University of Modena and Reggio Emilia
Department of Engineering "Enzo Ferrari"

XXXVI cycle of the International Doctorate School in
Information and Communication Technologies

From Images to 3D Space:
The Role of Semantic Keypoints for 3D Perception

Ph.D. Dissertation
in Computer Engineering and Science

Alessandro Simoni

Advisor: Prof. Roberto Vezzani
Director of the School: Prof. Luigi Rovati

Modena, 2024

Advisor:

Prof. Roberto Vezzani	University of Modena and Reggio Emilia

Director of the School:

Prof. Luigi Rovati	University of Modena and Reggio Emilia

Review Committee:

Prof. Lorenzo Seidenari	University of Florence
Silvia Zuffi, Ph.D.	IMATI-CNR

The work described in this thesis has been carried out within the International Doctorate in Information and Communication Technologies, at the AImageLab research laboratory of the University of Modena and Reggio Emilia.

This dissertation was typeset by the author using LaTeX 2_ε, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface.

To the most important people in my life, my parents.

They have always supported all my choices
and helped me achieve all my goals.

I can not be more proud of them.

From Images to 3D Space:
The Role of Semantic Keypoints for 3D Perception

Abstract

One of the goals of the Computer Vision community is to comprehend human 3D perception through 2D representations like images and videos. Extracting robust 3D insights from these analyses is a significant challenge. This dissertation focuses on the keypoint-based 3D representation, exploring applications in different real-world scenarios. Unlike traditional pointwise feature descriptors like ORB or SIFT, semantic keypoints establish correlations between specific 3D points belonging to a rigid or articulated object. Recent advances in Deep Learning, particularly in 2D keypoints detection, have paved the way for addressing complex 3D vision problem. This thesis demonstrates the application of these methods in autonomous driving and video surveillance, showcasing their robustness and precision in bridging the gap between 2D image planes and the 3D world.

In the automotive context, our investigation centers on the tasks of novel view synthesis and 3D reconstruction of vehicles within urban scenes. A 3D representation of a vehicle in a scene can be valuable for traffic analysis and accident prevention. To achieve this, we design a method leveraging a 2D keypoint localization network to augment visual features for accurate classification of 3D vehicle models. Ensuring a robust classification, we study how to improve the generation of synthetic vehicles from unseen novel views through a deep learning pipeline trained on a collection of single-view images. Additionally, to explore more sophisticated techniques for 3D object reconstruction from images, we introduce a deep learning architecture capable of reconstructing objects across multiple categories. This approach is trained on a dataset of single-view images and involves

the deformation of explicit 3D representations.

The second research area is focused on predicting the 3D skeletons of both humans and robots, observed from an external perspective, such as a video surveillance camera. The keypoints in this context are integrated into the definition of a skeleton, depicted as a graph of semantic points. Our initial focus is on the robotics domain, where an intelligent system for predicting 3D skeletons can be crucial for safety in collaborative environments shared by humans and robots. Given the challenges of obtaining real datasets in robotics, we emphasize the role of simulation. Our approach involves collecting a synthetic and real dataset, addressing the 3D pose estimation task through a double heatmap-based representation. We explore the domain gap between the synthetic and real data, utilizing depth maps to enhance accuracy. Introducing temporal cues, our pipeline embraces the novel Pose Nowcasting paradigm, where predicting future poses serves as an auxiliary task to refine current pose precision. Shifting to the human scenario, we propose a pose refinement framework based on depth map analysis. Simultaneously, our investigation extends to Human-Computer Interaction, where we present an unsupervised method for detecting and classifying dynamic hand gestures using data from a motion tracking sensor.

This thesis seeks to make a valuable contribution to the intersection of 3D Computer Vision and Deep Learning across various domains. Following an overview of the existing state-of-the-art in 3D reconstruction and 3D pose estimation tasks, we present our proposed methods with a comprehensive technical explanation supported by a detailed experimental investigation conducted on benchmark datasets widely acknowledged in the literature.

Dalle Immagini allo Spazio 3D:
il Ruolo dei Punti Chiave Semantici per la Percezione 3D

Sommario

Uno degli obiettivi della Computer Vision è quello di comprendere la percezione 3D umana attraverso rappresentazioni 2D come immagini e video. Estrarre informazioni 3D robuste da quest'analisi è una sfida significativa. Questa tesi si concentra su rappresentazioni 3D basate su punti chiave, esplorando applicazioni in differenti scenari reali. Differentemente dai tradizionali descrittori puntuali come ORB o SIFT, i punti chiave semantici stabiliscono correlazioni tra specifici punti 3D appartenenti a oggetti rigidi o articolati. I recenti sviluppi in Deep Learning, in particolar modo nel rilevamento di punti chiave 2D, hanno aperto la strada per affrontare problemi complessi di visione 3D. Questa tesi dimostra l'applicazione di questi metodi nella guida autonoma e nella videosorveglianza, evidenziando la robustezza e la precisione nel ridurre il divario tra il piano immagine 2D e il mondo 3D.

Nel conteso automotive, la nostro indagine si concentra sui problemi di sintetizzazione di nuove viste e di ricostruzione 3D di veicoli in ambienti urbani. La rappresentazione 3D di un veicolo in una scena può essere utile per l'analisi del traffico e la prevenzione di incidenti. Perciò, progettiamo un metodo che sfrutta la localizzazione di punti chiave 2D in modo da aumentare le caratteristiche visuali per l'accurata classificazione di modelli 3D di veicoli. Assicurando una classificazione robusta, studiamo come migliorare la generazione di veicoli sintetici da punti di vista non visti attraverso un sistema di Deep Learning trainato su un insieme di immagini da singoli punti di vista. In aggiunta, per esplorare tecniche più sofisticate di ricostruzione 3D di oggetti da immagini, introduciamo un'architettura di Deep Learning capace di ricostruire oggetti di diverse categorie.

Questo approccio è allenato su un insieme di immagini da singoli punti di vista e comporta la deformazione di rappresentazioni 3D esplicite.

La seconda area di ricerca si concentra sulla predizione di scheletri 3D di persone e robot osservati da una prospettiva esterna come le camere di videosorveglianza. I punti chiave in questo contesto sono integrati nella definizione di scheletro rappresentato come un grafo di punti semantici. Il focus iniziale è sul dominio robotico dove un sistema intelligente che predice scheletri 3D può essere cruciale per la sicurezza in ambienti collaborativi condivisi da persone e robot. Considerando le difficoltà nell'ottenere dataset reali in robotica, enfatizziamo il ruolo della simulazione. Il nostro approccio comporta la raccolta di un dataset sintetico e reale, affrontando il problema della stima della posa 3D attraverso una rappresentazione a due mappe di calore. Esploriamo il divario tra il dominio sintetico e reale utilizzando le mappe di profondità per aumentare l'accuratezza. Introducendo informazioni temporali, il nostro sistema sposa il nuovo paradigma di Pose Nowcasting, in cui predire le pose future rappresenta un problema ausiliario per raffinare la precisione della posa corrente. Passando allo scenario umano, proponiamo un sistema di raffinamento della posa basato sull'analisi di mappe di profondità. Contemporaneamente, la nostra indagine si estende all'interazione uomo-computer, in cui presentiamo un metodo non supervisionato per rilevare e classificare gesti delle mani dinamici usando dati di un sensore che traccia il movimento.

Questa tesi punta a dare un valido contributo all'intersezione tra la 3D Computer Vision e il Deep Learning in vari domini. Dopo uno sguardo sullo stato dell'arte esistente sui problemi di ricostruzione 3D e stima della posa 3D, presentiamo i nostri metodi con una spiegazione tecnica esaustiva supportata da indagini dettagliate dei risultati condotte su dataset ampiamente riconosciuti in letteratura.

Contents

1

Introduction

How can humans perceive the third dimension of the real world? This is one of the biggest questions that Computer Vision has always tried to answer through the analysis of images. Anatomically, humans can see the 3D real world primarily through binocular vision and the brain's ability to process visual information. In particular, the slight separation of the human eyes produces two overlapped viewpoints of the same scene that the brain integrates into a single three-dimensional perception. Indeed, our brain can interpret the slight differences between the two viewpoints, called binocular disparities, that provide information about the depth and distance of the objects in the scene. In addition to this information, the brain processes also some monocular cues. For example, when the objects get far away from the human eye their size and position change and the texture's details become less fine-grained. Moreover, if an object overlaps another one the obscured one is perceived as more distant. In complex scenarios, humans tend to disambiguate objects by observing their motion through time (*e.g.* temporal cues) or changing the point of view of the scene moving around the environment. All of this information is processed in real-time by our brains making us able to navigate the world, judge distances, and interact

with the surroundings exploiting depth, spatial, and temporal relationships.

1

Computer Vision has always tried to emulate human eyes through the analysis of visual data captured by camera sensors. With the recent advancements in the field, the task of 3D perception has gained more and more attention. However, since most of the visual sensors are monocular, a lot of challenges arise due to the lack of multi-view correlation. The ideal setup for a precise perception of the 3D space from monocular sensors is to have a multi-view camera setting and use triangulation reasoning over the viewpoints. However, acquiring a lot of sensors can be expensive, and setting up the multi-view system can be time-consuming because it requires an initial camera calibration step.

Thus, research has started to focus on the more suitable while challenging scenario of single-view images. Thanks to the success and fast development of Deep Learning algorithms, extracting visual cues from images has become more efficient. In particular, Convolutional Neural Networks extract features that can identify salient cues useful for localizing, segmenting, and classifying objects. These features or directly the task-related outputs of such architectures serve as 2D visual cues for finding semantic correspondences that can be leveraged for 3D perception.

1.1 Problem statement

In this thesis, we focus on the role of semantic keypoints as visual cues for 3D perception. Semantic keypoints represent specific landmarks on images that carry a semantic meaning. Usually, they are associated with particular object parts, playing a crucial role in different Deep Learning tasks like object recognition, pose estimation, and semantic segmentation. Different from standard Computer Vision descriptors like SIFT [152] and ORB [208], semantic keypoints are not arbitrary points of interest, but they describe the structure of an object. For example, semantic keypoints can describe salient features of a vehicle like mirrors and lights or joints of a human skeleton like elbows and knees. Semantic keypoints are a compact representation that enables useful reasoning about the 3D understanding of objects for many real-world tasks, such as 3D object reconstruction

or 3D pose estimation. Since they do not require a high computational load, further analysis involving the temporal evolution of these points can introduce additional constraints to improve the 3D perception analysis.

The study carried out in this dissertation regards three main problems of 3D perception described in the following.

3D OBJECT RECONSTRUCTION. The goal of image-based 3D reconstruction is to infer the 3D geometry and structure of objects and scenes from single or multiple images. The task can be crucial for many applications such as robot navigation, scene understanding, 3D modeling, and autonomous driving. In this work, we focus on the reconstruction of objects from single-view images leveraging novel view synthesis and differentiable rendering methods. An important role is played by semantic keypoints that are used to retrieve the correct pose of the vehicle with respect to the camera viewpoint. Moreover, we tackle this problem using only single-view images which requires collections of data with a balanced camera distribution to guarantee the generalization over all possible viewpoints.

3D POSE ESTIMATION. Retrieving a precise 3D pose of an articulated or rigid object is fundamental for augmented or virtual reality applications. In particular, in the Industry 4.0 scenario in which people and robots share the same workplace, having an intelligent surveillance system that predicts the 3D skeleton pose of the agents in the scene enables further analyses to detect potential anomalies or collisions. In this work, we focus more on the robotic scenario since the literature lacks datasets and methods for direct 3D pose estimation from images. We present a synthetic and real robotic dataset and propose a novel double heatmap representation of a 3D skeleton in which a heatmap consists of the localization of semantic keypoints of the 3D skeleton.

After a thorough analysis of the robotic scenario, we extend the pose estimation to the human case. We present an improved framework that leverages the heatmap-based representation adding temporal cues to improve the prediction of the current pose. The extraction of the past motion information of the 3D skeleton keypoints introduces additional information to generate a refined and thus more precise pose.

3D HAND GESTURE RECOGNITION. Another crucial task in 3D perception is enabling computers to interpret and respond to gestures made by users with their hands. The main goal is to create a natural and intuitive interface to interact with digital devices or systems analyzing hand movements. For example, in the automotive scenario, the interaction between the driver and the infotainment system can cause distraction, leading to potential car accidents. In this work, we tackle this problem in an into-the-wild scenario and propose an unsupervised method to detect a gesture using 3D semantic hand keypoints acquired by a hand tracking camera.

1.2 Organization

After an initial overview of the literature about the use of semantic keypoints in different scenarios in Chapter 2, we present the studies carried out in the three topics described in the previous section.

We first focus on the 3D object reconstruction task. In Chapter 3, we propose a Deep Learning pipeline to predict the visual future appearance of an urban scene. Since generating the whole scene in an end-to-end fashion is still a non-trivial task, a two-stage approach leverages interpretable information for each vehicle to generate its synthetic textured version to be placed in the scene according to its trajectory. Following this work, in Chapter 4, we present a multi-task framework that aims to improve car model classification by merging visual features obtained from an image classification network and local features extracted from a keypoint localization network. Finally, in Chapter 5, we propose a multi-category mesh reconstruction architecture that learns to infer the shape and texture of an object from a collection of single-view images deforming 3D triangle meshes initialized as spheres.

Moving to the 3D pose estimation domain, in Chapter 6 we present a novel heatmap-based representation of a 3D skeleton pose, called Semi-Perspective Decoupled Heatmaps (SPDH), that can be learned by adapting efficient deep networks designed for 2D Human Pose Estimation (HPE). In this work, we focus on the robotic scenario presenting the SimBa dataset that contains synthetic and real

sequences of a Baxter robot performing pick-n-place actions. Since the literature lacks datasets and methods for robot pose estimation, a thorough study of this topic is introduced in Chapter 7. Taking inspiration from the HPE domain and leveraging the SPDH representation, we tackle the robot pose estimation task in the Sim2Real scenario exploiting depth data to reduce the domain gap between the synthetic and real domain. It is worth noting that SPDH is a generic representation of semantic keypoints that can be used for any articulated object. In Chapter 8, we formulate the paradigm of Pose Nowcasting which leverages past pose information to improve the prediction of the current pose. A thorough experimental study demonstrates the validity of the approach in both robotic and human scenarios.

The final contribution of this dissertation is dedicated to the hand gesture recognition task in Chapter 9. In this work, we propose an unsupervised approach to detect dynamic hand gestures in a continuous temporal sequence using a Transformer-based architecture that takes as input the 3D hand joint locations together with their speed and acceleration.

To wrap up, Chapter 10 presents the conclusions for each topic along with some comments on potential future works and research directions.

1

2

Semantic Keypoints Survey

Discovering a good set of landmarks to describe objects has been investigated in different contexts in Computer Vision. In this chapter, we present the role of semantic keypoints in two main macro domains presenting the available datasets in the literature. Following the workflow of this dissertation, we start with an analysis of the autonomous driving scenario in Section 2.1 and then move to human-centric understanding in Section 2.2.

