying on simulation

		Train Pos. Err. (mm)	Hum. Unseen Ex Pos. Err. (mm)	Simulated Ex Pos. Err. (mm)	Simulation Suc. Rate
Keypoint	VIP-original				
	VIP-no-Hum.				
	VIP-original+no-Hum.				
	4D Pt.Cl.				
Frame	VIP				
	VIP-no-Hum.				
	VIP-original+no-Hum.				
	4D Pt.Cl.				

Table 8.7 – Training on human demonstrations, ego camera; deploying on simulation

LIST OF ABBREVIATIONS

The list of the abbreviations used in this thesis in alphabetical order.

— BC: Behavior Cloning

— HPE: Human Pose Estimation

ANNEXES

A Human-to-Robot and Sim-to-Real Policy transfer

A.1 Human-to-Robot transfer

Regarding Human-to-Robot transfer, most approaches leverage a form of CycleGan [207]. The CycleGan approach is based on the cycle consistency, which states that if we translate an image from domain X to domain Y, and then back from domain Y to domain X, the resulting image should be similar to the original image. In the Human-to-Robot transfer case, the domain X corresponds to the human demonstrations, and the domain Y corresponds to the robot deployment environment. The CycleGan architectures are then usually made of two generators, G_Y and G_X, and two discriminators D_Y and D_X. The goal of the first generator G_Y is to translates images from domain X to domain Y. The first discriminator, D_Y, is trained to discriminate real images from domain Y from generated images $\hat{Y} = G_Y(X)$, while the first generator is trained to fool the first discriminator, which is a typical GAN setup [52].

An second generator, G_X, is trained to translate back the images from domain Y to domain X, and a second discriminator is trained to discriminate real images from domain X from the images translated back from domain Y $\hat{X} = G_X(\hat{Y})$, which is a second GAN setup. An additionnal L_2 loss is usually added between original images from X and corresponding $\hat{X}$ generated images. The whole pipeline ensures that the generator G_Y learns to generate images that look like the images from domain Y, while the L_2 loss aims to ensure that the scene configuration is preserved during the translation.

Typical approaches leveraging a CycleGan on images for Human-to-Robot transfer are [134, 190].

The same cycle consistency principle can be applied in the latent space of a control policy already trained on out-of-domain data, instead of applying it on the image space [179].

However, all these approaches require to have access to the in-domain data for training, which means that the robot has to be teleoperated to collect the data, which is what we are trying to avoid by training the control policy from human demonstrations. This motivates the need to develop visual representations that are invariant to the embodiment and to the environment, and

that can be used to train the control policy from out-of-domain data.

A.2 Sim-to-Real transfer

B Object detection and segmentation benchmarks and metrics

B.1 Benchmarks

The PASCAL Visual Object Classes (VOC) Challenges [36, 37] provided leading object detection and segmentation benchmarks from 2005 to 2012. The latest version of PASCAL VOC datasets, VOC2012, consists of 20 object categories representing everyday objects, such as "aeroplane", "bicycle" or "person". It provided 11k images along with 31k annotated objects for object detection, and 3k images along with 7k annotated objects for object segmentation.

The ILSVRC dataset [147] arised from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) that ran from 2010 to 2017. The ILSVRC dataset consists of 500k images along with 500k annotated objects for 200 object categories for object detection, and does not provide object segmentation annotations.

The MS-COCO dataset [105] is a more challenging dataset than PASCAL VOC and ILSVRC, as it consists of almost a milion annotated objects for 164k images, and 80 object categories. All annotations are provided for object detection and object segmentation. It therefore provides smaller and more crowded scenes, which makes the object detection task more challenging. Additionnaly, later versions of the dataset provided annotations for human keypoints, which are the locations of human joints in the images. This benchmark appeared in 2014 and is still widely used as a benchmark for object detection models.

The Open Images Detection challenges [91] started in 2018, and provided a dataset of 9M images annotated with image-level labels, object bounding boxes, object segmentation masks, visual relationships, and localized narratives. The are 1.9M images annotated with 16M bounding boxes on 600 object categories. In terms of segmentation, it contains 2.8M object instances in 350 classes.

LVIS [59] is a dataset that was released in 2019, and that provides annotations for instance segmentation, and provides about 2 million high-quality instance segmentation masks for over 1000 entry-level object categories in 164k images.

LVIS and MS-COCO have recently been repurposed to provide benchmarks for the Open-

Set Object Detection problem by training the models on a subset of the classes, and testing them on novel classes, unseen during training [56, 3]. More recently, SA-1B [87] was released as the training dataset of segment-anything models [87], which will be discussed in the next section. It features 1B masks and 11M high resolution images (the shortest image side is set to 1500 pixel, where COCO images resolutioon was about 480×640 pixels).

B.2 Metrics

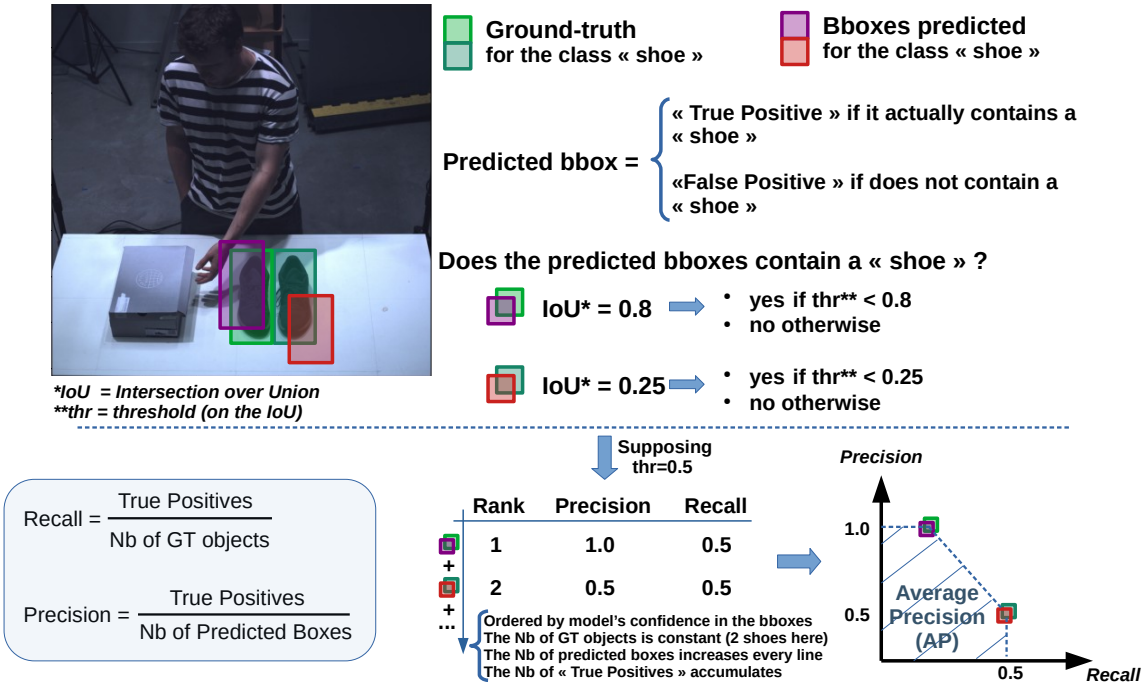


Figure B.4 – Explaination of the mAP metric, with an example featuring 2 objects to detect, and 2 predictions.

The mean Average Precision (mAP) is the most widely used metric to evaluate the performances of object detection models. It was introduced in the VOC2007 challenge [36].

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{8.1}$$

The Fig. B.4 will be used as a support to explain the mAP metric. Before defining the mAP metric, we first need to define Precision, Recall and Intersection over Union (IoU) metrics, which are used to compute the mAP. The definition of the Precision and Recall metrics depends on the notions of True Positives, False Positives, True Negatives and False Negatives. In the

context of object detection, the True Positives (TP) correpond to objects that are in the image and that are correctly detected by the model (the right object category at the right location). The False Positives (FP) correspond to bounding boxes that are predicted by the model at a location where there's either no object of interest or an object with a different category than the one predicted by the model. False Negatives (FN) correspond to objects that are in the image but that are not detected by the model. The True Negatives (TN) are more abstract in the context of object detection. They represent the correctly predicted in-existence of objects in the background regions of the images. In other words, the TN are the regions of the image that are not predicted as objects by the model, and that are not objects in the ground truth.