2.1 Autonomous driving

Semantic keypoints play a pivotal role in advancing autonomous driving technology by providing a detailed understanding of the 3D environment. These keypoints serve as distinctive landmarks on objects, allowing vehicles to accurately perceive and interpret their surroundings. In the context of autonomous driving, precise identification and tracking of semantic keypoints on objects such as cars, pedestrians, and traffic signs enable robust 3D object understanding. This information is essential for tasks like object recognition, localization, and path planning, contributing to the overall safety and efficiency of autonomous vehicles. By leveraging semantic keypoints, autonomous driving systems can enhance their

Dataset	Image source	3D property	Keypoints	# Images	# Car models
KITTI [64]	Self-driving	3D bbox	No	7481	16
PASCAL3D+ [54]	Natural	complete 3D	Yes (12)	6704	10
ObjectNet3D [259]	Natural	complete 3D	Yes (14)	7345	10
ApolloCar3D [223]	Self-driving	industrial 3D	Yes (66)	5277	79

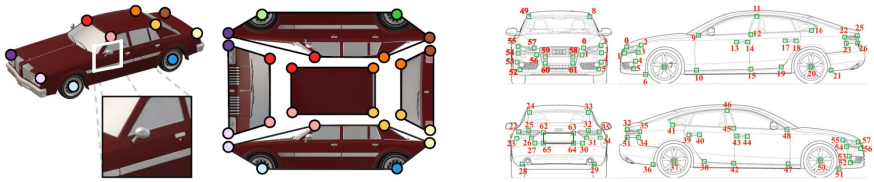
Table 2.1: Comparison between existing dataset with 3D car labels for autonomous driving.

ability to navigate complex scenarios, anticipate potential obstacles, and make informed decisions in real-time, thereby promoting the development of more reliable and intelligent autonomous vehicles for the future.

The prevailing technologies for comprehending the 3D properties of objects primarily depend on high-resolution LiDAR sensors instead of conventional cameras or image sensors. However, there are numerous drawbacks associated with the use of LiDAR, impeding its broader adoption. The most critical issue is that the captured 3D LiDAR points provide, at best, a sparse representation of the scene, particularly in distant and absorbing regions, when viewed from the front. Given the imperative need for a self-driving car to ensure a safe braking distance, the exploration of 3D understanding from a regular camera remains a promising and feasible approach. This approach has gained substantial attention and research interest from the vision community [73, 202].

One of the big challenges in 3D understanding is the lack of fully annotated datasets because the acquisition of large-scale training data is laborious and time-consuming. For example, when it comes to the challenge of comprehending 3D information about cars for autonomous driving, the datasets accessible for this task are notably constrained. In particular, 2D and 3D keypoints annotations can be useful to retrieve the object pose with respect to the camera and also infer its 3D shape using the keypoints as reference points for the object deformation.

The most popular dataset for self-driving is KITTI [64], but it contains only 200 labeled 3D cars in the form of bounding box only, without any detailed 3D shape information. To face the need for massive labeled training data, other datasets such as Pascal3D+ [54] and ObjectNet3D [259] contain more images, but the car instances are mostly isolated in controlled lab settings which are not suit-



(a) Pascal3D+ (b) ApolloCar3D

Figure 2.1: Samples of vehicle keypoints of Pascal3D+ (left, picture taken from [181]) and ApolloCar3D (right, picture taken from [223]).

able for generalizing in a real autonomous driving scenario. More recently, ApolloCar3D [223] dataset was built upon ApolloScape [93] dataset targeting the 3D car understanding research in self-driving scenarios. Indeed, this dataset contains a sufficient amount of cars on the street, large appearance variations, and multiple driving cases (*e.g.* local street, highway, intersections).

In Table 2.1, we present a comparison of the existing dataset in the autonomous driving setting. In this dissertation, we explore the 3D vehicle reconstruction task using the Pascal3D+ dataset for a fair comparison with the state-of-the-art. Since most of the presented architectures leverage vehicle keypoints to infer the 3D synthetic model, a visualization of the semantic meaning of each keypoint for Pascal3D+ and ApolloCar3D is depicted in Figure 2.1.

2.2 Human-centric understanding

Semantic keypoints play a crucial role in advancing the fields of human pose estimation and hand gesture recognition. In human pose estimation, identifying and tracking semantic keypoints on the human body, such as joints and key anatomical points, is essential for accurately understanding and representing body movements. This information is instrumental in applications ranging from fitness tracking to human-computer interaction. Similarly, in hand gesture recognition, semantic keypoints on the hand provide a detailed representation of gestures, enabling machines to efficiently interpret and respond to human communication. The precise localization of keypoints on hands facilitates the recognition of intricate gestures, contributing to applications in sign language interpre-

Dataset	Technology	# Keypoints	Skeleton
Berkeley-MHAD [177]	Marker-based MoCap	43	3D Joints
Human3.6M [97]	Marker-based MoCap	25	3D Joints
TotalCapture [236]	Marker-based MoCap	21	3D Joints
CMU-Panoptic [102]	Markerless MoCap	19	3D Joints
MPI-INF-3DHP [164]	Markerless MoCap	28	3D Joints
MuCo-3DHP [165]	Markerless MoCap	28	3D Joints
JTA [57]	Gaming engine	22	3D Joints
MOTSynth [55]	Gaming engine	22	3D Joints
SURREAL [241]	Scene compositing	24	SMPL
3DPeople [205]	Scene compositing	29	3D Joints
AGORA [190]	Scene compositing	66	SMPL[-X]
BEDLAM [23]	Scene compositing	66	SMPL[-X]

Table 2.2: Overview of 3D Human Pose Estimation datasets available in the literature.

tation, virtual reality interactions, and other human-machine interfaces. Overall, the integration of semantic keypoints significantly enhances the accuracy and robustness of algorithms in these domains, fostering advancements in human-centric technologies and interaction modalities.

HUMAN SKELETON. In the human pose domain, there are different ways to get accurate 3D annotations of the human skeleton. An overview of the most used available datasets in the literature for human pose is presented in Table 2.2.

The first technique is characterized by the use of a marker-based motion capture system. It relies on the attachment of reflective markers to specific anatomical points on a subject's body. These markers reflect light, allowing cameras to track their positions accurately. This method provides precise and detailed data, making it widely used in controlled environments such as studios. However, it requires the subject to wear specialized suits with attached markers, limiting its applicability in naturalistic settings. The most used datasets using this annotation style are Human3.6M [97], Berkeley-MHAD [177], and TotalCapture [236]. Due to the unfeasibility of using this system in realistic scenarios, all of these datasets are single-person with actors performing a set of predefined actions.

To overcome the markers' limitations, markerless motion capture systems have started to leverage Computer Vision algorithms and depth sensors to get the 3D

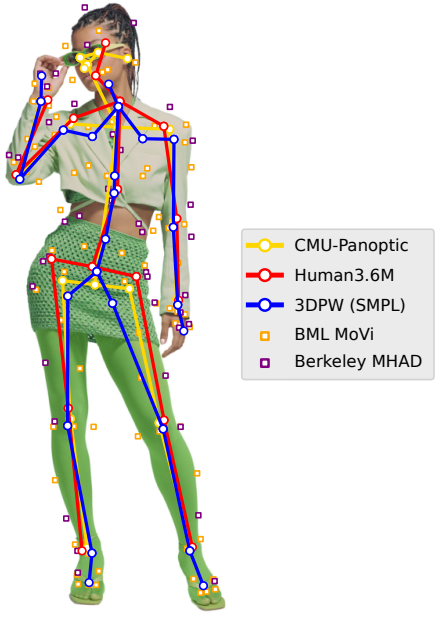


Figure 2.2: Different human pose datasets provide annotations for different sets of body landmarks. Picture taken from [212].

pose annotations. This approach extracts movement information directly from the subject's appearance and body structure, enabling more natural and unobtrusive motion capture. Markerless systems are advantageous for capturing motion in diverse environments, as they do not require subjects to wear specific gear. However, markerless motion capture may face challenges in accurately capturing fine details and may be influenced by factors such as occlusions and lighting conditions. Thanks to the use of Deep Learning techniques for collecting annotations, datasets with multiple subjects can be recorded opening the research to the multi-person pose estimation task. The most used datasets in this field are CMU-Panoptic [102], MPI-INF-3DHP [164], and MuCo-3DHP [165].

Thanks to the recent progress in computer graphics rendering and the introduction of the SMPL [149] parametric model as human shape representation, another automatic way to collect potential infinite datasets with human pose annotations is to leverage synthetic data. In particular, some datasets [57, 55] lever-

a) Camera sensors b) Wearable devices c) Sensory gloves

Figure 2.3: Different modalities of collecting hand gesture information.

age game engines of popular videogames to create realistic scenarios, in which people are spawned performing different actions. However, the photorealism of these datasets is strictly bound to the technology used for rendering. On the other hand, hybrid datasets have been collected with the use of the SMPL model rendered and fitted for each person in real scenarios. Although the first techniques [241, 205] of compositing 3D people and environments lack realism, recent datasets [190, 23] achieve high photorealism thanks to Unreal engine * rendering technology.

Despite the advancement in collecting large-scale datasets, an open problem in human pose estimation is the lack of standardization of the human skeleton. Indeed, each dataset has its own skeleton format as depicted in Figure 2.2.

HAND GESTURE. The fundamental objective in investigating gesture recognition is to develop a system capable of identifying distinct human gestures for communication or command and control functionalities. This involves not only tracking human movements but also interpreting those movements as meaningful commands. Generally, there are two approaches employed to interpret gestures for Human-Computer Interaction applications. The first approach utilizes data gloves, either wearable or in direct contact with the user, while the second approach relies on computer vision, eliminating the necessity for users to wear any sensors.

As depicted in Figure 2.3, collecting datasets with various modalities improves the capability to capture diverse hand movements and gestures. One common modality utilizes depth sensors, such as those found in devices like Microsoft

*https://www.unrealengine.com/

Kinect, Intel RealSense, and Leap Motion cameras. These sensors provide three-dimensional information about hand positions, enabling the creation of datasets that include spatial details crucial for accurate gesture recognition. Another modality involves recording RGB images using standard cameras, capturing the visual appearance of hand gestures. Wearable devices, equipped with inertial sensors like accelerometers and gyroscopes, offer a portable solution for collecting datasets on the go, capturing dynamic hand movements. Electromyography sensors can also be utilized to record muscle activity, providing insights into the subtle nuances of hand gestures. Moreover, glove-based sensors with embedded technology can capture detailed finger movements and hand poses. Combining multiple modalities often results in more comprehensive datasets, contributing to the development of robust and versatile hand gesture recognition systems. The choice of modality depends on factors such as the desired level of detail, portability, and specific requirements of the gesture recognition application.

3

Future Urban Scenes Generation
through Vehicles Synthesis

In the near future, smart interconnected cities will become a reality in various countries worldwide. In this scenario, vehicles – both autonomous and not – will play a fundamental role thanks to key technologies developed to connect them (e.g. 5G) and advanced sensors (e.g. lidars, radars) enabling a deeper understanding of the scene. Explainability is expected to be a mandatory requirement to ensure the safety of all other actors (including pedestrians, cyclists, ...). However, the current approach to autonomous driving-related tasks is still end-to-end, which greatly obscures the learned knowledge. Despite that, recent works [18, 88] have moved from this framework – where raw inputs are transformed into the final outputs/decision – to a more interpretable one, where an intermediate high-level representation is employed. Those representations can be easily understood by human operators and provide an effective parallelism between human and autonomous decision making.

This Chapter is related to the publication "A. Simoni *et al*., Future Urban Scenes Generation Through Vehicles Synthesis, ICPR 2020" [2]. See the list of Publications on page 151 for more details.

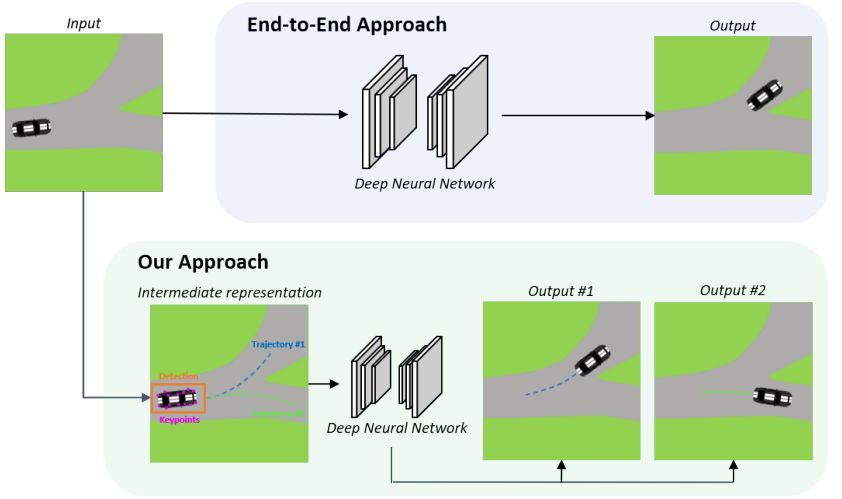


Figure 3.1: The difference between a black box end-to-end method and our approach, which exploits intermediate interpretable information to synthesize each vehicle individually.

In this work, we take a step forward and present a pipeline where the final output is produced by applying a sequence of operations that mimic those of a human operator for a specific task related to autonomous driving. In particular, we focus on generating realistic visual futures for urban scenes where vehicles are the main actors. In more detail, starting from one or multiple RGB frames, the final output is a clip of images where all the actors in the scene move following a plausible path. In doing so, as depicted in Figure 3.1, we rely heavily on the information a user can easily understand, such as bounding boxes, trajectories, and keypoints. Moreover, we wish to easily condition the output on that information; in particular, given a set of trajectories for the same vehicle (either by a state-of-the-art trajectory predictor or a user's input), we would like to generate a set of realistic visual representation of the vehicle following these trajectories. In the following, we focus on vehicles only and leave the analysis of other agents as future work.

It is worth noting how the same task can be tackled as an image-to-image problem, where a deep neural network transforms past frame/s into future ones, as depicted in Figure 3.1. While many end-to-end methods [69, 98, 287, 252] can

be applied to visual scene generation, they all share some intrinsic drawbacks. In particular: *i)* because they start from raw inputs (i.e. RGB images), it is not always clear which is the best way to include user's or geometric constraints; *ii)* despite recent advances in model explanation [225, 216], end-to-end methods are difficult to investigate either before or after critical faults, which is required for critical applications; *iii)* these methods do not focus on the actors but instead transform the entire image, including the static background: this wastes computational time while limiting the maximum resolution that these methods can handle; and *iv)* they can hardly leverage any established state-of-the-art method for additional information, such as vehicle detection or trajectory prediction.

Contrarily, we frame the task as a two-stage pipeline where only vehicles are individually transformed. First, we extract interpretable information from raw RGB frames, including bounding boxes and trajectory estimations. Second, we employ it to produce visual intermediate inputs. Finally, these inputs condition a deep convolutional neural network [53, 181] to generate the final visual appearance of the vehicle in the future. We argue that this approach is closer to the human way of thinking and, as such, better suits a human-vehicle interactions setting. Similarly to what [18, 88] devise for autonomous planning, our method offers an interpretable intermediate representation a user can naturally understand and interact with. Finally, the input resolution does not represent a limit in our proposal. In fact, as only individual vehicles are processed in our pipeline, the input resolution is typically much lower than the full frame one.
To sum up, we:

- provide a novel pipeline that leverages interpretable information to produce a deterministic visual future grounded on those constraints;
- proves that our method is not limited to a uni-modal output, but allows us to generate *"alternative futures"* by acting on the intermediate constraints;
- shows how this approach outperforms end-to-end image-to-image translation solutions both visually and quantitatively.

3.1 Related work

Image-to-Image Translation. Generative Adversarial Networks (GANs) [69, 47, 163, 209, 14] have been widely used to perform image transformations with impressive results. They exploit an *adversarial loss* to constrain generated images to be as similar as possible to the real ones. This supervision signal generates sharper results when compared with standard maximum estimation-based losses, and allows these methods to be employed for image generation and editing tasks associated with computer graphics.

Recent works [167, 98, 287] prove that GANs can help solve conditioned image generation, where the network yields an output image conditioned on an observed input image x and an optional random noise z. This can be applied for example to transform a segmentation map into an image, or a picture taken in day time into one acquired at night time as presented by [98].

Wang et al. [252] propose a framework able to synthesize high-resolution images (*pix2pixHD*), while Zhu et al. [287] define the concept of *cycle consistency loss* to supervise GANs training without the need for coupled data; their goal is to define a function G, which maps from the first domain to the second, and a function F, which performs the opposite. The two domains are bound to be consistent with each other at training time.