The Precision is defined as the ratio of the True Positives to the sum of the True Positives and the False Positives. In other words, the Precision is the ratio of the correctly detected objects to the total number of objects detected by the model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{8.2}$$

The Recall is defined as the ratio of the True Positives to the sum of the True Positives and the False Negatives. In other words, the Recall is the ratio of the correctly detected objects to the total number of objects in the ground truth.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{8.3}$$

The question that arises is how to determine if a predicted bounding box is a True Positive or a False Positive. This is where the Intersection over Union (IoU) metric comes into play. The IoU is defined as the ratio of the intersection of the predicted bounding box and the ground truth bounding box to the union of the predicted bounding box and the ground truth bounding box. In other words, the IoU is the ratio of the overlapping area between the predicted bounding box and the ground truth bounding box to the total area of the two bounding boxes.

To detemine if a predicted bounding box is a True Positive or a False Positive, the closest ground truth bounding box corresponding to the object category of the predicted bounding box is selected. If the IoU between the predicted bounding box and the selected ground truth bounding box is above a certain threshold, the predicted bounding box is considered as a True Positive. Otherwise, the predicted bounding box is considered as a False Positive.

To compute the Average Precision curve (AP) of a model, the bounding boxes are sorted by decreasing confidence scores (the probability that the bounding box contains an object of interest), which is also predicted by the model for each bounding box. The Precision and Recall

are computed iteratively for each bounding box, starting from the highest confidence score. At each iteration, the Precision and Recall scores are computed by considering all the bounding boxes that have a confidence score higher than the confidence score of the current bounding box, which is the set of all the bouning boxes seen in the previous iterations. We can notice that the more we add bounding boxes to the set, the more chance we have to detect the objects in the image, which increases the Recall. However, the more bounding boxes we add to the set, the more chance we have to predict False Positives, which decreases the Precision. The goal is to find the right trade-off between the Precision and the Recall. Once the Precision and Recall scores are computed for each bounding box, they are plotted on a graph, where the Recall is on the x-axis and the Precision is on the y-axis. The AP is then computed as the area under the Precision-Recall curve.

The whole AP computation process is illustrated in the Fig. B.4 with a simple example, where the goal of the model is to detect "shoes" in the image. The example features 2 ground-truth (GT) objects to detect, and 2 predicted bounding boxes. In this example, we suppose that the IoU between the purple predicted bounding box and light-green GT bounding box is 0.8, and the IoU between the red predicted bounding box and dark-green GT bounding box is 0.25. The Top part of the figure highlights how this assumption impacts the computation of True vs False Positives, with regard to the IoU threshold. The bottom part of the figure highlights how the Precision and Recall are computed iteratively for each bounding box, and how the AP is computed as the area under the Precision-Recall curve.

We can notice that the AP metric is very sensitive the chosen IoU threshold. While the AP metric introduced in the VOC challenges used an IoU threshold of 0.5, the COCO benchmark introduced an improved version of the AP metric, which computes the AP for different IoU thresholds between 0.5 and 0.95, and averages them. Furthermore, the COCO AP is computed using a different number og Precision-Recall points than the VOC AP. Both metrics are averaged over all object categories to provide the final mean average precision (mAP) of the model. After the introduction of the COCO benchmark, the AP is often provided both for fixed IoU thresholds (e.g., 0.5, 0.75) and for the averaged AP over different IoU thresholds (e.g., 0.5:0.95). In the following sections, we will refer to the Pascal VOC AP metric as the VOC-mAP@0.5, and to the COCO AP metric as the COCO-mAP@[0.5:0.95] when the AP is averaged over different IoU thresholds and as COCO-mAP@p when the AP is computed for a fixed IoU threshold p, e.g., COCO-mAP@0.5.

C Depth estimation models and sensors

C.1 Depth estimation models

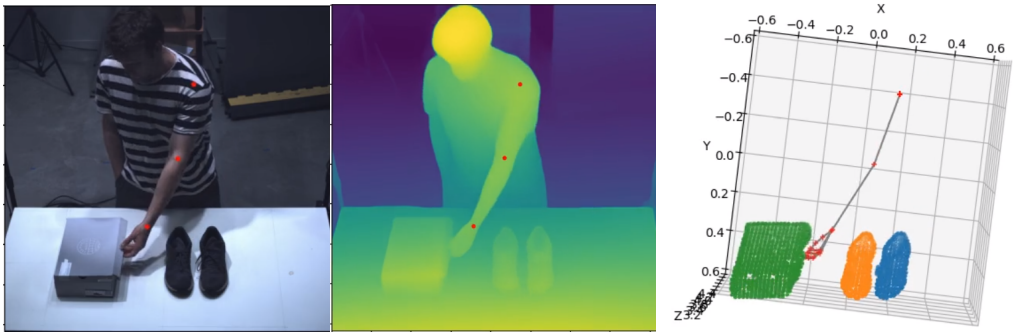
The monocular depth estimation task consists in predicting the depth map of a scene from a single RGB image. It allows to reduce the cost of 3D data collection, as it does not require the use of depth sensors, such as LiDAR or stereo cameras. Moreover, it can allow to estimates depth map of scenes where some depth sensors face limitations, such as transparent objects, or scenes with reflective surfaces, if the training set of the model contains such scenes. With the cumulative increase in the amount of available datasets for monocular depth estimation, including DIML [25], HRWSI [186], IRS [178], MegaDepth [100], TartanAir [180], KITTI [47], NYUv2 [154], milions of labeled images are now available for training monocular depth estimation models. This has led to the development of models that are trained on large-scale datasets, and that are able to generalize to a wide range of scenes, including indoor, outdoor, dynamic scenes, simulated/real, transparent objects, aerial, underwater, etc.

Several approaches have proposed to use diffusion models to generate depth maps from RGB images, including Marigold [86] and geowizard [42]. On the other hand, the Midas [133] studies investigates the use of pre-trained vision models, such as Vit [34], Beit [4] or Swin Transformers [110] finetuned for depth estimation. More recently, the Depth-Anything models [196, 195] follow the same approach by finetuning a pretrained Dinov2 [122] on mixtures of synthetic and real depth estimation datasets.

Since Depth-Anything models show better generalization performances than the other approaches, we investiguated the feasibility of using such models to estimate the depth of the objects in our manipulation tasks for imitation learning purpose.

The first limitation of these approaches is that they generate relative depth maps, which means that the depth values are normalized between 0 and 1. In other words, the actual metric depth values are not provided by the model, which makes it difficult to use the depth maps for robotic applications. The authors of the Depth-Anything did actually finetune their model for metric depth estimation, but on specific datasets, such as the NYUv2 or KITTI datasets, which are not representative of the scenes we are interested in. Therefore, we need to denormalize the depth maps to obtain metric depth values.