To predict multiple frames, several works [151, 244, 201, 122] extend the image-to-image approach by including time. Authors of PredNet [151] propose a network based on Long Short Term Memory (LSTM) [86] combined with convolutional operations to extract features from input images. In [244], an LSTM-based network is trained without any additional information (e.g. optical flow. segmentation masks,...) by leveraging the concept of "network capacity maximization". Qi et al. [201] decompose the task of video prediction into ego and foreground motion and leverage RGBD input for 3D decomposition. Finally, authors from [122] address the issue of low-quality predictions for the distant future by training a network to predict both future and past frames and by enforcing retrospective consistency.

View synthesis. In the last few years, deep generative models have been ap-

plied also to novel view generation, i.e. synthesizing the aspect of an object from different points of view. Many works [155, 280] achieve impressive results on human pose appearance generation. Among them, VUnet [53] is based on a U-Net architecture [207] which combines a GAN mapping an estimated shape y to the target image x with a Variational AutoEncoder (VAE) [114] that is conditioned on the appearance z. This network aims to find the maximum a posteriori $p(x|y,z)$, i.e. the best object synthesis conditioned on both appearance and shape constraints. Yang et al. [269] propose a recurrent encoder-decoder network to learn how to generate different views of the same object from different rotations. The initial object appearance is encoded into a fixed low-dimensional representation, while sequential transformations act on a separate embedding. Finally, the decoder combines both vectors and yields the final prediction.

In the automotive field, Tatarchenko et al. [230] train a CNN to estimate the appearance and the depth map of an object after a viewpoint transformation. The transformation is encoded as azimuth-elevation-radius and is concatenated to the appearance embedding after being forwarded through fully connected layers. By combining multiple predicted depth maps, their approach can generate reconstructed 3D models from a single RGB image. Again, Zhou et al. [284] extract appearance flow information to guide pixel locations after an arbitrary rotation. Their model leverages a spatial transformer [100] to output a grid of translation coefficients. Contrarily, Warp&Learn [181] first extracts 2D semantic patches from the vehicle input image and warps them to the output viewpoint through an affine transformation. Then, an image completion network is employed to seamlessly merge the warped patches and produce the final result. Park et al. [184] draw inspiration from [284] to relocate pixels visible both in the input and target view before using an image completion network based on adversarial training to refine the intermediate result.

3.2 Proposed method

We present here the two fundamental stages of our approach, as illustrated in Figure 3.2. In the first one *(interpretable information extraction)*, we focus on ac-

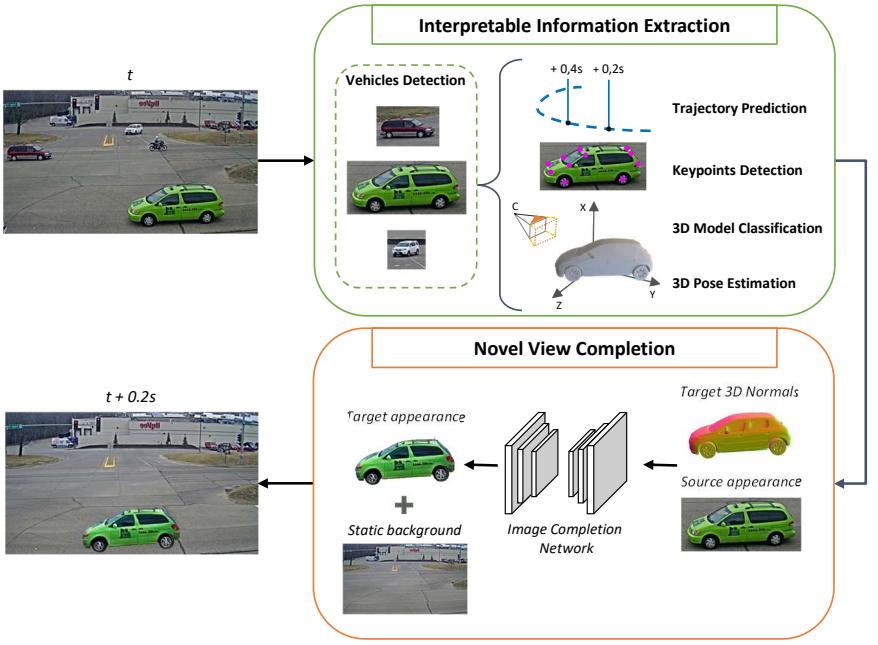


Figure 3.2: Our model pipeline composed of two stages: (i) *interpretable information extraction* for each vehicle (detection & tracking), and (ii) *novel view completion* process exploiting the 3D projected rendering of the object (**target 3D normals**) and its appearance from the cropped image (**source appearance**).

quiring high-level interpretable information for each vehicle in the scene. That information is then exploited by the second stage (*novel view completion*) to generate the final appearance of each vehicle individually.

3.2.1 Interpretable information extraction

During this stage, high-level interpretable information is gathered from raw RGB frames. Vehicles are first detected and their trajectories predicted. However, these trajectories are bound to the 2D plane, which is not sufficient to produce realistic movements (e.g. a car taking a turn). As such, we also detect vehicle 2D keypoints and align them to 3D annotations of Pascal3D+ through a perspective-n-point algorithm, obtaining a roto-translation matrix. This way, we can lift both the vehicle and the trajectory from 2D to 3D, and simulate realistic movements.

Components from this stage are not the focus of this work. Indeed, we are not interested in advancing the research in any of these tasks here, and we make use of pre-trained state-of-the-art methods when possible.

Vehicle Detection. We employ the SSD detection network [147] to detect vehicles in the scene. Starting from the input frame, SSD outputs a set of bounding boxes (one for each detected object) in a single forward pass, along with their class probabilities. We filter the bounding boxes to keep only those associated with a vehicle and use them to crop the visual appearance of each of them.

Trajectory Prediction. We employ TrackletNet [247] as a trajectory predictor; it compares each vehicle tracklet – composed of the detected bounding box and the appearance features – along a time window of 64 consecutive frames. Using a similarity measure between tracklets, a graph is created where vertices under a certain distance threshold represent the same object.

Keypoints Localisation. We adapt a state-of-the-art network for human pose estimation, namely the *Stacked Hourglass* [175], to localize vehicle keypoints. The network is characterized by a tunable number of encoder-decoder stacks. The final decoder outputs a set of planes (one per keypoint) where the maximum value localizes the keypoint location. We changed the final output structure to produce 12 keypoints: (i) four wheels, (ii) four lights, and (iii) four front and back windshield corners.

Pose estimation. We frame vehicle pose estimation as a *perspective-n-point* problem, leveraging correspondences between 2D and 3D keypoints. While the former are the outputs of the previous step, the latter come from annotated 3D vehicle models. We exploit the 10 annotated models included in Pascal3D+, and we train a VGG19-based network [221] to predict the corresponding model given the vehicle crop. We argue these 10 CADs cover the vast majority of urban vehicles, as they have been deemed sufficient to annotate all vehicle images in the Pascal3D+ car set by authors from [260]. Then, we adopt a *Levenberg-Marquardt* iterative optimization strategy [43] to find the best roto-translation parameters by minimizing the reprojection error or *residual* between the 2D original keypoints and the correspondent 3D projections. We follow the stop criteria pre-

sented in [43]. Once the source roto-translation matrix V_s is known, the predicted model can follow the 3D lifted trajectory by applying consecutive transformations defined by the vehicle trajectory – i.e. the roto-translation between consecutive trajectory positions converted from pixel to GPS meter coordinates through a homography matrix computed between the camera and Google Maps viewpoints of the scene. After each transformation, we obtain the target roto-translation matrix V_t.

3.2.2 Novel view completion

Once we know what to move and where to move it, we require a method to condition a reprojected 3D model with the original 2D appearance from the vehicle detection module. Theoretically, any view synthesis approach from Section 3.1 can be used. In practice, a vast majority of them [100, 269, 230, 184] is only able to handle a specific setting known as *"look-at-camera"*, where the vehicle is placed in the origin and the camera z axis points at it. However, in our setting, both V_s and V_t are generic roto-translation matrices. Moreover, some of the methods involve voxel spaces [100], which makes it infeasible to find a correspondence.

Because our focus is on real-world data, we also exclude works that require direct training supervision and can thus only be trained on synthetic data [269]. In fact, as of today, no real-world vehicle dataset can be exploited for supervised novel view synthesis training, as they all lack multiple views for the same vehicle annotated with pose information. This also prevents us from using any method based on [252] for this task. In the following, we thus employ two approaches [53, 181] that are able to handle generic transformations and can be trained in an unsupervised fashion on real-world data.

Giving as input the crop depicting the vehicle x_s observed by a source camera viewpoint V_s, we project the 3D model with the roto-translation outlined by V_t. To enrich the representation, we render a $2.5D$ sketch with normal information. The newly produced output is then pasted into a static background, and the process is repeated for each moving vehicle.

We rely on foreground suppression to generate a static background for the

output clip. We also experimented with inpainting networks [174] but found that results were less realistic by visual inspection due to the presence of several artifacts. We leave further investigation and the extension to moving cameras as a future work.

3.3 Experiments

In this section, we present, both visually and quantitatively, the results of our proposed pipeline and we compare them with those from various end-to-end approaches (referred to as baselines in the following). We also introduce the employed datasets and the metrics of interest, as well as implementation details to ensure experiments reproducibility.

3.3.1 Dataset

CityFlow [229]. It is a multi-target multi-camera tracking and re-identification vehicle dataset, introduced for the 2019 Nvidia AI City Challenge. It comprises more than 3 hours of high-resolution traffic camera videos with more than $200K$ bounding boxes and 600 vehicle identities, split between train and test sets. The dataset also includes homography matrices for bird's eye visualization. Vehicle detection and tracking have been annotated automatically using SSD [147] and TrackletNet [247] as detector and tracker respectively. All baselines have been trained on the train split of this dataset.

Pascal3D+ [260]. It is composed of 4081 training and 1024 testing images, preprocessed to guarantee the vehicle is completely visible. Every image is also classified into one of ten 3D models. Both 3D and 2D keypoints are included. Because 2D keypoints localization is crucial in our pipeline, we extend the Pascal training set by including frames from CarFusion [50]. We train models for our first stage on this dataset to ensure the generalization of our approach when tested on CityFlow.

3.3.2 Metrics

We evaluate all methods using both pixel level and perceptual metrics. The former evaluates the exact spatial correspondence between the predicted and the ground truth target and is very sensitive to 2D transformations (e.g. translations). Contrarily, the latter evaluates the matching between the content of the two images. It is worth noting that we only compare a tight crop around each vehicle for all methods, instead of the full generated image. We argue this choice leads to a better understanding of true performance because it removes a vast portion of the image – with only a static background in it – we are not interested in evaluating.

Pixelwise Metrics. We employ the Mean Squared Error (MSE) as a measure of pixel distance between the target crop x_t and the predicted one x_p as follows:

$$MSE(x_t, x_p) = \|x_t - x_p\|_2^2 \tag{3.1}$$

values are then averaged over to compute the final score.

Perceptual Metrics. We employ the Structural Similarity Index (SSIM) [253] as a measure of the degradation in the image quality due to image data manipulation, defined as:

$$SSIM(x_t, x_p) = \frac{(2\mu_{x_t}\mu_{x_p} + c_1)(2\sigma_{x_t x_p} + c_2)}{(\mu_{x_t}^2 + \mu_{x_p}^2 + c_1)(\sigma_{x_t}^2 + \sigma_{x_p}^2 + c_2)} \tag{3.2}$$

As another measure of content similarity, we measure the Fréchet Inception Distance (FID) [85, 154] computed between activations from the last convolutional layer of an InceptionV3 model pretrained on ImageNet [119]. We compute the FID as follows:

$$FID = \|m_t - m_p\|_2^2 + Tr\big(C_t + C_p - 2\big(C_t C_p\big)^{1/2}\big) \tag{3.3}$$

where m, C refer to the mean and covariance and follow the same notation as above for the target and predicted image.

Finally, we also compute the Inception Score (IS) [209] to measure the gener-

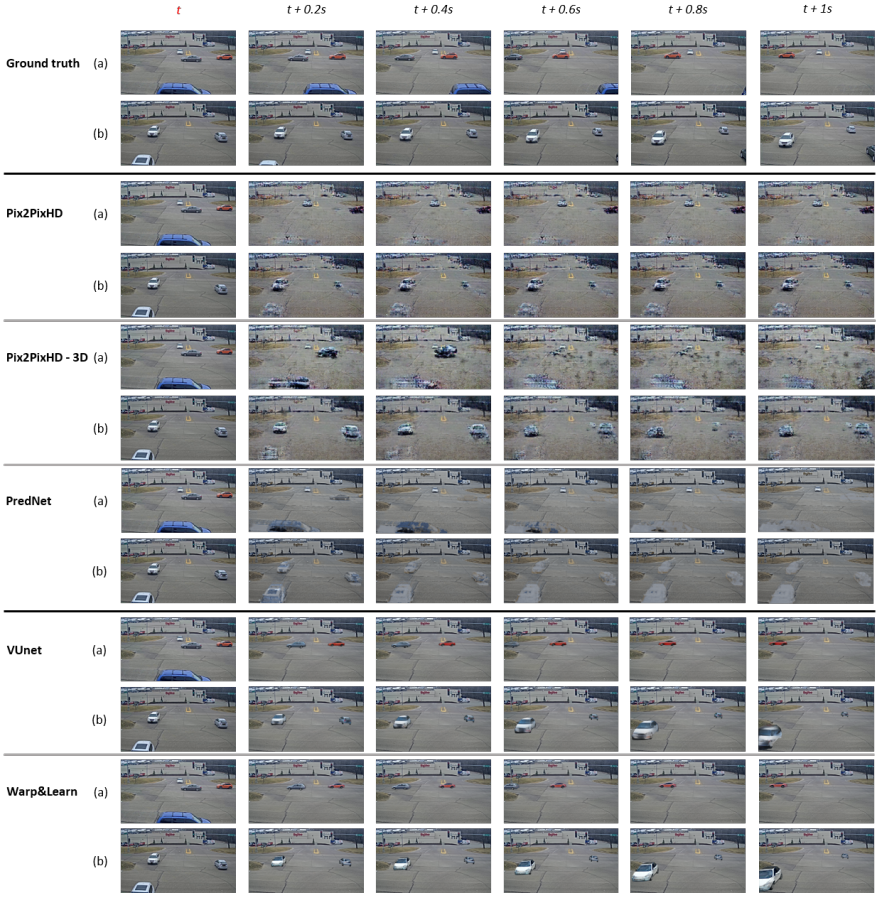


Figure 3.3: Visual results using the methods (Pix2PixHD, Pix2PixHD-3D, PredNet, VUnet, Warp&Learn) on two ground truth video sequences (a) and (b) with different vehicles behavior. Images at time t refer to the ground truth, while images within 1 second in the future represent a method prediction.

ated image variety as:

$$IS(G) = exp(\frac{1}{N}\sum_{i=1}^{N} D_{KL}(p(y \mid x^{(i)}) || \hat{p}(y))) \tag{3.4}$$

where x is an image, N is the total number of samples and $p(\hat{y})$ is an empirical marginal class distribution.

3.3.3 Baselines

Pix2Pix. We adapt Pix2PixHD [252] for future frame prediction. Because it is trained in an end-to-end fashion, we can trivially include any high-level information in the input. Still, we need to condition the output to generate a specific frame (e.g. 0.2 seconds in the future) given the input image. As such, we stack the input with a set of binary maps along the channel dimension. During training, a random frame in the future is selected and the correspondent map is set to 1, while the others are set to 0. It is worth noting that predicting movements given a single input image is an ill-posed task. For this reason, we also include another Pix2Pix baseline – referred to as Pix2Pix-3D in the following – which is time-aware. We provide this baseline with a set of past frames and replace 2D convolutions in the encoder with 3D ones. As such, this baseline version has access to past frames and can therefore exploit temporal information to determine if and how a vehicle is moving. However, this comes with an increase in memory footprint.