We collected a set of 30 demonstrations of a "shoe packaging task" on a multi-view setup that allowed us to recover accurate 3D human pose estimation using the triangulation approach described in the Chapter 4. Concretely, the 3D human pose estimation allows to recover the 3D coordinates of the human joints in the cameras coordinate systems. Using 2D human pose

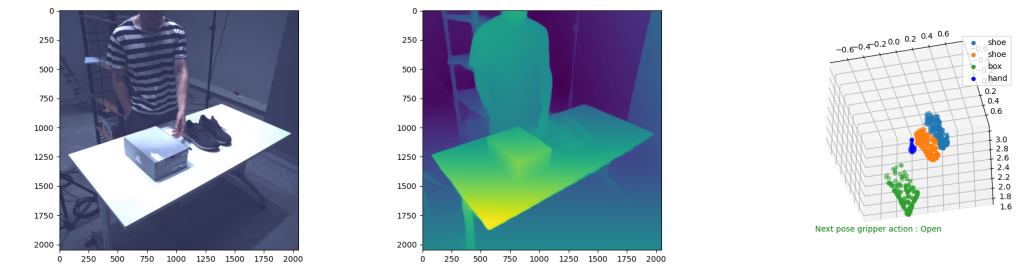


(a) RGB (b) Depth Map (c) Point Cloud

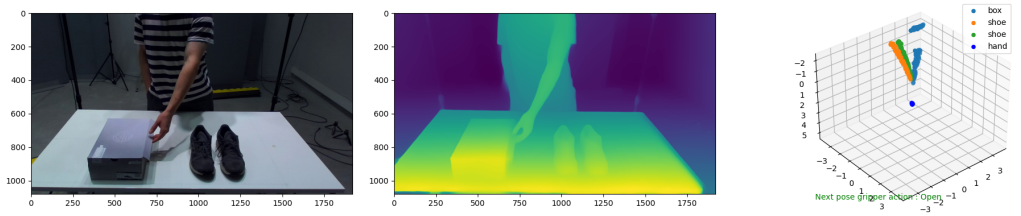
Figure C.5 – Denormalization success.

estimation on the images, we can recover the position of the same joints in the images, and by extension, in the relative depth maps generated by the Depth-Anything model. The z-coordinate of the 3D human joints obtained by triangulation correspond to the metric depth values of the joints in the cameras coordinate systems. Therefore, we can solve a simple system of equations to obtain the linear transformation from the relative depth values to the metric depth values. The tranformation has to be found for each images, since the relative depth values are not consistent between the images, as they depend on the maximum and minimum depth values of the scene.

After the transformation matrix from the relative depth values to the metric depth values is found, we can apply it to the whole depth maps before projecting them into point clouds using the method descibed in section 2.4 of the Chapter 2. The Fig. C.5 shows an example of successful denormalization of the depth map generated by the Depth-Anything model. However, most of the demonstrations will contain failed denormalizations, as shown in the Fig. C.6. Several problems can lead to failed denormalizations. The first problem is illustrated by the first row of the Fig. C.6 (a, b and c). We can notice that the edge of the box (as well as the table) are way too close to the camera in terms of depth values (the lighter the values, the closer to the camera). These edges are not consistent with the rest of the object they belong to> This is a limitation of the Depth-Anything model, which is not able to consistently generate accurate depth values for our use case. The second row of the Fig. C.6 (d, e and f) highlights a more severe problem, still due to the Depth-Anything model depth inconsistencies. We can see that the elbow is further away from the camera than the wrist, which is not the case in the 3D scene considered. The problem is that those 2 joints are used to recover the denormalization matrix. Since the relative depth values are wrong for these 2 joints, the denormalization matrix is wrong, and the whole depth map is denormalized incorrectly, resulting in a completely incorrect point cloud. While



(a) RGB (b) Depth Map (c) Point Cloud



(d) RGB (e) Depth Map (f) Point Cloud

Figure C.6 – Denormalization fails.

the depth estimation of the wrist and ankle failed in this case because of depth-anything model limitations, note that the depth estimation of crutial points can also happen if those joints are occluded in the image, or if the 2D human pose estimation is incorrect. All in all, the denormalization process is not reliable enough to be used for imitation learning purposes yet, which indicates that using depth sensors during the data collection phase is still necessary.

C.2 Depth sensors

The release of the Kinect sensor in 2010 has revolutionized the field of 3D vision, by providing a cheap and easier-to-use depth sensor. It was followed byy the realsense series, zed and zed 2 cameras, as well as the kinect 2 released in 2013. Pinto et al. showed that the Kinect 2, equipped with a ToF sensor, provides a distance error of less than 5mm for distances less than 3m [127]. Tadic et al. [163] compared the realsense (stereo + Infrared projector and camera), zed and zed 2i cameras (solely stereo), and showed that

Besides, Zhao et al. [205] argued that most existing RGB-D datasets are collected using Realsense or Kinect depth cameras, which require dedicated computing ressources and are not easily portable. Therefore, it severely restricts the scene diversity, and causes a large domain gap between the collected dataset and real-world applications. To address this issue, they developped

a new dataset, called ARKitTrack, which is collected using an iPhone 12 Pro, which is more portable and easier to use than above mentionned depth sensors. The iphone 12 Pro is equipped with a LiDAR sensor, which provides depth information, as well as RGB cameras for the color information. The iphone is an ideal device to collect large scale RGB-D data, since it is widely used by the general public, and easily portable. Moreover, it embeds imu sensors, which can be used to collect data from extensively various viewpoints, and to provide the ground truth poses of the camera, which allows to define a world coordinate system common to all the frames of a given video. For those reasons, we chose to collect our dataset using iPhones 13 Pro, in order to investiguate the feasability and limitations of such devices to collect RGB-D data for robotic applications.

BIBLIOGRAPHY

[1] Kai Arulkumaran et al., « Deep reinforcement learning: A brief survey », *in*: *IEEE Signal Processing Magazine* 34.*6* (2017), pp. 26–38 (cit. on p. 46).

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, « Segnet: A deep convolutional encoder-decoder architecture for image segmentation », *in*: *IEEE transactions on pattern analysis and machine intelligence* 39.*12* (2017), pp. 2481–2495 (cit. on p. 22).

[3] Ankan Bansal et al., « Zero-shot object detection », *in*: *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 384–400 (cit. on p. 141).

[4] Hangbo Bao et al., « Beit: Bert pre-training of image transformers », *in*: *arXiv preprint arXiv:2106.08254* (2021) (cit. on p. 144).

[5] Adrien Bardes et al., « Revisiting feature prediction for learning visual representations from video », *in*: *arXiv preprint arXiv:2404.08471* (2024) (cit. on p. 7).

[6] Vasileios Belagiannis et al., « 3D Pictorial Structures Revisited: Multiple Human Pose Estimation », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.*10* (2016), pp. 1929–1942, DOI: 10.1109/TPAMI.2015.2509986 (cit. on p. 66).

[7] Alex Bewley et al., « Simple online and realtime tracking », *in*: *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468, DOI: 10.1109/ICIP.2016.7533003 (cit. on p. 27).

[8] Raunaq Bhirangi et al., « ReSkin: versatile, replaceable, lasting tactile skins », *in*: *arXiv preprint arXiv:2111.00071* (2021) (cit. on pp. 39, 61).

[9] Antonio Bicchi, J Kenneth Salisbury, and David L Brock, « Contact sensing from force measurements », *in*: *The International Journal of Robotics Research* 12.*3* (1993), pp. 249–262 (cit. on p. 35).

[10] Biswajit Bose, Xiaogang Wang, and Eric Grimson, « Multi-class object tracking algorithm that handles fragmentation and grouping », *in*: *2007 IEEE conference on computer vision and pattern recognition*, IEEE, 2007, pp. 1–8 (cit. on p. 27).

[11] Simon Bultmann and Sven Behnke, « Real-time multi-view 3D human pose estimation using semantic feedback to smart edge sensors », *in*: *Proceedings of the Robotics: Science and Systems (RSS)*, 2021 (cit. on pp. 67, 74).

[12] Zhongang Cai et al., « Humman: Multi-modal 4d human dataset for versatile sensing and modeling », *in*: *European Conference on Computer Vision*, Springer, 2022, pp. 557–577 (cit. on p. 74).

[13] Nicolas Carion et al., « End-to-end object detection with transformers », *in*: *European conference on computer vision*, Springer, 2020, pp. 213–229 (cit. on p. 19).

[14] Mathilde Caron et al., « Emerging properties in self-supervised vision transformers », *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660 (cit. on pp. 7, 10).

[15] Yu-Wei Chao et al., « DexYCB: A benchmark for capturing hand grasping of objects », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9044–9053 (cit. on p. 72).

[16] Chaofan Chen et al., « HAPGN: Hierarchical Attentive Pooling Graph Network for Point Cloud Segmentation », *in*: *IEEE Transactions on Multimedia* 23 (2021), pp. 2335–2346, DOI: 10.1109/TMM.2020.3009499 (cit. on p. 38).