PredNet. We also adapt PredNet [151] as a recurrent-based approach to the task. Differently from the previous two, this baseline generates frames in the future via a recurrent structure. However, generated frames have to be forwarded as part of the input to produce frames further in the future, causing errors to propagate and performance to degrade in the long run.

3.3.4 Implementation details

All baselines are trained for 150 epochs on frames from the Cityflow train set, resized to 640x352 pixels. Pix2PixHD-based models employ batch size equal to 4 with an initial learning rate of $2e^{-4}$ and linear decay as defined in [98]. PredNet is trained according to the original paper parameters. As for our pipeline, the Keypoints localization network is trained for 100 epochs employing batch size 10 and a learning rate of $1e^{-3}$ halved every 20 epochs, while VUnet and Warp&Learn models are trained following the policies described respectively in [53, 181]. All models except those for detection and tracking are trained on Pascal3D+ vehicle images resized to 256x256 pixels.

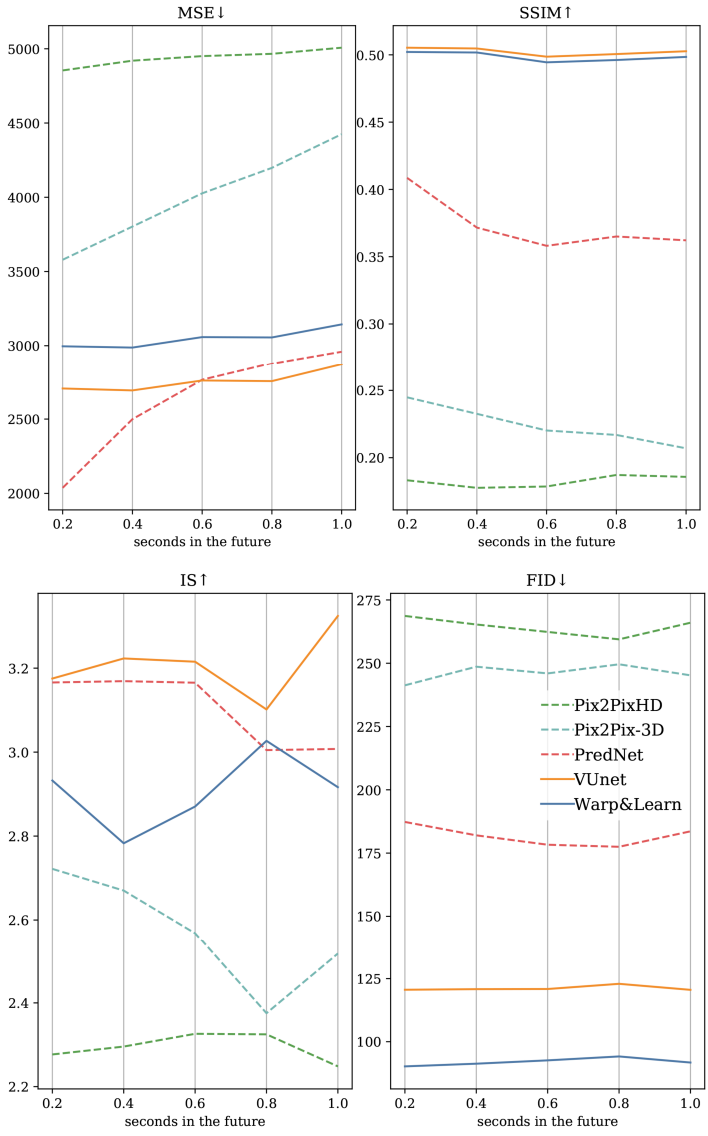


Figure 3.4: Comparison between our approach with two different types of image completion network (solid lines) and multiple baselines (dash lines) using Mean Squared Error (MSE) (lower is better), Structural Similarity Index (SSIM) (higher is better), Inception Score (IS) (higher the better) and Frechet Inception Distance (FID) (lower is better).

Method	*+0.2s*	*+0.4s*	*+0.6s*	*+0.8s*	*+1.0s*
Pix2PixHD [252]	4854	4919	4950	4966	5007
Pix2Pix-3D	3579	3802	4026	4198	4424
PredNet [151]	**2037**	**2499**	2765	2877	2959
Ours (VUnet [53])	2705	2692	**2759**	**2755**	**2870**
Ours (Warp&Learn [181])	2996	2987	3058	3055	3153

Table 3.1: Comparison on the test set using Mean Squared Error (MSE). Each column refers to a future displacement. Lower is better.

Method	*+0.2s*	*+0.4s*	*+0.6s*	*+0.8s*	*+1.0s*
Pix2PixHD [252]	0.18	0.17	0.17	0.18	0.18
Pix2Pix-3D	0.24	0.23	0.22	0.21	0.20
PredNet [151]	0.40	0.37	0.35	0.36	0.36
Ours (VUnet [53])	**0.50**	**0.50**	**0.49**	**0.50**	**0.50**
Ours (Warp&Learn [181])	**0.50**	**0.50**	**0.49**	0.49	0.49

Table 3.2: Comparison on the test set using Structural Similarity Index (SSIM). Each column refers to a future displacement. Higher is better.

Code has been developed using the PyTorch [189] framework and the Open3D library [283] has been employed to manipulate and render the 3D CAD in the scene. Inference is performed on Cityflow test set videos resized to 1280x720 pixels. The code is publicly available *.

3.3.5 Results

Comparisons of the different methods are reported in Tables 3.1, 3.2, 3.3, 3.4 and in Figure 3.4. Our proposed approach outperforms the baselines for all metrics in the long run while scoring second behind PredNet for the first two predictions according to the MSE.

However, it is worth noting how Prednet is not effectively capturing movements, as shown in Figure 3.3. While the first outputs look realistic, performance degrades quickly when predictions are employed as inputs for the LSTM. Our approach proves to be superior for all the metrics that reward the content realism

*https://github.com/alexj94/future_urban_scene_generation

Method	*+0.2s*	*+0.4s*	*+0.6s*	*+0.8s*	*+1.0s*
Pix2PixHD [252]	2.27	2.29	2.32	2.32	2.24
Pix2Pix-3D	2.72	2.67	2.56	2.37	2.51
PredNet [151]	3.16	3.16	3.16	3.00	3.00
Ours (VUnet [53])	**3.17**	**3.22**	**3.21**	**3.10**	**3.32**
Ours (Warp&Learn [181])	2.93	2.78	2.87	3.02	2.91

Table 3.3: Comparison on the test set using Inception Score (IS). Each column refers to a future displacement. Higher is better

Method	*+0.2s*	*+0.4s*	*+0.6s*	*+0.8s*	*+1.0s*
Pix2PixHD [252]	274.2	268.6	265.3	262.3	259.4
Pix2Pix-3D	240.6	241.2	248.6	245.9	249.5
PredNet [151]	197.1	197.2	196.4	193.4	196.3
Our(VUnet [53])	192.8	187.3	182.0	178.3	177.49
Our(Warp&Learn [181])	**90.4**	**90.22**	**91.2**	**92.6**	**94.1**

Table 3.4: Comparison on the test set using Frechet Inception Distance (FID). Each column refers to a future displacement. Lower is better

(i.e. FID, IS, and SSIM) and suffer less performance degradation for long-time predictions. This highlights how focusing on individual vehicles is crucial in visual future scene prediction. Between the two novel view synthesis methods, [53] achieves better performance for 2 (IS, MSE) out of 4 metrics, with comparable results for the SSIM. On the other hand [181] outperforms all other methods by a consistent margin for the FID metric.

Figure 3.3 reports a visual comparison between different methods on two sequences. It can be appreciated how our approach produces higher quality results, both for the static background and the foreground. On the other hand, baseline methods struggle to produce crisp images, often resulting in extremely blurry images. As expected, Pix2PixHD fails to predict vehicle movement and collapses into a static image output. While Pix2Pix-3D partially solves this issue, it still focuses mostly on the background. Finally, PredNet can guess correctly the evolution of the scene, but performance degrades in the long run, with vehicles progressively fading away. It is worth noting that for the first sequence, the

vehicle closer to the camera is not modeled by our method, and thus disappears immediately. This is due to an SSD miss-detection. Even though our final output depends on many modules we argue this is not a weakness in the long run. Indeed, it's trivial to replace a single component with another with better performance, while the same consideration does not hold for end-to-end approaches.

3.3.6 Constrained futures generation

Thanks to its two-stage pipeline our methods can be trivially constrained using high-level interpretable information. Figure 3.5 illustrates an example of this process where the constraint is provided in the form of trajectories. Three futures are generated from the same input frame by providing different trajectories. It can be appreciated how the vehicle closely follows the designated path, which can be easily drawn by a non-expert user. Other constraints that can be provided out of the box include a different CAD model or a different appearance. The same interaction is not well-defined for end-to-end methods.

Figure 3.5: Visual results from our approach for three plausible futures constrained over different trajectories. The detected vehicle follows the indicated trajectory while preserving the original appearance. Best viewed in color.

4

Improving Car Model Classification
through Vehicle Keypoint Localization

The classification of vehicles, and specifically car models, is a crucial task in many real-world applications, especially in the automotive scenario, where controlling and managing traffic can be quite complex. Moreover, since visual traffic surveillance has an important role in computer vision, car classification can be an enabling feature for other tasks like vehicle re-identification or 3D vehicle reconstruction. Despite these considerations, little effort has been made in the computer vision community to improve the accuracy of the existing systems and to propose specialized architectures based on the recent deep learning paradigm. Indeed, from a general point of view, car model classification is a challenging task in the computer vision field, due to the large number of different models produced by many car companies and the large differences in the appearance with unconstrained poses [182]. Therefore, viewpoint-aware analyses and robust classification algorithms are strongly demanded.

This Chapter is related to the publication "A. Simoni *et al.*, Improving Car Model Classification through Vehicle Keypoint Localization, VISAPP 2021" [4]. See the list of Publications on page 151 for more details.

Only recently, some works in the literature have faced the classification problem trying to distinguish between vehicle macro-classes, such as airplanes, cars, and bicycles. For instance, in [9] a multi-task CNN architecture that performs vehicle classification and viewpoint estimation simultaneously has been proposed. In [171] a coarse-to-fine hierarchical representation has been presented to perform object detection, estimate the 3D pose, and predict the sub-category vehicle class. However, we note that learning to discriminate between macro-classes is less challenging than categorizing different specific car models. In [71] the task is addressed through the use of depth images computed from the 3D models. The proposed method not only estimates the vehicle pose but also performs a 3D model retrieval task. Other works [262, 118] are focused on the vehicle and object classification task under partial occlusions. The work most closely related to our system has been proposed by Simoni et al. in [2], where a framework to predict the visual future appearance of an urban scene is described. In this framework, a specific module is committed to classify the car model from RGB images, in order to select a similar 3D model to be placed into the final generated images.

In this paper, we address the specific task of car model classification, in terms of vehicle typology (e.g., pick-up, sedan, race car, and so on). Our starting intuition is that the localization of 2D keypoints on the RGB images can be efficiently exploited to improve the car model classification task. As a training and testing dataset, we exploit the Pascal3D+ [260], one of the few datasets containing a great amount of data annotated with 3D vehicle models, 2D keypoints and pose in terms of 6DoF. In the evaluation procedure, we investigate how the architectures currently available in the literature can deal with the car model classification and the keypoint detection task. Specifically, we investigate the performance of these models applied to specific tasks. Then, we present how to merge visual information and the 2D skeleton data, encoded from an RGB image of the car, proposing a new multi-task framework. We show that exploiting both information through a multi-task system leads to an improvement of the classification task, without degrading the accuracy of the pose detector.

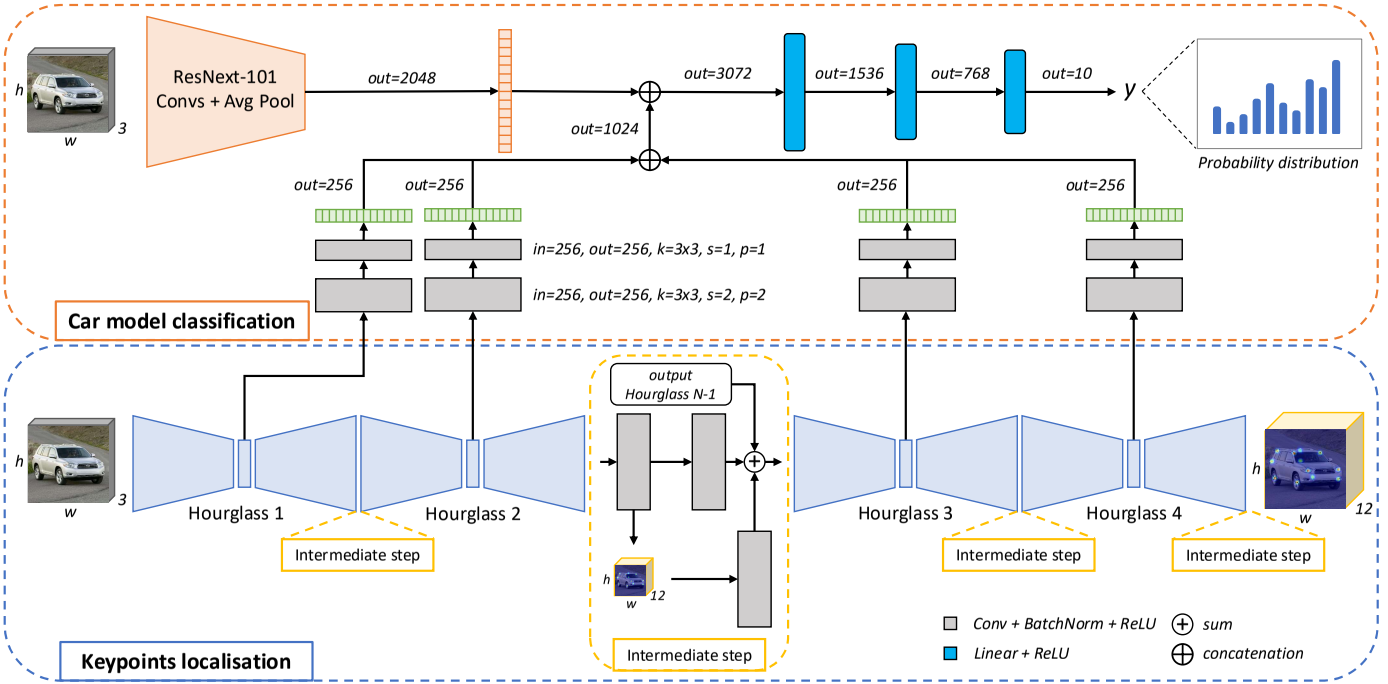


Figure 4.1: Overview of the proposed framework. At the top, the car model classifier is reported, where the visual features – extracted from ResNeXt-101 – are combined with the intermediate feature maps computed by the keypoint localization network – Stacked-Hourglass – shown at the bottom. The input is a single RGB image for both modules, while the output is the keypoint heatmaps and the classified car model.

4.1 Proposed method

In this section, we describe our method that improves the accuracy of the car model classification by leveraging the side task of 2D keypoint localization. The architecture is composed of two sub-networks, each tackling a different task as detailed in the following.