[17] Siheng Chen et al., « 3D Point Cloud Processing and Learning for Autonomous Driving: Impacting Map Creation, Localization, and Perception », *in*: *IEEE Signal Processing Magazine* 38.*1* (2021), pp. 68–86, DOI: 10.1109/MSP.2020.2984780 (cit. on p. 35).

[18] Xi Chen et al., « Focalclick: Towards practical interactive image segmentation », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1300–1309 (cit. on p. 23).

[19] Xiaoyu Chen et al., « Understanding domain randomization for sim-to-real transfer », *in*: *arXiv preprint arXiv:2110.03239* (2021) (cit. on p. 42).

[20] Ho Kei Cheng and Alexander G Schwing, « Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model », *in*: *European Conference on Computer Vision*, Springer, 2022, pp. 640–658 (cit. on p. 27).

[21] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang, « Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5559–5568 (cit. on p. 23).

[22] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang, « Rethinking space-time networks with improved memory coverage for efficient video object segmentation », *in*: *Advances in Neural Information Processing Systems* 34 (2021), pp. 11781–11794 (cit. on p. 27).

[23] Ho Kei Cheng et al., « Putting the Object Back into Video Object Segmentation », *in*: *arXiv*, 2023 (cit. on p. 27).

[24] Ho Kei Cheng et al., « Tracking anything with decoupled video segmentation », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 1316–1326 (cit. on p. 27).

[25] Jaehoon Cho et al., « Diml/cvl rgb-d dataset: 2m rgb-d images of natural indoor and outdoor scenes », *in*: *arXiv preprint arXiv:2110.11590* (2021) (cit. on p. 144).

[26] Keh-Shih Chuang et al., « Fuzzy c-means clustering with spatial information for image segmentation », *in*: *computerized medical imaging and graphics* 30.*1* (2006), pp. 9–15 (cit. on p. 21).

[27] Angela Dai et al., « ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes », *in*: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017 (cit. on p. 38).

[28] Murtaza Dalal, Deepak Pathak, and Ruslan Salakhutdinov, « Accelerating Robotic Reinforcement Learning via Parameterized Action Primitives », *in*: *Advances in Neural Information Processing Systems*, ed. by A. Beygelzimer et al., 2021, URL: https://openreview.net/forum?id=48uzkHOKMfz (cit. on p. 117).

[29] Navneet Dalal and Bill Triggs, « Histograms of oriented gradients for human detection », *in*: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, Ieee, 2005, pp. 886–893 (cit. on p. 17).

[30] David J Daniels, « A review of GPR for landmine detection », *in*: *Sensing and imaging* 7.*3* (2006), p. 90 (cit. on p. 35).

[31] Jia Deng et al., « Imagenet: A large-scale hierarchical image database », *in*: *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255 (cit. on pp. 8, 17).

[32] Coline Devin et al., « Learning modular neural network policies for multi-task and multi-robot transfer », *in*: *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 2169–2176 (cit. on p. 118).

[33] Jacob Devlin et al., « Bert: Pre-training of deep bidirectional transformers for language understanding », *in*: *arXiv preprint arXiv:1810.04805* (2018) (cit. on p. 7).

[34] Alexey Dosovitskiy et al., « An image is worth 16x16 words: Transformers for image recognition at scale », *in*: *arXiv preprint arXiv:2010.11929* (2020) (cit. on pp. 22, 144).

[35] Frederik Ebert et al., « Visual foresight: Model-based deep reinforcement learning for vision-based robotic control », *in*: *arXiv preprint arXiv:1812.00568* (2018) (cit. on p. 117).

[36] Mark Everingham et al., « The pascal visual object classes (voc) challenge », *in*: *International journal of computer vision* 88 (2010), pp. 303–338 (cit. on pp. 16, 140, 141).

[37] Mark Everingham et al., « The pascal visual object classes challenge: A retrospective », *in*: *International journal of computer vision* 111 (2015), pp. 98–136 (cit. on p. 140).

[38] Cristiana de Farias et al., « Simultaneous tactile exploration and grasp refinement for unknown objects », *in*: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3349–3356 (cit. on p. 39).

[39] Pedro Felzenszwalb, David McAllester, and Deva Ramanan, « A discriminatively trained, multiscale, deformable part model », *in*: *2008 IEEE conference on computer vision and pattern recognition*, Ieee, 2008, pp. 1–8 (cit. on p. 17).

[40] Pedro F Felzenszwalb and Daniel P Huttenlocher, « Efficient graph-based image segmentation », *in*: *International journal of computer vision* 59 (2004), pp. 167–181 (cit. on p. 21).

[41] Sascha Fleer et al., « Learning efficient haptic shape exploration with a rigid tactile sensor array », *in*: *PLoS ONE* 15.1 (2020), pp. 1–22, ISSN: 19326203, DOI: 10.1371/journal.pone.0226880, arXiv: 1902.07501, URL: http://dx.doi.org/10.1371/journal.pone.0226880 (cit. on pp. 39, 51).

[42] Xiao Fu et al., « Geowizard: Unleashing the diffusion priors for 3d geometry estimation from a single image », *in*: *arXiv preprint arXiv:2403.12013* (2024) (cit. on p. 144).

[43] Jianfeng Gao et al., « K-VIL: Keypoints-based Visual Imitation Learning », *in*: *arXiv preprint arXiv:2209.03277* (2022) (cit. on p. 118).

[44] Shuo Gao, Yanning Dai, and Arokia Nathan, « Tactile and vision perception for intelligent humanoids », *in*: *Advanced Intelligent Systems* 4.*2* (2022), p. 2100074 (cit. on p. 61).

[45] Raman Garimella et al., « Capturing joint angles of the off-site human body », *in*: *2018 IEEE SENSORS*, IEEE, 2018, pp. 1–4 (cit. on p. 66).

[46] Andreas Geiger, Philip Lenz, and Raquel Urtasun, « Are we ready for autonomous driving? The KITTI vision benchmark suite », *in*: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361, DOI: 10.1109/CVPR.2012.6248074 (cit. on p. 38).

[47] Andreas Geiger et al., « Vision meets robotics: The kitti dataset », *in*: *The International Journal of Robotics Research* 32.*11* (2013), pp. 1231–1237 (cit. on p. 144).

[48] Ross Girshick, « Fast r-cnn », *in*: *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448 (cit. on p. 18).

[49] Ross Girshick et al., « Region-based convolutional networks for accurate object detection and segmentation », *in*: *IEEE transactions on pattern analysis and machine intelligence* 38.*1* (2015), pp. 142–158 (cit. on pp. 17, 22).

[50] Ross Girshick et al., « Rich feature hierarchies for accurate object detection and semantic segmentation », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587 (cit. on pp. 17, 22).

[51] Michael Gleicher, « Animation from observation: Motion capture and motion editing », *in*: *ACM SIGGRAPH Computer Graphics* 33.*4* (1999), pp. 51–54 (cit. on p. 66).

[52] Ian Goodfellow et al., « Generative adversarial nets », *in*: *Advances in neural information processing systems* 27 (2014) (cit. on p. 139).

[53] Leo Grady, « Random walks for image segmentation », *in*: *IEEE transactions on pattern analysis and machine intelligence* 28.*11* (2006), pp. 1768–1783 (cit. on p. 21).

[54] Kristen Grauman et al., « Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19383–19400 (cit. on p. 72).

[55] Kristen Grauman et al., « Ego4d: Around the world in 3,000 hours of egocentric video », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18995–19012 (cit. on p. 7).

[56] Xiuye Gu et al., « Open-vocabulary object detection via vision and language knowledge distillation », *in*: *arXiv preprint arXiv:2104.13921* (2021) (cit. on pp. 20, 141).

[57] Yulan Guo et al., « Deep learning for 3d point clouds: A survey », *in*: *IEEE transactions on pattern analysis and machine intelligence* (2020) (cit. on p. 38).