4.1.1 Car model classification

The car model classification task aims to extract visual information from RGB images of vehicles and to classify them into one of the possible classes, each corresponding to a specific 3D vehicle model. Among several classifiers, we choose the ResNeXt-101 network from [264], which is a slightly modified version of the ResNet architecture [82]. The network takes as input an RGB vehicle image of dimension 256×256 and outputs a probability distribution over n possible car model classes. The distinctive aspect of this architecture is the introduction of an aggregated transformation technique that replaces the classical residual blocks with C parallel embedding transformations, where the parameter C is also called *cardinality*. The resulting embeddings can be aggregated with three equivalent operations: i) sum, ii) concatenation or iii) grouped convolutions. This data transformation has proved to obtain higher-level features than the ones obtained from the residual module of ResNet. This statement is also confirmed by better performance on our task, as shown later in Section 4.2. We refer to this section also for a comparison between different visual classifiers.

4.1.2 2D keypoints localization

The second task in hand is the localization of semantic keypoints representative of the vehicle skeleton. Finding 2D object landmarks and having their corresponding 3D model can be useful to estimate and reproduce the object pose in the 3D world using well-known correspondence methods and resolving a *PnP* problem. Similarly to the classification task, many CNNs can solve the 2D keypoint localization task. We choose the architecture presented by [175] between

several alternatives, whose comparison is reported in Section 4.2. This network is called *Stacked-Hourglass* since it is composed of an encoder-decoder structure, called *hourglass*, which is repeated N times composing a stacked architecture. The network takes as input the RGB vehicle image of dimension 256×256 and every hourglass block outputs an intermediate result, which is composed by k Gaussian heatmaps where the maximum value identifies the keypoint location. The presence of multiple outputs through the architecture allows to finely supervise the training by applying the loss to every intermediate output. We tested the network with $N = [2, 4, 8]$ and chose to employ an architecture with $N = 4$ hourglass blocks which obtains the best trade-off between score and performance.

4.1.3 Combined approach

Testing the sole ResNeXt model as a visual classifier proved that car classification is a non-trivial task. We propose to improve the car model prediction by leveraging a multi-task technique that embraces the keypoint localization task too. As depicted in Figure 4.1, we combine pose features extracted by Stacked-Hourglass and visual features extracted by ResNeXt to obtain a more reliable car model classification.

In practice, we leverage the features coming from each hourglass block and analyze them with two convolutional layers with 256 kernels of size 3×3, shared weights, and ReLU activation function. While the first layer has stride and padding equal to 2, the second one has stride and padding 1. Since the Stacked-Hourglass architecture has $N = 4$ hourglass blocks, 4 pose features are obtained. We thus combine them with an aggregation function and concatenate them to the visual features extracted by ResNeXt-101. The fused features are passed through 2 fully connected layers, with 1536 and 768 hidden units and ReLU activation functions. Finally, a linear classifier with n units followed by a softmax layer provides the probability distribution over the 3D car models.

Two different approaches are taken into account for the aggregation of the features obtained by Stacked-Hourglass. One approach consists of summing the

Figure 4.2: Image samples from Pascal3D+ dataset for each car model class.

1D tensors of every encoder and concatenating the summed tensors to the 1D tensor containing the visual features of ResNeXt. Another approach corresponds to first concatenating the 1D tensors of every encoder and then the one extracted

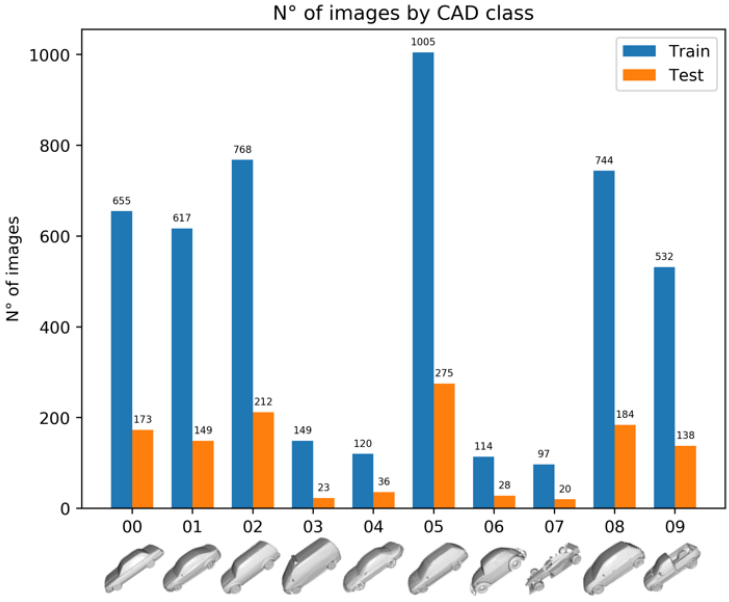


Figure 4.3: Images distribution through train and test set for each vehicle sub-category in Pascal3D+ dataset.

by ResNeXt. In both cases, the resulting features are passed through the 3 fully connected layers that perform the classification task.

To further explain our method, we describe the mathematical formulation of the performed operations. Our multi-task car classification method can be defined as a function

$$\Phi : \mathbb{R}^{w \times h \times c} \rightarrow \mathbb{R}^{n} \tag{4.1}$$

that maps an RGB image I to a probability distribution of n possible car model classes. This function is composed of the two subnetworks presented above and is defined as follows.

The Stacked-Hourglass architecture is a function

$$\mathrm{H} : \mathbb{R}^{w \times h \times c} \rightarrow \mathbb{R}^{w \times h \times k} \tag{4.2}$$

that maps the RGB image I to k heatmaps representing the probability distribu-

tion of each keypoint. The keypoint location is retrieved computing the maximum of each heatmap.
The function H is composed of N encoder-decoder blocks called hourglass. Each encoder E_i of the i-th hourglass outputs a set of features $\mathbf{v}^i_{\text{enc}}$ containing $m = 256$ channels. A series of two convolutional layers are further applied to the output of the encoder block:

$$\Psi : \mathbb{R}^{(w/64)\times(h/64)\times m} \rightarrow \mathbb{R}^m \tag{4.3a}$$

$$\mathbf{u}^i_{\text{pose}} = \Psi(\mathbf{v}^i_{\text{enc}}) \tag{4.3b}$$

where the resulting features have lost their spatial resolution.

Similarly, the parallel ResNeXt architecture, used as a visual feature extractor, can be represented as a function

$$\mathrm{G} : \mathbb{R}^{w\times h\times c} \rightarrow \mathbb{R}^l \tag{4.4a}$$

$$\mathbf{u}_{\text{vis}} = \mathrm{G}(I) \tag{4.4b}$$

that extracts $l = 2048$ visual features from the RGB image I.

The pose features extracted from the hourglass architecture can be aggregated in two ways, as defined previously. Following the *sum* approach, the operation is defined as

$$\dot{\mathbf{u}}_{\text{pose}} = \mathbf{u}^1_{\text{pose}} + \ldots + \mathbf{u}^N_{\text{pose}} \tag{4.5}$$

Alternatively, the *concatenation* approach is defined as

$$\dot{\mathbf{u}}_{\text{pose}} = \mathbf{u}^1_{\text{pose}} \oplus \ldots \oplus \mathbf{u}^N_{\text{pose}} \tag{4.6}$$

In both cases, $N = 4$ is the number of hourglass blocks, and $\oplus$ represents the concatenation operation.

Then, the pose and visual features are combined and given as input to a series of two fully connected layers followed by a linear classifier

$$\mathbf{y} = Y(\dot{\mathbf{u}}_{\text{pose}} \oplus \mathbf{u}_{\text{vis}}) \tag{4.7}$$

obtaining a probability distribution over the n classes of 3D car models.

4.2 Experiments

In this paragraph, we report details about the dataset, the training procedure, and the results in terms of several metrics, execution time, and memory consumption.

4.2.1 Dataset

The Pascal3D+ dataset [260] was presented for the 3D object detection and pose estimation tasks. However, to the best of our knowledge, it is still one of the few datasets that contains RGB images annotated with both 3D car models and 2D keypoints. The dataset is split into 12 main categories from which we select the *car* category. This category is further split into 10 car models (*e.g.* sedan, hatchback, pickup, SUV) and contains 12 keypoints, listed in Table 4.5. As can be seen in Figure 4.2 and 4.3, every image is classified into one of ten 3D models sub-categories and both 3D and 2D keypoints are included. Filtering the images of the *car* class, we obtain a total of 4081 training and 1024 testing images. We process these images in order to guarantee that each vehicle, with its keypoints, is completely visible, *i.e.* contained in the image. All the images are center-cropped and resized to a dimension of 256×256 pixels. Following the dataset structure, we set the number of predicted classes $n = 10$ and the number of predicted heatmaps $k = 12$.

4.2.2 Training

The training of our model can be defined as a *two-step* procedure. Therefore, to extract meaningful pose features for vehicle keypoints, we first train the Stacked-Hourglass model on Pascal3D+ for 100 epochs, using an initial learning rate of $1e^{-3}$ and decreasing it by a factor of 10 every 40 epochs. The network is trained with a Mean Squared Error loss computed between the predicted and the ground truth keypoints heatmaps.

Method	Fusion	Accuracy
[2]	-	65.91%
ResNeXt-101	-	66.96%
Stacked-HG-4 + [2]	*sum*	67.61%
Stacked-HG-4 + [2]	*concat*	69.07%
Ours	*sum*	68.26%
Ours	***concat***	**70.54%**

Table 4.1: Average accuracy results on features fusion classification method.

Network	Layers	Accuracy
VGG16 [221]	*last fc*	65.18%
VGG16 [221]	*all fc*	65.10%
ResNet-18 [82]	*last fc*	59.01%
ResNet-18 [82]	*all*	58.20%
DenseNet-161 [92]	*last fc*	65.02%
ResNeXt-101 [264]	***last fc***	**66.96%**

Table 4.2: Average accuracy results over the 10 car model classes. The second column shows the trained layers while other layers are pre-trained on ImageNet.

The second step starts by freezing both a ResNeXt model, pre-trained on ImageNet [46], and the Stacked-Hourglass model, trained on Pascal3D+; the aim is to train the convolutional layers, that modify the hourglass embedding dimensions, and the final fully connected layers, that take as input the concatenated features, on the classification task. This training lasts for 100 epochs using a fixed learning rate of $1e^{-4}$. In this case, we employ the standard Categorical Cross Entropy loss.

Code has been developed using the PyTorch [189] framework and for each step, we used Adam [113] as optimizer.

4.2.3 Results

Here, we report the results obtained by our multi-task technique and compare them with a baseline, *i.e.* the plain ResNeXt-101 finetuned on the car model classes, and the literature.

As detailed in Section 4.1, the proposed method can combine the pose features encoded by the Stacked-Hourglass network with two different approaches, namely *sum* and *concatenation* (concat). As a baseline, we employ the ResNeXt-101 architecture, finetuned with the Categorical Cross Entropy loss for 100 epochs. To compare with the literature, we report the results obtained by [2], which employ a VGG-19 architecture for the task of car model classification. Moreover, we adapt our proposed architecture, which combines Stacked-Hourglass

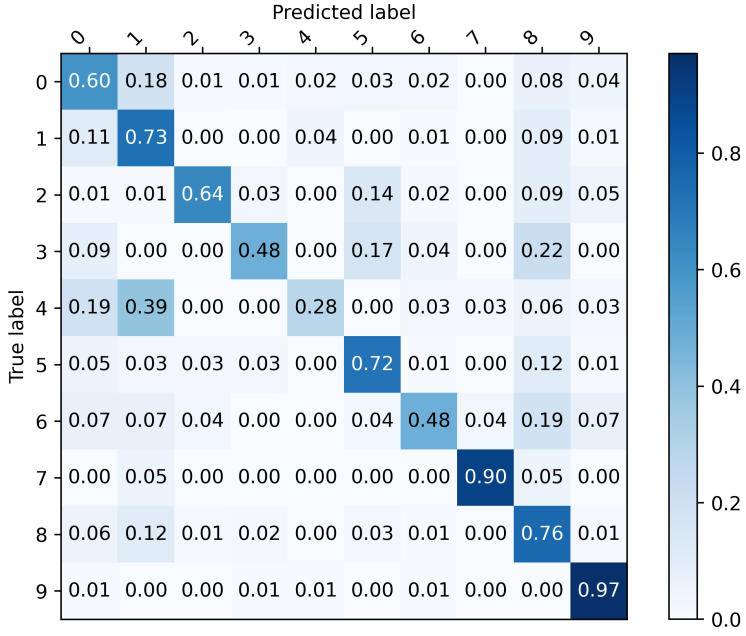


Figure 4.4: Normalized confusion matrix for features concatenation classification method.

and ResNeXt-101, to integrate Stacked-Hourglass, for the keypoint localization, with the method proposed in [2], for the model car classification. In Table 4.1, we show the results in terms of car model classification accuracy. As shown, the proposed method outperforms the baseline and the competitors. Moreover, the combination of the keypoint localization task and the car model classification one steadily improves the results, regardless of the employed classification architecture. Regarding the different combination approaches, the *sum* approach improves the classification score of an absolute $+1.3\%$ with respect to the baseline (ResNext-101). The *concat* approach benefits even more the classification results doubling the accuracy improvement ($+3.6\%$) with respect to the sum approach.

We report in Figure 4.4 the confusion matrix of the proposed method, in the concatenation setting. As it can be seen, most of the classes are recognized with high accuracy, *i.e.* 60% or higher. The sole exceptions are the classes 3, 4, and 6 that are, along with class 7, the less represented classes in both the train and the test set. In particular, even though the class 7 is one of the less represented classes,

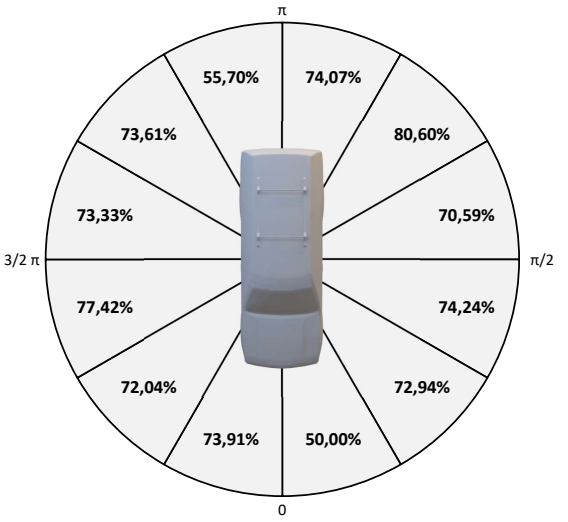


Figure 4.5: Average accuracy results with regard to vehicle viewpoint orientation.

it has a high classification score because it represents sports cars whose image features are more likely to be different from the other classes. It is worth noting that we are aware of the class imbalance problem of the dataset (as depicted in Figure 4.3), but, as we observed in some experiments using an inverse weighting during training (*i.e.* samples from the most common classes are weighted less than samples from the uncommon classes), the results do not have any relevant improvements.

In addition, we show the model accuracy with respect to the azimuth of the vehicle in Figure 4.5. Among values steadily above the 70%, there is a significant drop in accuracy for the angles ranging in $[0, \frac{\pi}{6}]$ and $[\pi, \frac{7}{6}\pi]$. This may be caused by the viewpoint, that may be less informative than the others, by a less represented azimuth range in the training set, or by a more frequent azimuth range for rare or complex cars. This behavior will be the subject of future investigation.

4.2.4 Ablation study

This section covers a quantitative ablation study over several classification and keypoint localization networks, showing the results for both tasks.