[58] Abhishek Gupta et al., « Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning », *in*: *arXiv preprint arXiv:1910.11956* (2019) (cit. on pp. 11, 14).

[59] Agrim Gupta, Piotr Dollar, and Ross Girshick, « Lvis: A dataset for large vocabulary instance segmentation », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5356–5364 (cit. on p. 140).

[60] Tuomas Haarnoja et al., « Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor », *in*: *International conference on machine learning*, PMLR, 2018, pp. 1861–1870 (cit. on p. 47).

[61] Frederik Hagelskjær and Anders Glent Buch, « Pointvotenet: Accurate object detection and 6 dof pose estimation in point clouds », *in*: *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, pp. 2641–2645 (cit. on p. 38).

[62] Shreyas Hampali et al., « Honnotate: A method for 3d annotation of hand and object poses », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3196–3206 (cit. on p. 72).

[63] Richard I Hartley and Peter Sturm, « Triangulation », *in*: *Computer vision and image understanding* 68.2 (1997), pp. 146–157 (cit. on pp. 66, 67).

[64] Matthew Hausknecht and Peter Stone, « On-policy vs. off-policy updates for deep reinforcement learning », *in*: *Deep Reinforcement Learning: Frontiers and Challenges, IJCAI 2016 Workshop*, AAAI Press New York, NY, USA, 2016 (cit. on p. 54).

[65] Kaiming He et al., « Deep residual learning for image recognition », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778 (cit. on pp. 8, 9).

[66] Kaiming He et al., « Mask r-cnn », *in*: *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969 (cit. on p. 22).

[67] Kaiming He et al., « Masked autoencoders are scalable vision learners », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16000–16009 (cit. on pp. 7, 10).

[68] Kaiming He et al., « Momentum contrast for unsupervised visual representation learning », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738 (cit. on pp. 7, 9).

[69] Kaiming He et al., « Spatial pyramid pooling in deep convolutional networks for visual recognition », *in*: *IEEE transactions on pattern analysis and machine intelligence* 37.*9* (2015), pp. 1904–1916 (cit. on p. 17).

[70] Hadi Heidari, Nicoleta Wacker, and Ravinder Dahiya, « Bending induced electrical response variations in ultra-thin flexible chips and device modeling », *in*: *Applied Physics Reviews* 4.*3* (2017) (cit. on p. 39).

[71] João F. Henriques, Rui Caseiro, and Jorge Batista, « Globally optimal solution to multi-object tracking with merged measurements », *in*: *2011 International Conference on Computer Vision*, 2011, pp. 2470–2477, DOI: 10.1109/ICCV.2011.6126532 (cit. on p. 27).

[72] Ashley Hill et al., *Stable Baselines*, https://github.com/hill-a/stable-baselines, 2018 (cit. on p. 49).

[73] Seunghoon Hong, Suha Kwak, and Bohyung Han, « Orderless Tracking through Model-Averaged Posterior Estimation », *in*: *2013 IEEE International Conference on Computer Vision*, 2013, pp. 2296–2303, DOI: 10.1109/ICCV.2013.285 (cit. on p. 27).

[74] Weiming Hu et al., « Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.*12* (2012), Cited by: 128, pp. 2420–2440, DOI: 10.1109/TPAMI.2012.42, URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84867787204&doi=10.1109%2fTPAMI.2012.42&partnerID=40&md5=87233708c95ccf9c3663314dc2e8b1c9 (cit. on p. 27).

[75] Lucie A Huet, John W Rudnicki, and Mitra JZ Hartmann, « Tactile sensing with whiskers of various shapes: Determining the three-dimensional location of object contact based on mechanical signals at the whisker base », *in*: *Soft robotics* 4.*2* (2017), pp. 88–102 (cit. on p. 39).

[76] Catalin Ionescu et al., « Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments », *in*: *IEEE transactions on pattern analysis and machine intelligence* 36.*7* (2013), pp. 1325–1339 (cit. on pp. 73, 74, 77, 88, 96).

[77] Karim Iskakov et al., « Learnable triangulation of human pose », *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7718–7727 (cit. on pp. 66, 67, 86, 89).

[78] Nawid Jamali et al., « Active perception: Building objects' models using tactile exploration », *in*: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2016, pp. 179–185 (cit. on p. 39).

[79] Stephen James and Andrew J Davison, « Q-attention: Enabling efficient learning for vision-based robotic manipulation », *in*: *IEEE Robotics and Automation Letters* 7.*2* (2022), pp. 1612–1619 (cit. on p. 124).

[80] Stephen James et al., « Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13739–13748 (cit. on pp. 124–126, 128).

[81] Stephen James et al., « Rlbench: The robot learning benchmark learning environment », *in*: *IEEE Robotics and Automation Letters* 5.*2* (2020), pp. 3019–3026 (cit. on pp. 91, 97, 123).

[82] Sheng Jin et al., « Whole-body human pose estimation in the wild », *in*: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, Springer, 2020, pp. 196–214 (cit. on pp. 71, 73, 77).

[83] Hanbyul Joo et al., « Panoptic studio: A massively multiview system for social motion capture », *in*: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3334–3342 (cit. on pp. 73, 74).

[84] Mohsen Kaboli et al., « Tactile-based active object discrimination and target object search in an unknown workspace », *in*: *Autonomous Robots* 43.*1* (2019), pp. 123–152 (cit. on p. 39).

[85] Bingyi Kang et al., « Few-shot object detection via feature reweighting », *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8420–8429 (cit. on p. 32).

[86] Bingxin Ke et al., « Repurposing diffusion-based image generators for monocular depth estimation », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9492–9502 (cit. on p. 144).

[87] Alexander Kirillov et al., « Segment Anything », *in*: *arXiv:2304.02643* (2023) (cit. on pp. 23, 141).

[88] Ilya Kostrikov, Denis Yarats, and Rob Fergus, « Image augmentation is all you need: Regularizing deep reinforcement learning from pixels », *in*: *arXiv preprint arXiv:2004.13649* (2020) (cit. on p. 124).

[89] Ilya Kostrikov et al., « Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning », *in*: *International Conference on Learning Representations*, 2019, URL: https://openreview.net/forum?id=Hk4fpoA5Km (cit. on p. 124).

[90] Danica Kragic, Markus Vincze, et al., « Vision for robotics », *in*: *Foundations and Trends® in Robotics* 1.*1* (2009), pp. 1–78 (cit. on p. 5).

[91] Ivan Krasin et al., « Openimages: A public dataset for large-scale multi-label and multi-class image classification », *in*: *Dataset available from https://github. com/openimages* 2.*3* (2017), p. 18 (cit. on p. 140).

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, « Imagenet classification with deep convolutional neural networks », *in*: *Advances in neural information processing systems* 25 (2012) (cit. on p. 17).

[93] Tejas D Kulkarni et al., « Unsupervised learning of object keypoints for perception and control », *in*: *Advances in neural information processing systems* 32 (2019) (cit. on p. 118).

[94] Aviral Kumar et al., « Stabilizing off-policy q-learning via bootstrapping error reduction », *in*: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on p. 117).

[95] Vladimir Kuts et al., « Adaptive industrial robots using machine vision », *in*: *ASME International Mechanical Engineering Congress and Exposition*, vol. 52019, American Society of Mechanical Engineers, 2018, V002T02A093 (cit. on p. 35).

[96] Mike Lambeta et al., « Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation », *in*: *IEEE Robotics and Automation Letters* 5.*3* (2020), pp. 3838–3845 (cit. on pp. 39, 61).

[97] Hei Law and Jia Deng, « Cornernet: Detecting objects as paired keypoints », *in*: *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750 (cit. on p. 19).

[98] Marius Leordeanu and Martial Hebert, « A spectral technique for correspondence problems using pairwise constraints », *in*: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, IEEE, 2005, pp. 1482–1489 (cit. on p. 21).

[99] Wenhao Li et al., « Mhformer: Multi-hypothesis transformer for 3d human pose estimation », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13147–13156 (cit. on pp. 94, 99, 121).