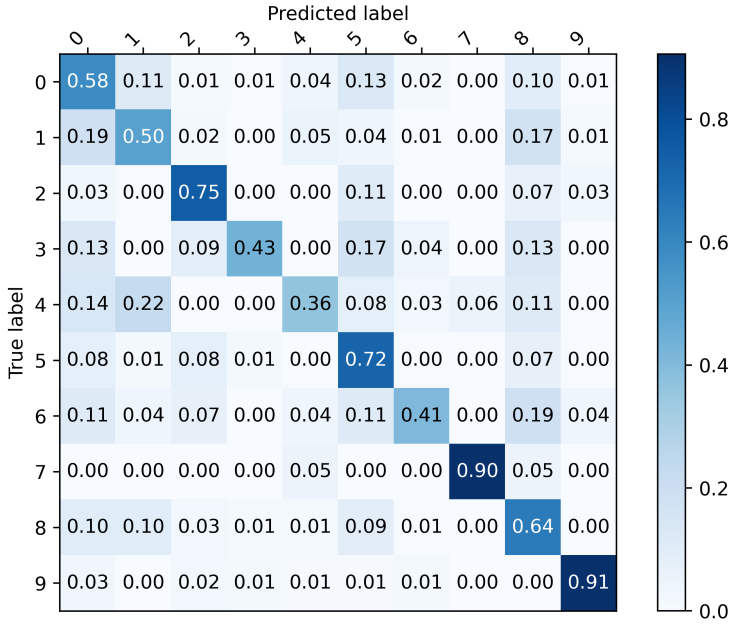


Figure 4.6: Normalized confusion matrix for ResNeXt-101 classification network.

CLASSIFICATION. As shown in Table 4.2, we tested several baselines as visual classifiers. We trained each network for 150 epochs with a fixed learning rate of $1e^{-4}$ and the Adam optimizer. The objective is the Categorical Cross Entropy loss.

It's worth noting that the best results are obtained by ResNeXt-101 with an average accuracy of 66.96%, despite the fact all networks except ResNet-18 are quite close to each other. The results also reveal that networks with a good amount of parameters (see Table 4.4) tend to perform better on the Pascal3D+ dataset than smaller networks like ResNet-18.
Moreover, the good performance of ResNeXt-101 can be clearly observed in Figure 4.6, where the accuracy score is defined for each class. We noted, compared to the other networks, that ResNeXt-101 generates a less sparse confusion matrix, *i.e.* the classifier tends to swap fewer classes with one another.

KEYPOINTS LOCALIZATION. Similarly to the classification task, we tested three architectures, named respectively OpenPose [29], HRNet [248], and Stacked-

Model	PCKh@0.5
[148]	55.7%
[239]	81.3%
OpenPose-ResNet152 [29]	84.87%
OpenPose-DenseNet161 [29]	86.68%
[285]	90.00%
HRNet-W32 [248]	91.63%
HRNet-W48 [248]	92.52%
[193]	93.40%
Stacked-HG-2 [175]	93.41%
Stacked-HG-4 [175]	**94.20%**
Stacked-HG-8 [175]	93.92%

Table 4.3: Average PCK score (PCKh@0.5 with $\alpha = 0.1$) for every keypoint localization baseline (HG = Hourglass).

Model	Parameters (M)	Inference (ms)	VRAM (GB)
VGG19	139.6	6.843	1.239
ResNet-18	11.2	3.947	0.669
DenseNet-161	26.5	36.382	0.995
ResNeXt-101	86.8	33.924	1.223
Stacked-HG-4	13.0	41.323	0.941
OpenPose	29.0	19.909	0.771
HRNet	63.6	60.893	1.103
Ours	106.8	68.555	1.389

Table 4.4: Performance analysis of the proposed method. We report the number of parameters, the inference time, and the amount of video RAM (VRAM) needed to reproduce experimental results. We used an NVidia 1080Ti graphic card.

Hourglass [175], to address the keypoints localization. These architectures are studied as human pose estimation architectures, but we adapt them to our vehicle keypoint estimation task. Each network is trained for 100 epochs using a starting learning rate of $1e^{-3}$ decreased every 40 epochs by a factor of 10 and the Adam optimizer. To evaluate each network we use the PCK metric presented in [13]. In detail, we adopt the PCKh@0.5 with $\alpha = 0.1$, which represents the percentage of keypoints whose predicted location is not further than a threshold from the ground truth. The value 0.5 is a threshold applied on the confidence score of each keypoint heatmap while α is the tunable parameter that controls the area surrounding the correct location where a keypoints should lie to be considered correctly localized.

Although recent architectures like OpenPose and HRNet demonstrate impressive results on human joint prediction, the older Stacked-Hourglass overcomes these competitors in the estimation of the 12 semantic keypoints of the Pascal3D+ vehicles, as shown in Table 4.3 and Table 4.5. It is worth noting that its precision is not only superior on the overall PCK score averaged on all keypoints listed in Table 4.3, but also on the single PCK score for each localized keypoint, as shown in Table 4.5.

Keypoint (∗)	**HG-2**	**HG-4**	**HG-8**	**OP-ResNet**	**OP-DenseNet**	**HRNet-W32**	**HRNet-W48**
lb trunk	93.27	**94.69**	94.18	83.94	86.86	91.72	94.45
lb wheel	92.27	**94.17**	93.09	81.58	84.85	90.26	91.78
lf light	92.85	**93.27**	93.22	86.29	86.34	90.87	91.27
lf wheel	94.41	**95.49**	94.27	86.10	87.70	91.48	89.17
rb trunk	92.59	**92.97**	92.72	83.19	87.00	91.94	92.25
rb wheel	91.50	**91.87**	93.33	79.67	84.35	92.00	91.61
rf light	93.01	**93.79**	93.28	86.47	84.81	89.59	91.54
rf wheel	91.73	**92.71**	92.54	81.52	82.00	89.12	91.16
ul rearwindow	94.67	**95.82**	95.18	86.34	88.06	91.08	93.63
ul windshield	96.00	**96.51**	96.10	89.29	91.37	94.47	95.62
ur rearwindow	93.27	**93.52**	93.91	85.21	87.45	92.39	92.82
ur windshield	95.47	**95.58**	95.17	88.80	89.32	94.59	94.91

Table 4.5: PCK scores (%) for each vehicle keypoint (HG = Hourglass, OP = OpenPose). (∗) lb = left back, lf = left front, rb = right back, rf = right front, ul = upper left, ur = upper right

4.2.5 Performance analysis

We also assess the performance of the tested and the proposed methods in terms of the number of parameters, inference time on a single GPU, and VRAM occupancy on the graphic card. In particular, we compare our approach to all the baselines that perform separately each task. We test them on a workstation with an *Intel Core* i7-7700K and a *Nvidia GeForce GTX* 1080Ti. As illustrated in Table 4.4, our approach has a large number of parameters, but it can perform both keypoints localization and car model classification at once. Taking into account the inference time and the memory consumption, our architecture works largely in real-time speed with low memory requirements while performing two tasks in an end-to-end fashion obtaining better results than using the single networks.

5

Multi-Category Mesh Reconstruction from Image Collections

In recent years, the inference of 3D object shapes from 2D images has shown astonishing progress in the computer vision community. By addressing the task as an inverse graphics problem, *i.e.* considering the 2D image as the rendering of a 3D model, several methods [105, 67, 238] have shown that deep models are capable of restoring the shape, pose, and texture of the portrayed object. While previous methods rely on direct 3D supervision [39, 66, 250, 263] or multiple views [230, 77, 240, 143], recent approaches only require segmentation masks, object keypoints, and coarse camera poses [105, 67, 238]. In the last couple of years, some methods have lessened the dependency on keypoints [238] and even on the camera viewpoint [67]. All these methods share the same underlying approach: a deep model learns a mean 3D shape, called *meanshape*, for the object category during training; then, instance-specific deformation, texture, and camera pose are predicted and applied to the learned meanshape to regress

This Chapter is related to the publication "A. Simoni *et al*., Multi-Category Mesh Reconstruction from Image Collections, 3DV 2021" [7]. See the list of Publications on page 151 for more details.

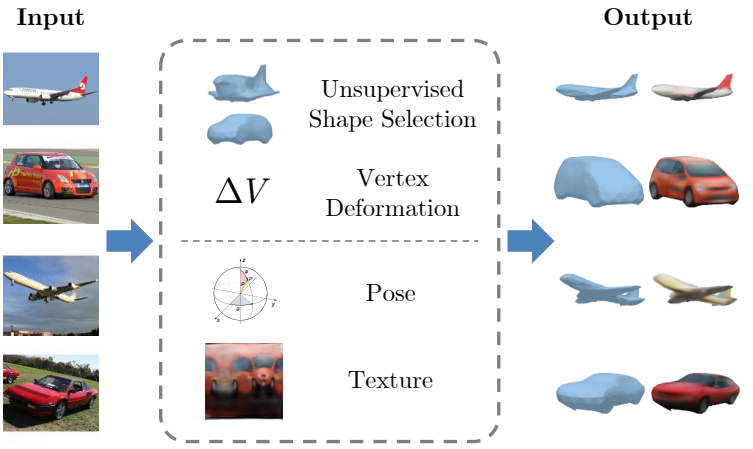


Figure 5.1: Overview of the proposed approach. The method predicts realistic 3D textured shapes of objects of different categories and their 3D pose from a single RGB image.

the 3D model of the object.

A major limitation of existing methods is that they are category-specific: they must be trained and evaluated on image collections of a single object category. This choice has been motivated by the need for category-specific priors in order to recover the 3D shape from 2D images, which is indeed an ill-posed problem unless additional constraints are taken into account. Moreover, most of the approaches [105, 67, 238] initialize the learnable meanshape with a category-specific representative 3D model. To the best of our knowledge, there have been no attempts to extend these methods to scenarios where image collections of multiple categories are available both in training and at inference time.

In this paper, we present a multi-category approach that learns to infer the 3D mesh of an object from a single RGB image. As illustrated in Figure 5.1, the method learns a series of deformable 3D models and predicts a set of instance-specific deformation, pose, and texture based on the input image. Differently from previous approaches, the proposed framework is trained with images of multiple object categories using only foreground masks and rough camera poses as supervision. While rough camera poses could depend on the object category, this is not strictly needed for classes that share semantic keypoints. The method

learns several 3D models in an unsupervised manner, *i.e.* without explicit category supervision, starting from a set of spheres and automatically selects the proper one during inference. Moreover, the instance-specific deformation is inferred by a network that independently predicts the displacement of each vertex of the learned 3D mesh, given the 3D position of the vertex and conditioned on the selected shape and the visual features extracted from the input image. The predicted deformation is naturally smooth and the number of vertices and triangles of the 3D mesh can be dynamically changed during training, with either a global or a local subdivision.

To showcase the quality of the proposed method, we present a variety of experiments in different settings on two datasets, namely Pascal3D+ [260] and CUB [246], and run several ablation studies. For instance, we test the method on multiple object categories related to the automotive environment of the Pascal3D+ dataset (*i.e.* bicycle, bus, car, and motorbike) and on the entire set of Pascal3D+ categories. Qualitative and quantitative results confirm the quality of the proposed approach and show that the model is capable of learning category-specific shape priors without direct supervision.

To sum up, our main contributions are as follows:

- We present an approach that recovers the 3D shape, pose, and texture of an object from a 2D image. The method is trained using image collections with foreground masks and coarse camera poses, but no explicit category nor 3D supervision.

- Our multi-category framework learns to distinguish between different object categories and produces meaningful meanshapes starting from a set of 3D spheres.

- Our approach predicts single vertex deformations, resulting in smooth 3D surfaces and enabling the dynamic subdivision of the learned meshes.

Approach	Supervision			w/o 3D	Multi	Dynamic
	Keypoint	Camera	Mask	Template	category	subdiv.
CSDM [107]	✖	✖	✖			
CMR [105]	✖	✖	✖			
VPL [108]		✖	✖			
CSM [121]			✖			
A-CSM [120]			✖			
IMR [238]			✖			
U-CMR [67]			✖			
UMR [142]			✖	✔		
Ours		✖	✖	✔	✔	✔

Table 5.1: Comparison between available approaches based on training supervision, independence from offline-computed 3D templates, multi-category and dynamic subdivision support.

5.1 Related work

In the last decade, many methods have been proposed to tackle the task of 3D reconstruction from a single image. However, the majority of these methods require supervisory signals which are hard to obtain in the real world and in the wild, such as 3D models [39, 66, 288, 156, 250, 206, 263, 15, 135] or multi-view image collections [230, 266, 77, 257, 240, 237, 96, 143].

Recently, thanks to the development of several differentiable renderers [150, 110, 180, 145, 34], a handful of methods [107, 84, 105] have shown that the task can be addressed as an inverse graphics problem using fewer supervisory signals, such as 2D segmentation masks and object keypoints. Following methods have even relaxed these constraints, training without keypoint supervision [34, 109, 108] or known camera poses [238, 67, 142]. However, these methods require image collections of a single object category and some of them need a meaningful initialization of a category-specific shape. Differently, our method is capable of jointly learning shapes of several object categories using only foreground masks and coarse camera poses as supervision.

Another group of works that exploit differentiable renderers address the reconstruction task as a canonical surface mapping [121, 120] or a surface estimation task [133]. These methods usually require 3D supervision [133] or category-

specific shape templates [121, 120]. In this paper, we focus on the 3D mesh reconstruction from single-view images without any category-specific template.

Recently, Li *et al*. [141] proposed a video-based method and the use of multiple meanshapes (referred to as "base shapes") that are combined to produce a single deformable shape. This is the most similar work to our approach, but it has some key differences. Firstly, the meanshapes are defined offline and set before training, thus they are not learned. Then, they are introduced for one single dataset to exclusively cover the intra-class variation. On the contrary, our meanshapes are learned during training without category supervision and our approach can deal with several object categories and their intra- and inter-class variations.

A comparative study of literature methods is proposed in Table 5.1, highlighting the differences in terms of training supervision, independence from offline-computed 3D templates, multi-category and dynamic subdivision support. As shown, the proposed method still relies on camera supervision but introduces some unique features. Indeed, it learns category-specific shape priors in an unsupervised manner and instance-specific deformations from multi-category image collections. Moreover, the method exploits multiple steps of subdivision during the training process.

5.2 Proposed method

In this section, we present the components of our method, from the input image to the reconstructed 3D textured mesh. The architecture is illustrated in Figure 5.2. Moreover, a comparative study of literature methods is proposed in Table 5.1, highlighting the differences in terms of training supervision, independence from offline-computed 3D templates, multi-category and dynamic subdivision support. As shown, the proposed method still relies on camera supervision but introduces some unique features. Indeed, it learns category-specific shape priors in an unsupervised manner and instance-specific deformations from multi-category image collections. Moreover, the method exploits multiple steps of subdivision during the training process.