[100] Zhengqi Li and Noah Snavely, « Megadepth: Learning single-view depth prediction from internet photos », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2041–2050 (cit. on p. 144).

[101] Zhuwen Li, Qifeng Chen, and Vladlen Koltun, « Interactive image segmentation with latent diversity », *in*: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 577–585 (cit. on p. 23).

[102] Tingting Liang et al., « Cbnet: A composite backbone network architecture for object detection », *in*: *IEEE Transactions on Image Processing* 31 (2022), pp. 6893–6906 (cit. on p. 20).

[103] Tsung-Yi Lin et al., « Feature pyramid networks for object detection », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125 (cit. on pp. 11, 18, 22).

[104] Tsung-Yi Lin et al., « Focal loss for dense object detection », *in*: *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988 (cit. on p. 19).

[105] Tsung-Yi Lin et al., « Microsoft coco: Common objects in context », *in*: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755 (cit. on pp. 16, 20, 71, 73, 140).

[106] Shilong Liu et al., « Grounding dino: Marrying dino with grounded pre-training for open-set object detection », *in*: *arXiv preprint arXiv:2303.05499* (2023) (cit. on p. 21).

[107] Wei Liu et al., « Ssd: Single shot multibox detector », *in*: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37 (cit. on p. 19).

[108] Yongcheng Liu et al., « Densepoint: Learning densely contextual representation for efficient point cloud processing », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5239–5248 (cit. on p. 38).

[109] Yongcheng Liu et al., « Relation-shape convolutional neural network for point cloud analysis », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8895–8904 (cit. on p. 38).

[110] Ze Liu et al., « Swin transformer: Hierarchical vision transformer using shifted windows », *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10012–10022 (cit. on pp. 19, 22, 144).

[111] Jonathan Long, Evan Shelhamer, and Trevor Darrell, « Fully convolutional networks for semantic segmentation », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440 (cit. on pp. 21, 22).

[112] Matthew Loper et al., « SMPL: A skinned multi-person linear model », *in*: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 851–866 (cit. on p. 70).

[113] Shan Luo et al., « Robotic tactile perception of object properties: A review », *in*: *Mechatronics* 48 (2017), pp. 54–67 (cit. on pp. 35, 39).

[114] Wenhan Luo et al., « Multiple object tracking: A literature review », *in*: *Artificial intelligence* 293 (2021), p. 103448 (cit. on p. 27).

[115] Yecheng Jason Ma et al., « Vip: Towards universal visual reward and representation via value-implicit pre-training », *in*: *arXiv preprint arXiv:2210.00030* (2022) (cit. on pp. 12–14).

[116] Dimitrios Mallis et al., « From Keypoints to Object Landmarks via Self-Training Correspondence: A novel approach to Unsupervised Landmark Discovery », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023) (cit. on p. 118).

[117] Ajay Mandlekar et al., « What matters in learning from offline human demonstrations for robot manipulation », *in*: *arXiv preprint arXiv:2108.03298* (2021) (cit. on pp. 7, 8, 114).

[118] Dushyant Mehta et al., « Monocular 3d human pose estimation in the wild using improved cnn supervision », *in*: *2017 international conference on 3D vision (3DV)*, IEEE, 2017, pp. 506–516 (cit. on p. 74).

[119] Suraj Nair et al., « R3m: A universal visual representation for robot manipulation », *in*: *arXiv preprint arXiv:2203.12601* (2022) (cit. on pp. 7, 11, 14).

[120] Soroush Nasiriany, Huihan Liu, and Yuke Zhu, « Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks », *in*: *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 7477–7484 (cit. on p. 117).

[121] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, « Representation learning with contrastive predictive coding », *in*: *arXiv preprint arXiv:1807.03748* (2018) (cit. on pp. 10, 12).

[122] Maxime Oquab et al., « Dinov2: Learning robust visual features without supervision », *in*: *arXiv preprint arXiv:2304.07193* (2023) (cit. on p. 144).

[123] Roi Otero et al., « Mobile indoor mapping technologies: A review », *in*: *Automation in Construction* 120 (2020), p. 103399 (cit. on p. 35).

[124] Anshul Paigwar et al., « Attentional pointnet for 3d-object detection in point clouds », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0 (cit. on p. 38).

[125] Simone Parisi et al., « The unsurprising effectiveness of pre-trained vision models for control », *in*: *international conference on machine learning*, PMLR, 2022, pp. 17359–17371 (cit. on pp. 7, 10, 11).

[126] Georgios Pavlakos et al., « Harvesting multiple views for marker-less 3d human pose annotations », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6988–6997 (cit. on p. 67).

[127] Andry Maykol Pinto et al., « Evaluation of depth sensors for robotic applications », *in*: *2015 IEEE international conference on autonomous robot systems and competitions*, IEEE, 2015, pp. 139–143 (cit. on p. 146).

[128] Charles R Qi et al., « Pointnet: Deep learning on point sets for 3d classification and segmentation », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660 (cit. on pp. 38, 44, 51).

[129] Charles R Qi et al., « Pointnet++: Deep hierarchical feature learning on point sets in a metric space », *in*: *arXiv preprint arXiv:1706.02413* (2017) (cit. on p. 38).

[130] Chen Qian et al., « Realtime and robust hand tracking from depth », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1106–1113 (cit. on p. 71).

[131] Alec Radford et al., « Language models are unsupervised multitask learners », *in*: *OpenAI blog* 1.*8* (2019), p. 9 (cit. on p. 7).

[132] Alec Radford et al., « Learning transferable visual models from natural language supervision », *in*: *International conference on machine learning*, PMLR, 2021, pp. 8748–8763 (cit. on pp. 8, 16, 20).

[133] René Ranftl et al., « Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer », *in*: *IEEE transactions on pattern analysis and machine intelligence* 44.*3* (2020), pp. 1623–1637 (cit. on p. 144).

[134] Kanishka Rao et al., « Rl-cyclegan: Reinforcement learning aware simulation-to-real », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11157–11166 (cit. on p. 139).

[135] Siddharth Reddy, Anca D Dragan, and Sergey Levine, « Sqil: Imitation learning via reinforcement learning with sparse rewards », *in*: *arXiv preprint arXiv:1905.11108* (2019) (cit. on p. 124).

[136] Joseph Redmon et al., « You only look once: Unified, real-time object detection », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788 (cit. on p. 18).

[137] Shaoqing Ren et al., « Faster r-cnn: Towards real-time object detection with region proposal networks », *in*: *Advances in neural information processing systems* 28 (2015) (cit. on pp. 18, 22).

[138] Tianhe Ren et al., *Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks*, 2024, arXiv: 2401.14159 [cs.CV] (cit. on pp. 22, 23).

[139] Kevin Riou, Kevin Subrin, and Patrick Le Callet, « REINFORCEMENT LEARNING BASED POINT-CLOUD ACQUISITION AND RECOGNITION USING EXPLORATION-CLASSIFICATION REWARD COMBINATION », *in*: *IEEE International Conference on Multimedia and Expo 2022*, Taipei, Taiwan, July 2022, URL: https://hal.archives-ouvertes.fr/hal-03690362 (cit. on pp. 36, 39, 40, 49, 52).

[140] Kevin Riou, Kevin Subrin, and Patrick Le Callet, « Reinforcement Learning Based Point-Cloud Acquisition and Recognition Using Exploration-Classification Reward Combination », *in*: *2022 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2022, pp. 1–6 (cit. on p. 34).

[141] Kevin Riou et al., « Few-shot object detection in real life: case study on auto-harvest », *in*: *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2020, pp. 1–6 (cit. on p. 32).

[142] Kevin Riou et al., « From Temporal-evolving to Spatial-fixing: A Keypoints-based Learning Paradigm for Visual Robotic Manipulation », *in*: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 1728–1734 (cit. on p. 116).

[143] Kevin Riou et al., « Reinforcement Learning Based Tactile Sensing for Active point cloud Acquisition, Recognition and Localization », *in*: *IEEE Journal of Selected Topics in Signal Processing* (2024) (cit. on p. 34).