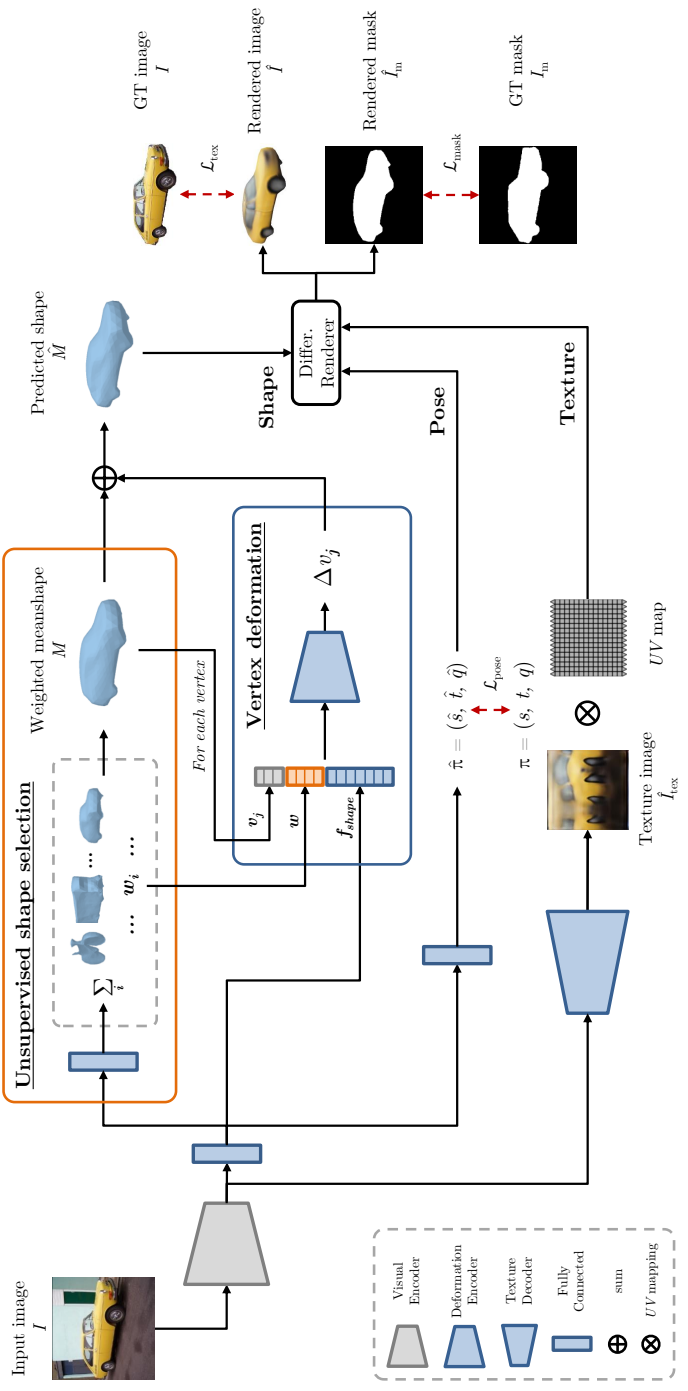


Figure 5.2: Overview of the proposed method. The *unsupervised shape selection* module predicts the category meanshape while the *vertex deformation* module infers the instance-specific deformation, obtaining the predicted shape. In parallel, pose and texture are estimated and then provided, along with the shape, to a differentiable renderer that renders the textured image.

5.2.1 Preliminary definitions

Shape. As other approaches in the literature [105, 108, 67, 142], we use the triangle mesh as 3D shape representation, which is defined by a set of vertices $V = \{v_j = [x, y, z],\ j = [1, \ldots, k]\}$ and a set of triangle faces F. The faces determine the connectivity between vertices but are also related to the texture mapping.

In our approach, we leverage this connectivity property and change dynamically, during training, the number of vertices and faces of the 3D shape aiming for smoothness and better textures. We refer to this technique as *dynamic mesh subdivision*.

Texture. The triangle mesh texture is represented by a texture image I_{tex} and a color map UV which maps between the 2D coordinate space of I_{tex} and the 3D coordinate space of the mesh surface of a sphere. Thus, the UV mapping is defined by spherical coordinates.

Pose. We use a weak-perspective camera projection to define the 3D object pose, as commonly done in the literature. This geometric projection is a simplified version of the standard perspective projection. Thus, the object pose is parametrized by a scale factor $s \in \mathbb{R}$, a translation $t = (x, y)$ in image coordinates, and a quaternion rotation q obtained by a rotation matrix computed from Euler angles (*i.e.* azimuth, elevation, and roll). We define $\pi = (s, t, q)$ as the weak-perspective camera projection.

Rendering. In order to render a 3D shape with its texture, we rely on the differentiable renderer Soft Rasterizer [145]. It takes a triangle mesh, a texture image I_{tex} and an object pose π as input and outputs the rendering of the textured object as the RGB image $\hat{I}$ and the foreground mask $\hat{I}_{\text{m}}$.

5.2.2 Multi-category mesh reconstruction

In this paper, we aim to recover the 3D shape of an object from a single image. In the literature, this task has been often addressed by splitting it into two parts: on the one hand, the definition or learning of a category-specific base shape, named

meanshape; on the other hand, the prediction of an instance-specific deformation of the learned shape. Differently from the majority of previous works (see Table 5.1), we do not need a category-specific initialization of these shapes and propose the joint and unsupervised training of shapes for multiple object categories. In the following, we provide the details of our approach.

FEATURE EXTRACTION. Given an RGB image $I \in \mathbb{R}^{3\times w\times h}$ as input, the first step of our framework is the extraction of visual features with a convolutional encoder (*e.g.* ResNet-18 [82] in our experiments). These features are defined as f_{tex} and used to estimate the 3D object texture with a specific decoder. The same features are flattened and mapped into a compact version f_{shape}, used to recover the shape and its viewpoint.

UNSUPERVISED SHAPE SELECTION. In contrast to current literature approaches, which are category-specific, we propose an unsupervised technique that automatically learns to distinguish between different object categories. Instead of a single meanshape, we define a set of N deformable spheres and use a network to select the instance-specific meanshape according to the input image. The features f_{shape} are passed through a set of fully connected layers and a softmax function. Then, the resulting scores are used to compute a weighted sum of the mesh vertices and obtain a single mesh, approximating the argmax function over the N meanshapes. While the meanshapes are initially defined as spheres, they are updated during the training process and progressively specialize in different object categories. Formally, let $\mathcal{M}_i = (V_i, F)$ be one of the N meanshapes and $\mathbf{w} = [w_1, \ldots, w_N]$ be the output of the network. The weighted meanshape $\mathcal{M}$ is computed as:

$$\mathcal{M} = (V, F) = \left(\sum_{i=1}^{N} w_i V_i\,,\; F\right) \tag{5.1}$$

This mesh $\mathcal{M}$ will be deformed according to the object depicted in the input image I, as explained in the following.

VERTEX DEFORMATION. Inspired by previous works [74, 187], we develop a lightweight network which deforms the meanshape $\mathcal{M}$ taking as input the features f_{shape} and the 3D coordinates of a single meanshape vertex v_j at a time. We

further condition the output on the selected meanshape giving the weighting scores $\mathbf{w}$ produced by the previous module as additional input. In this way, we enforce the connection between the weighted meanshape M and the predicted deformation. The module outputs a 3D displacement or deformation Δv_j of the vertex v_j in the 3D space. This approach makes the architecture independent of the number of vertices of the mesh, enabling us to predict the deformation of meshes of variable sizes. Given a set of deformations ΔV for each vertex of a meanshape M, the predicted shape can be defined as $\hat{M} = M + \Delta V = (V + \Delta V, F)$.

Dynamic mesh subdivision. To improve the smoothness of the predicted deformed shape, we apply during training a *dynamic subdivision* of the triangle mesh. In particular, we use a global subdivision that divides each triangle of a mesh M into 4 equal parts. Other methods that make use of mesh subdivision (*e.g.* [250, 135]) need architectural changes that drastically increase the required memory and inference time. On the contrary, our method is not heavily affected by the mesh subdivision operation and does not require any architectural changes, thanks to the per-vertex prediction of the deformation network.

3D pose regression. We further predict the object viewpoint with a supervised regression technique using two fully connected layers which take as input the features f_{shape} and output a 3D weak-perspective pose $\hat{\pi} = (\hat{s}, \hat{t}, \hat{q})$.

Texture prediction. In order to produce a realistic 3D shape, we finally predict the texture that the differentiable renderer applies to the predicted deformed mesh $\hat{M}$. Similar to the work of Goel *et al.* [67], we use a convolutional decoder that takes as input the visual features f_{tex}, preserving the spatiality, and directly outputs an RGB image $\hat{I}_{\text{tex}}$. The texture is mapped onto the UV space of the shape, which is homeomorphic to a sphere so that it can be exploited by the renderer to produce the final image $\hat{I}$.

5.2.3 Losses and priors

The shape prediction is supervised only by two supervisory signals, *i.e.* the binary object mask I_{m} and the 3D camera pose π.

We first handle the shape deformation applying a mask loss $\mathcal{L}_{\text{mask}} = ||I_{\text{m}} -$

$\hat{I}_{\text{m}}||_2^2$ where $\hat{I}_{\text{m}}$ is the binary object mask produced by the renderer using the ground truth pose π. In addition to this loss, we also use some priors to maintain a certain smoothness of the object's surface. The first prior is a laplacian smoothing loss $\mathcal{L}_{\text{smooth}} = ||LV||_2$ where the Laplace-Beltrami operator [224] minimizes the mean curvature; we apply this smoothing prior both to the predicted deformations ΔV and the vertices of the deformed shape $\hat{M}$. The second prior is a regularization term $\mathcal{L}_{\text{def}} = ||\Delta V||_2$ which prevents the network from learning large deformations and helps to produce more realistic meanshapes. Our final shape loss is represented by:

$$\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{smooth}} + \mathcal{L}_{\text{def}} \tag{5.2}$$

For the pose regression module, we use a loss defined as:

$$\mathcal{L}_{\text{pose}} = ||\hat{s} - s||_2^2 + ||\hat{t} - t||_2^2 + (1 - |q * (\hat{q} \odot -\hat{q})|) \tag{5.3}$$

where the first two terms consist of the mean squared error for scale and translation and the last term is the geodesic quaternion loss. The operator $*$ is the Hamilton product and $\odot$ is the concatenation between the original quaternion and its version rotated by 360 degrees, representing the same rotation. Moreover, following the approach proposed by Pavllo *et al*. [196], we further regularize the quaternion prediction with the penalty term $\mathcal{L}_{\text{pose\_reg}} = w^2 + x^2 + y^2 + z^2 - 1^2$ that forces the quaternion to have unit length and thus representing a valid rotation. The overall camera loss is set as:

$$\mathcal{L}_{\text{cam}} = \mathcal{L}_{\text{pose}} + \mathcal{L}_{\text{pose\_reg}} \tag{5.4}$$

To produce realistic colors and details for the object texture, we convert the rendered RGB image and the masked input image to the LAB color space and apply the following losses: a color loss $\mathcal{L}_{\text{color}} = ||\hat{I}_{ab} - (I \cdot I_{\text{m}})_{ab}||_2^2$ on the AB channels for more faithful texture details and a style loss $\mathcal{L}_{\text{style}} = ||\hat{I}_L - (I \cdot I_{\text{m}})_L||_2^2$ on the L channel for sharper high-frequency details. Moreover, we apply a perceptual loss $\mathcal{L}_{\text{percept}} = F_{\text{dist}}(\hat{I}, I \cdot I_{\text{m}})$ where F_{dist} is the metric defined by Zhang

et al. [276] using a VGG16 backbone as feature extractor. The final texture loss is defined by:

$$\mathcal{L}_{\text{tex}} = \mathcal{L}_{\text{color}} + \mathcal{L}_{\text{style}} + \mathcal{L}_{\text{percept}} \tag{5.5}$$

The overall objective applied during training is a weighted sum of the shape, camera, and texture losses, obtaining balanced learning of the different network modules.

5.3 Experiments

In this section, we first present the employed datasets and the experimental setting. Then, we present quantitative and qualitative evaluations of our approach in comparison with literature methods. Finally, we report an ablation study on the key elements of the proposed approach.

5.3.1 Experimental setup

Two common datasets, namely Pascal3D+ [260] and CUB-200-2011 [246], have been used to evaluate the proposed approach on a diverse set of object categories and, at the same time, to obtain a comparison with the current state-of-the-art methods. As done in previous works [105, 67], 2D image collections, foreground masks, and coarse camera/object poses – manually or automatically annotated – are used for training. We do not take advantage of annotated keypoint positions or coarse 3D model correspondences.

Pascal3D+. The Pascal3D+ dataset [260] contains images of 12 object classes, from both PASCAL VOC [54, 79] and ImageNet [46], associated with 3D models of each category and coarse viewpoints [228, 181]. Manually-annotated foreground masks are available for the PASCAL VOC subset, while an off-the-shelf segmentation algorithm [81] is used for the other subset, as done in previous works [105, 67, 238]. We evaluate the system using the same train/test split and categories, *i.e*. *airplane* and *car*, of the competitors. In addition, we use the segmentation masks obtained by the novel PointRend architecture [115] and evaluate our model on a set of automotive classes, *i.e*. *bicycle, bus, car, motorbike*, and

on the entire set of 12 classes in the ablation study.

CUB. We also use the images of 200 bird species and their foreground masks provided in CUB-200-2011 [246] and the camera poses computed by Kanazawa *et al*. [105], as done in previous works [105, 67, 238]. The dataset also contains 312 binary attribute labels divided into several categories.

NETWORK ARCHITECTURE. Our model is composed of 5 modules: (i) a visual encoder, defined as a pre-trained ResNet-18, with an additional convolutional layer, (ii) an unsupervised shape selection module composed of two fully connected layers and a softmax activation function, (iii) a vertex deformation network with four 512-dimensional fully connected layers with random dropout and a tanh activation function, (iv) a camera pose regressor with two fully connected layers and random dropout, and (v) a texture decoder that follows the implementation of the SPADE architecture [188] with 6 upsampling steps.

TRAINING PROCEDURE. We train our network on both datasets for 500 epochs with an initial learning rate of $1e^{-4}$. The meanshapes are initialized as icospheres with 162 vertices and 320 faces (corresponding to the subdivision level 3). After 350 epochs, we apply the dynamic subdivision to the 3D shapes (roughly obtaining the subdivision level 4) and reduce the learning rate to $1e^{-5}$. Our final 3D shape has roughly the same number of vertices and faces as the competitor approaches [105, 67] which use a deformable template with subdivision level fixed to 4.

All input images are cropped using the object bounding box and resized to a dimension of 256×256 and the model predicts a texture image of the same size. As data augmentation, we apply standard random jittering on the bounding box size and location and random horizontal image flipping. In addition, instead of forcing the shape to be symmetric with post-processing steps (as done in other works, *e.g*. [105, 67, 142]), we force the network to predict symmetric shapes with the following approach, similar to what is done in the work of Wu *et al*. [258]. During training, the predicted shape (*i.e*. its pose) is randomly rotated by 180 degrees around the vertical axis and compared with the flipped versions of the ground truth image and mask. In this way, the network is forced to predict

Approach	Training	Airplane	Car	Avg
CSDM [107]	indep.	0.400	0.600	0.500
DRC [240]	indep.	0.420	0.670	0.545
CMR [105]	indep.	**0.460**	0.640	0.550
IMR [238]	indep.	0.440	0.660	0.550
U-CMR [67]	indep.	-	0.646	-
Ours (N meanshapes)	indep.	**0.460**	**0.684**	**0.572**
Ours (2 meanshapes)	**joint**	**0.448**	**0.686**	**0.567**

Table 5.2: 3D IoU on Pascal3D+ dataset [260]. Our method is trained on airplanes and cars independently using N meanshapes (one for each subclass) or on airplanes and cars jointly with 2 meanshapes.

symmetric shapes (along the vertical axis) and thus to consistently minimize the losses without computational overhead.

We use a batch size of 16 and Adam [113] as optimizer with a momentum of 0.9. The code is developed using the PyTorch [189] framework.

5.3.2 Results

In this section, we provide a thorough comparison between the proposed method and the competitors on the two previously presented datasets, Pascal3D+ and CUB.