[144] Kevin Riou et al., « Seeing By Haptic Glance: Reinforcement Learning Based 3d Object Recognition », *in*: *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3637–3641, DOI: 10.1109/ICIP42928.2021.9506734 (cit. on pp. 34, 39, 44, 49).

[145] Javier Romero, Dimitrios Tzionas, and Michael J Black, « Embodied hands: Modeling and capturing hands and bodies together », *in*: *arXiv preprint arXiv:2201.02610* (2022) (cit. on pp. 70–72).

[146] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, « U-net: Convolutional networks for biomedical image segmentation », *in*: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, Springer, 2015, pp. 234–241 (cit. on p. 22).

[147] Olga Russakovsky et al., « Imagenet large scale visual recognition challenge », *in*: *International journal of computer vision* 115 (2015), pp. 211–252 (cit. on p. 140).

[148] Walter J Scheirer et al., « Toward open set recognition », *in*: *IEEE transactions on pattern analysis and machine intelligence* 35.7 (2012), pp. 1757–1772 (cit. on p. 16).

[149] Ramprasaath R Selvaraju et al., « Grad-cam: Visual explanations from deep networks via gradient-based localization », *in*: *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626 (cit. on p. 11).

[150] Pierre Sermanet et al., « Time-contrastive networks: Self-supervised learning from video », *in*: *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 1134–1141 (cit. on p. 11).

[151] Rutav Shah and Vikash Kumar, « Rrl: Resnet as representation for reinforcement learning », *in*: *arXiv preprint arXiv:2107.03380* (2021) (cit. on pp. 7, 8).

[152] Hui Shuai, Lele Wu, and Qingshan Liu, « Adaptive Multi-View and Temporal Fusing Transformer for 3D Human Pose Estimation », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.*4* (2023), pp. 4122–4135, DOI: 10.1109/TPAMI.2022.3188716 (cit. on pp. 70, 74, 76, 78).

[153] Leonid Sigal, Alexandru O Balan, and Michael J Black, « Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion », *in*: *International journal of computer vision* 87.*1-2* (2010), pp. 4–27 (cit. on pp. 73, 74, 88, 97, 99, 101).

[154] Nathan Silberman et al., « Indoor segmentation and support inference from rgbd images », *in*: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, Springer, 2012, pp. 746–760 (cit. on p. 144).

[155] Bi Song et al., « A Stochastic Graph Evolution Framework for Robust Multi-target Tracking », *in*: *Computer Vision – ECCV 2010*, ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 605–619, ISBN: 978-3-642-15549-9 (cit. on p. 27).

[156] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao, « Sun rgb-d: A rgb-d scene understanding benchmark suite », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576 (cit. on p. 38).

[157] Alessandra Spreafico et al., « The ipad pro built-in lidar sensor: 3d rapid mapping tests and quality assessment », *in*: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2021), pp. 63–69 (cit. on p. 35).

[158] Robin Strudel et al., « Learning to combine primitive skills: A step towards versatile robotic manipulation § », *in*: *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 4637–4643 (cit. on p. 117).

[159] Ioan A. Sucan, Mark Moll, and Lydia E. Kavraki, « The Open Motion Planning Library », *in*: *IEEE Robotics & Automation Magazine* 19.*4* (2012), pp. 72–82, DOI: 10.1109/MRA.2012.2205651 (cit. on p. 123).

[160] Ke Sun et al., « Deep high-resolution representation learning for human pose estimation », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5693–5703 (cit. on pp. 75, 77, 117, 118, 121).

[161] Yongbin Sun et al., « Pointgrow: Autoregressively learned point cloud generation with self-attention », *in*: *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 61–70 (cit. on pp. 38, 45).

[162] Supasorn Suwajanakorn et al., « Discovery of latent 3d keypoints via end-to-end geometric reasoning », *in*: *Advances in neural information processing systems* 31 (2018) (cit. on pp. 91, 118).

[163] Vladimir Tadic et al., « Perspectives of realsense and zed depth sensors for robotic vision applications », *in*: *Machines* 10.*3* (2022), p. 183 (cit. on p. 146).

[164] Saeid Asgari Taghanaki et al., « RobustPointSet: A Dataset for Benchmarking Robustness of Point Cloud Classifiers », *in*: *arXiv preprint arXiv:2011.11572* (2020) (cit. on pp. 36, 38, 51).

[165] Gusi Te et al., « Rgcnn: Regularized graph cnn for point cloud segmentation », *in*: *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 746–754 (cit. on p. 38).

[166] Olga Lucia Lopera Tellez and Bart Scheers, « Ground-penetrating radar for close-in mine detection », *in*: *Mine Action-The Research Experience of the Royal Military Academy of Belgium* (2017), p. 82 (cit. on p. 35).

[167] Yaroslav Tenzer, Leif P Jentoft, and Robert D Howe, « The feel of MEMS barometers: Inexpensive and easily customized tactile array sensors », *in*: *IEEE Robotics & Automation Magazine* 21.*3* (2014), pp. 89–95 (cit. on pp. 39, 61).

[168] Jonathan Tompson et al., « Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks », *in*: *ACM Transactions on Graphics* 33 (Aug. 2014) (cit. on p. 71).

[169] Matthew Trumble et al., « Total capture: 3d human pose estimation fusing video and inertial sensors », *in*: *Proceedings of 28th British Machine Vision Conference*, 2017, pp. 1–13 (cit. on p. 74).

[170] Roger Y Tsai, Reimar K Lenz, et al., « A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration », *in*: *IEEE Transactions on robotics and automation* 5.*3* (1989), pp. 345–358 (cit. on p. 29).

[171] Hanyue Tu, Chunyu Wang, and Wenjun Zeng, « Voxelpose: Towards multi-camera 3d human pose estimation in wild environment », *in*: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, Springer, 2020, pp. 197–212 (cit. on p. 67).

[172] Jasper RR Uijlings et al., « Selective search for object recognition », *in*: *International journal of computer vision* 104 (2013), pp. 154–171 (cit. on p. 17).

[173] Ashish Vaswani et al., « Attention is all you need », *in*: *Advances in neural information processing systems* 30 (2017) (cit. on p. 19).

[174] Paul Viola and Michael Jones, « Rapid object detection using a boosted cascade of simple features », *in*: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, Ieee, 2001, pp. I–I (cit. on p. 17).

[175] Kentaro Wada et al., « Morefusion: Multi-object reasoning for 6d pose estimation from volumetric fusion », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14540–14549 (cit. on p. 118).

[176] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, « YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 7464–7475 (cit. on p. 20).

[177] Panqu Wang et al., « Understanding convolution for semantic segmentation », *in*: *2018 IEEE winter conference on applications of computer vision (WACV)*, Ieee, 2018, pp. 1451–1460 (cit. on p. 22).

[178] Qiang Wang et al., « Irs: A large naturalistic indoor robotics stereo dataset to train deep models for disparity and surface normal estimation », *in*: *2021 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2021, pp. 1–6 (cit. on p. 144).

[179] Tianyu Wang et al., « Cross-Embodiment Robot Manipulation Skill Transfer using Latent Space Alignment », *in*: *arXiv preprint arXiv:2406.01968* (2024) (cit. on p. 139).

[180] Wenshan Wang et al., « Tartanair: A dataset to push the limits of visual slam », *in*: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 4909–4916 (cit. on p. 144).

[181] Xin Wang et al., « Frustratingly Simple Few-Shot Object Detection », *in*: *International Conference on Machine Learning*, PMLR, 2020, pp. 9919–9928 (cit. on p. 32).

[182] Xuan Wang et al., « Point cloud segmentation from iPhone-based LiDAR sensors using the tensor feature », *in*: *Applied Sciences* 12.*4* (2022), p. 1817 (cit. on p. 35).

[183] Yue Wang et al., « Dynamic graph cnn for learning on point clouds », *in*: *ACM Transactions on Graphics (tog)* 38.*5* (2019), pp. 1–12 (cit. on p. 38).

[184] Wenxuan Wu, Zhongang Qi, and Li Fuxin, « Pointconv: Deep convolutional networks on 3d point clouds », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630 (cit. on p. 38).

[185] Zhirong Wu et al., « 3d shapenets: A deep representation for volumetric shapes », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920 (cit. on p. 38).

[186] Ke Xian et al., « Structure-guided ranking loss for single image depth prediction », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 611–620 (cit. on p. 144).

[187] Yu Xiang, Alexandre Alahi, and Silvio Savarese, « Learning to Track: Online Multi-object Tracking by Decision Making », *in*: *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4705–4713, DOI: 10.1109/ICCV.2015.534 (cit. on p. 27).

[188] Yu Xiang et al., « Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes », *in*: *arXiv preprint arXiv:1711.00199* (2017) (cit. on p. 72).

[189] Tete Xiao et al., « Masked visual pre-training for motor control », *in*: *arXiv preprint arXiv:2203.06173* (2022) (cit. on p. 7).

[190] Haoyu Xiong et al., « Learning by watching: Physical imitation of manipulation skills from human videos », *in*: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 7827–7834 (cit. on p. 139).

[191] Ning Xu et al., « Deep interactive object selection », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 373–381 (cit. on p. 23).

[192] Zhuo Xu et al., « Toward modularization of neural network autonomous driving policy using parallel attribute networks », *in*: *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 1400–1407 (cit. on p. 118).

[193] Chuanyu Yang et al., « Multi-expert learning of adaptive legged locomotion », *in*: *Science Robotics* 5.*49* (2020), eabb2174 (cit. on p. 118).

[194] Jingyun Yang et al., « Learning periodic tasks from human demonstrations », *in*: *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 8658–8665 (cit. on p. 118).

[195] Lihe Yang et al., « Depth Anything V2 », *in*: *arXiv preprint arXiv:2406.09414* (2024) (cit. on p. 144).

[196] Lihe Yang et al., « Depth anything: Unleashing the power of large-scale unlabeled data », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10371–10381 (cit. on pp. 31, 144).

[197] Zongxin Yang and Yi Yang, « Decoupling features in hierarchical propagation for video object segmentation », *in*: *Advances in Neural Information Processing Systems* 35 (2022), pp. 36324–36336 (cit. on p. 27).

[198] Denis Yarats et al., « Improving sample efficiency in model-free reinforcement learning from images », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 12, 2021, pp. 10674–10681 (cit. on p. 124).

[199] Ju Hong Yoon et al., « Online multi-object tracking via structural constraint event aggregation », *in*: *Proceedings of the IEEE Conference on computer vision and pattern recognition*, 2016, pp. 1392–1400 (cit. on p. 27).

[200] Yuhang Zang et al., « Open-vocabulary detr with conditional matching », *in*: *European Conference on Computer Vision*, Springer, 2022, pp. 106–122 (cit. on p. 20).

[201] Alireza Zareian et al., « Open-vocabulary object detection using captions », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14393–14402 (cit. on p. 16).

[202] Chuyu Zhang et al., « Cascaded Sparse Feature Propagation Network for Interactive Segmentation », *in*: *arXiv preprint arXiv:2203.05145* (2022) (cit. on p. 23).

[203] Lu Zhang and Laurens van der Maaten, « Structure Preserving Object Tracking », *in*: *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1838–1845, DOI: 10.1109/CVPR.2013.240 (cit. on p. 27).

[204] Song-Hai Zhang et al., « Pose2seg: Detection free human instance segmentation », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 889–898 (cit. on p. 73).

[205] Haojie Zhao et al., « Arkittrack: a new diverse dataset for tracking using mobile RGB-D data », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5126–5135 (cit. on p. 146).

[206] Hengshuang Zhao et al., « PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing », *in*: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5560–5568, DOI: 10.1109/CVPR.2019.00571 (cit. on p. 38).

[207] Yihao Zhao, Ruihai Wu, and Hao Dong, « Unpaired image-to-image translation using adversarial consistency loss », *in*: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, Springer, 2020, pp. 800–815 (cit. on p. 139).

[208] Wu Zheng et al., « Cia-ssd: Confident iou-aware single-stage object detector from point cloud », *in*: *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 4, 2021, pp. 3555–3562 (cit. on p. 38).

[209] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun, « Open3D: A Modern Library for 3D Data Processing », *in*: *arXiv:1801.09847* (2018) (cit. on p. 30).

[210] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl, « Objects as points », *in*: *arXiv preprint arXiv:1904.07850* (2019) (cit. on p. 19).

[211] Christian Zimmermann et al., « Freihand: A dataset for markerless capture of hand pose and shape from single rgb images », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 813–822 (cit. on p. 72).

[212] Zhengxia Zou et al., « Object detection in 20 years: A survey », *in*: *Proceedings of the IEEE* 111.*3* (2023), pp. 257–276 (cit. on p. 17).

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Nantes Université

Titre : titre (en français).....

Mot clés : de 3 à 6 mots clefs

Résumé : Eius populus ab incunabulis primis ad usque pueritiae tempus extremum, quod annis circumcluditur fere trecentis, circummurana pertulit bella, deinde aetatem ingressus adultam post multiplices bellorum aerumnas Alpes transcendit et fretum, in iuvenem erectus et virum ex omni plaga quam orbis ambit inmensus, reportavit laureas et triumphos, iamque vergens in senium et nomine solo aliquotiens vincens ad tranquilliora vitae discessit. Hoc inmaturo interitu ipse quoque sui pertaesus excessit e vita aetatis nono anno atque vicensimo cum quadriennio imperasset. natus apud Tuscos in Massa Veternensi, patre Constantio Constantini fratre imperatoris, matreque Galla. Thalassius vero ea tempestate praefectus praetorio praesens ipse quoque adrogantis ingenii, considerans incitationem eius ad multorum augeri discrimina, non maturitate vel consiliis mitigabat, ut aliquotiens celsae potestates iras principum molliverunt, sed adversando iurgandoque cum parum congrueret, eum ad rabiem potius evibrabat, Augustum actus eius exaggerando creberrime docens, idque, incertum qua mente, ne lateret adfectans. quibus mox Caesar acrius efferatus, velut contumaciae quoddam vexillum altius erigens, sine respectu salutis alienae vel suae ad vertenda opposita instar rapidi fluminis irrevocabili impetu ferebatur. Hae duae provinciae bello quondam piratico catervis mixtae praedonum.

Title: titre (en anglais).....

Keywords: de 3 à 6 mots clefs

Abstract: Eius populus ab incunabulis primis ad usque pueritiae tempus extremum, quod annis circumcluditur fere trecentis, circummurana pertulit bella, deinde aetatem ingressus adultam post multiplices bellorum aerumnas Alpes transcendit et fretum, in iuvenem erectus et virum ex omni plaga quam orbis ambit inmensus, reportavit laureas et triumphos, iamque vergens in senium et nomine solo aliquotiens vincens ad tranquilliora vitae discessit. Hoc inmaturo interitu ipse quoque sui pertaesus excessit e vita aetatis nono anno atque vicensimo cum quadriennio imperasset. natus apud Tuscos in Massa Veternensi, patre Constantio Constantini fratre imperatoris, matreque Galla. Thalassius vero ea tempestate praefectus praetorio praesens ipse quoque adrogantis ingenii, considerans incitationem eius ad multorum augeri discrimina, non maturitate vel consiliis mitigabat, ut aliquotiens celsae potestates iras principum molliverunt, sed adversando iurgandoque cum parum congrueret, eum ad rabiem potius evibrabat, Augustum actus eius exaggerando creberrime docens, idque, incertum qua mente, ne lateret adfectans. quibus mox Caesar acrius efferatus, velut contumaciae quoddam vexillum altius erigens, sine respectu salutis alienae vel suae ad vertenda opposita instar rapidi fluminis irrevocabili impetu ferebatur. Hae duae provinciae bello quondam piratico catervis mixtae praedonum.