Pascal3D+. We show the results of our method compared to the state of the art on the Pascal3D+ dataset in Table 5.2, using the 3D IoU metric as proposed by Tulsiani *et al*. [240]. We present two different versions of our method. Firstly, we employ the same approach used by competitors: train a different model for each class of Pascal3D+ (experiments marked as "independent training"). In this case, we set the number of meanshapes equal to the number of subclasses of Pascal3D+, *i.e*. $N = 8$ for the airplane class, $N = 10$ for the car class. As reported in the second-to-last row of Table 5.2, our method can leverage the use of multiple meanshapes and the dynamic subdivision obtaining state-of-the-art results on this dataset. In addition, we jointly train our method on both the airplane and

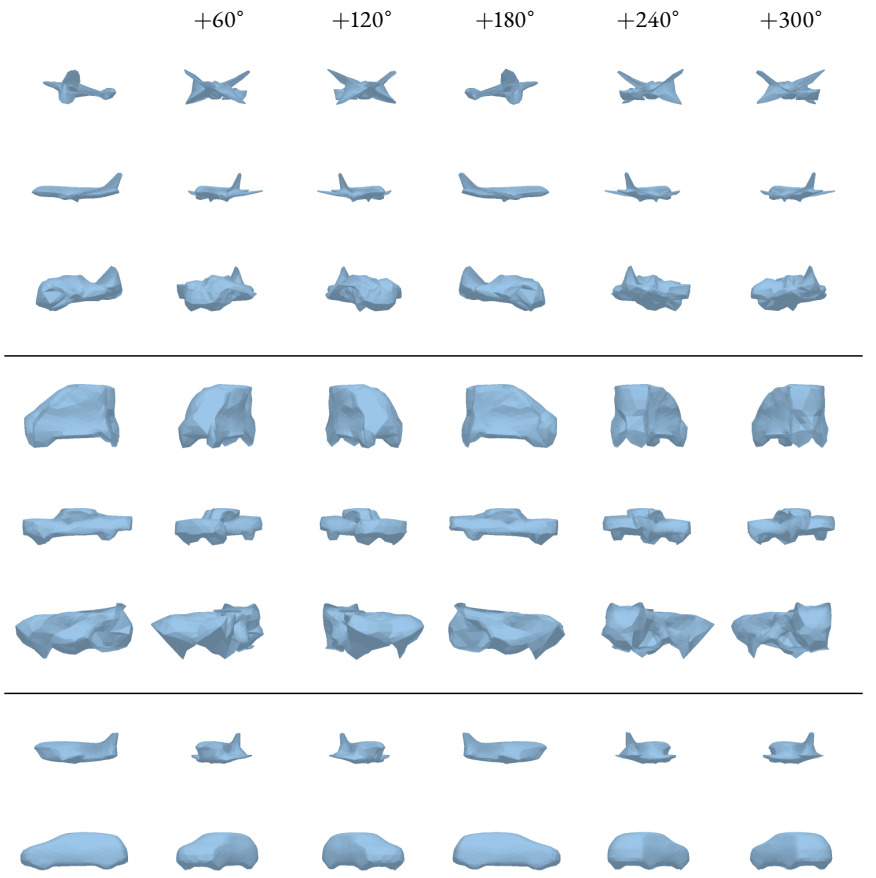


Figure 5.3: Some of the meanshapes learned during training on Pascal3D+. First group: airplane class (8 meanshapes); second group: car class (10 meanshapes); third group: airplane and car classes (2 meanshapes).

the car classes using 2 meanshapes and letting the network distinguish between the two classes. Even in this more complex scenario, we obtain comparable or state-of-the-art scores on both classes (see last row of Table 5.2). The learned meanshapes for these three experiments, *i.e.* training on airplanes, on cars, and on airplanes and cars jointly, are shown in Figure 5.3. We observe that the set of meanshapes on the single classes contains both recognizable and less explainable shapes (Figure 5.3, top and middle). On the other hand, the two meanshapes

Approach	Mask IoU ↑		Texture metrics		
	Pred cam	GT cam	SSIM ↑	L1 ↓	FID ↓
CMR [105]	**0.706**	0.734	**0.718**	**0.063**	290.32
DIB-R [34]	-	**0.757**	-	-	-
U-CMR [67]	0.637	-	0.689	0.077	**190.35**
Ours (1 meanshape)	0.658	0.721	0.717	0.064	227.24
Ours (14 meanshapes)	0.642	0.723	0.715	0.065	231.95

Table 5.3: Mask IoU and texture metrics on CUB dataset [246]. Our method is trained using 1 or 14 meanshapes.

Training classes	Number of meanshapes	3D IoU ↑	Mask IoU ↑		Texture metrics		
			Pred cam	GT cam	SSIM ↑	L1 ↓	FID ↓
airplane, car	1	0.532	0.592	0.689	0.736	0.066	365.01
airplane, car	2	**0.552**	**0.671**	**0.702**	**0.737**	**0.062**	**344.80**
bicycle, bus, car, motorbike	1	0.517	0.665	0.751	0.601	0.100	390.41
bicycle, bus, car, motorbike	4	**0.543**	**0.711**	**0.759**	**0.607**	**0.094**	**380.15**
12 Pascal3D+ classes	1	0.409	0.602	0.670	0.660	0.088	357.51
12 Pascal3D+ classes	12	**0.425**	**0.620**	**0.685**	**0.665**	**0.086**	**345.90**

Table 5.4: Ablation study comparing the usage of several meanshapes (our proposal) against a single meanshape (as a baseline) on Pascal3D+ dataset [260] using segmentation masks obtained with PointRend [115].

learned in an unsupervised manner using images of airplanes and cars correspond to these two classes (Figure 5.3, bottom). We show qualitative results of the joint setting on airplanes and cars in Figure 5.6 (second block).

CUB. We also evaluate our method on the CUB dataset. Results in terms of foreground mask IoU and texture metrics (SSIM [253], L1, and FID [85, 154]) are reported in Table 5.3. Differently from the previous case, the CUB dataset does not have a clear subdivision in classes and literature approaches have only been tested on the whole dataset. Thus, we test our method in two different settings. On the one hand, we evaluate the use of a single meanshape (as done by competitors). On the other hand, we test our method initializing N deformable meanshapes, as done in previous experiments. We empirically set $N = 14$, which is equal to the number of different values of the annotated categorical attribute

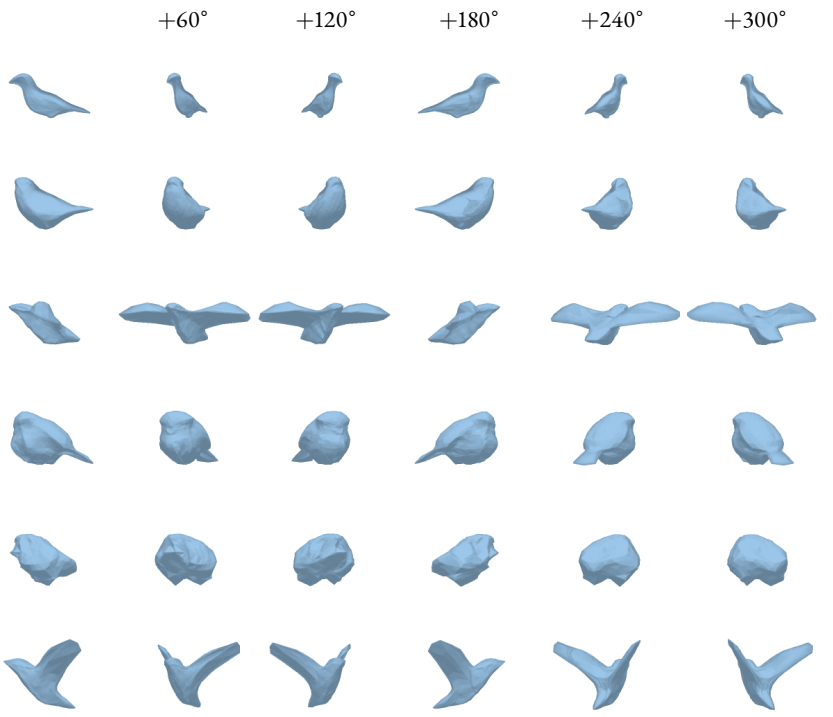


Figure 5.4: Some of the meanshapes learned during training on the CUB dataset using our method initialized with 14 spherical meanshapes.

"has_shape". As shown, even if this dataset does contain objects of the same class "bird", our method obtains comparable results with respect to literature approaches, on both shape and texture metrics. Even if the experiment with multiple shapes does not seem to increase the overall scores, it produces a set of insightful meanshapes learned in an unsupervised manner, as shown in Figure 5.4. Qualitative results are reported in Figure 5.6 (first block).

5.3.3 Ablation study

In this section, we investigate the impact of using one or multiple meanshapes. In addition, we evaluate the influence of the dynamic subdivision approach compared to the static one. In these experiments, we use the Pascal3D+ dataset and

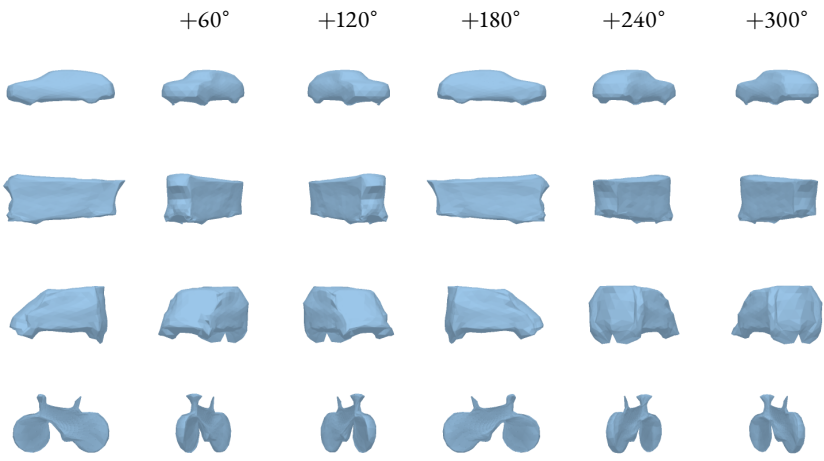


Figure 5.5: Meanshapes learned during training on the classes bicycle, bus, car, motorbike of the Pascal3D+ dataset [260].

extract precise foreground masks with PointRend [115].

Unsupervised shape selection. As our first analysis, we evaluate the impact of the proposed unsupervised shape selection, which enables the training with multiple meanshapes and classes. We test three different training settings using the following object categories: (i) *airplane*, *car*, (ii) *bicycle*, *bus*, *car*, *motorbike*, (iii) all the 12 Pascal3D+ classes. Each setting has been tested using both a single meanshape and a set of N meanshapes, in order to verify the contribution of the usage of multiple learnable shapes and their unsupervised selection. The obtained results are reported in Table 5.4 in terms of 3D IoU, foreground mask IoU, and texture metrics. Our approach with multiple meanshapes provides the best results in all the experimental settings. Furthermore, the meanshapes learned with the four-category setting are depicted in Figure 5.5. Even if the meanshapes do not exactly correspond to the four classes (*e.g.*, the motorbike is missing), the meanshapes are meaningful and represent different object categories. Qualitative results are shown in Figure 5.6 and Figure 5.7.

Dynamic mesh subdivision. We evaluate the contribution of the dynamic mesh subdivision during the training process using the four automotive classes.

Subdivision level	Mask IoU ↑		Texture metrics		
	Pred cam	GT cam	SSIM ↑	L1 ↓	FID ↓
3	0.701	**0.759**	0.600	0.096	395.96
4	0.685	0.756	0.593	0.101	385.68
$3 \rightarrow 4$	**0.711**	**0.759**	**0.607**	**0.094**	**380.15**

Table 5.5: Ablation study comparing different subdivision levels on Pascal3D+ dataset [260]. Model trained on 4 classes (bicycle, bus, car, motorbike) using 4 meanshapes.

We compare three different settings of the 3D mesh connectivity, in terms of icosphere subdivision level: (i) level set to 3, (ii) level set to 4, and (iii) dynamic subdivision starting from level 3 and going up to level 4. Results are reported in Table 5.5. As shown, the method can converge to good results even using a fixed subdivision level. However, a higher level does not always lead to better scores, as in the case of fixed subdivision level 4. On the contrary, increasing the subdivision level during training leads to higher results in terms of both mask IoU and texture metrics. Indeed, dynamic subdivision allows us to take advantage of low subdivision levels during the initial training phase – optimizing the shape smoothness in a faster and easier way – and at the same time leveraging the higher number of faces of high subdivision levels in the second part of the training – improving the finer details and the quality of the texture.

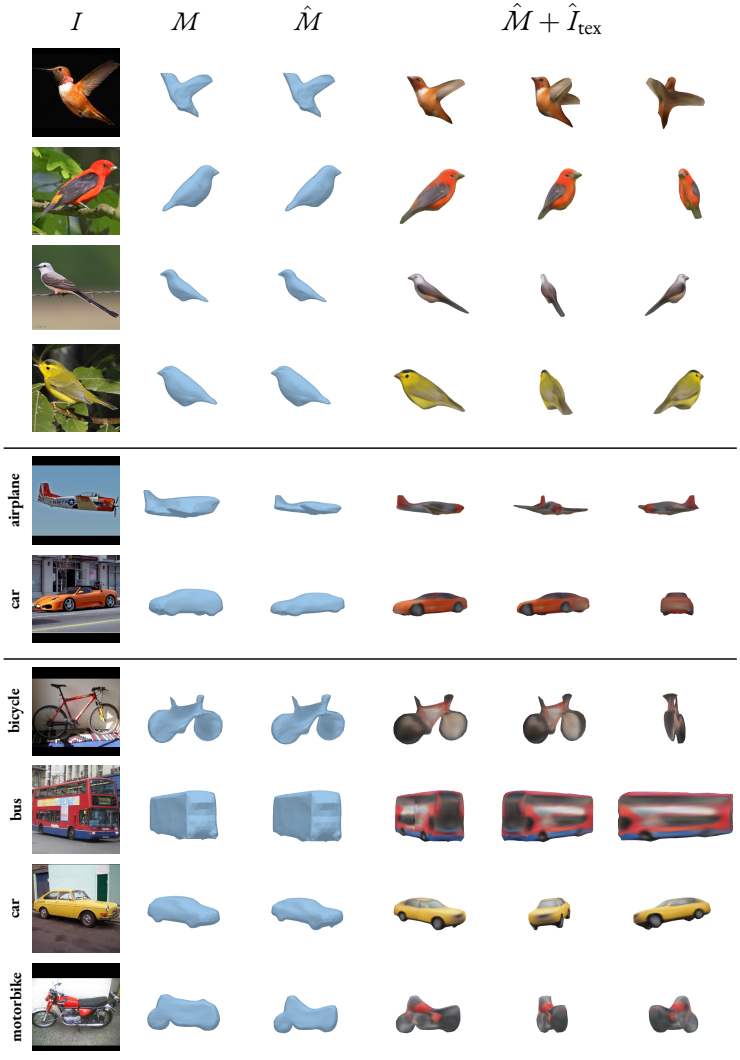


Figure 5.6: Qualitative results on different settings: CUB [246] (birds) and Pascal3D+ [260] (airplane and car, 4 automotive classes). We show the input image I, the output M of the unsupervised shape selection module, the predicted shape $\hat{M}$ and the predicted textured shape $\hat{M} + \hat{I}_{\text{tex}}$ under several 3D rotations over the vertical axis of the predicted pose $\hat{\pi}$.

Figure 5.7: Qualitative results on all 12 classes of Pascal3D+ [260]. We show the input image I, the output M of the unsupervised shape selection module, the predicted shape $\hat{M}$ and the predicted textured shape $\hat{M} + \hat{I}_{\text{tex}}$ under several 3D rotations over the vertical axis of the predicted pose $\hat{\pi}$.

6

Semi-Perspective Decoupled Heatmaps
for 3D Robot Pose Estimation from Depth Maps

Collaborative robots, or *cobots* [198], have entered the automation market for several years now. They have achieved a rather rapid and wide diffusion, also in the corporate world, thanks to the newly introduced paradigm of interaction [70] and collaboration [243]. About 20 years after their introduction, they still have unexplored potential and challenges that have not yet been fully investigated and solved in the literature.

Among others, the knowledge of the instantaneous pose of robots and humans is a key element to set up an effective and fruitful collaboration between them, allowing several applications, ranging from solutions for the safety of the interaction [40] to the behavior analysis [168].

Despite robots usually provide their encoder status through dedicated communication channels, enabling the estimation of their pose and the interaction level [65] through forward kinematics, an external method is desirable in certain

This Chapter is related to the publication "A. Simoni *et al.*, Semi-Perspective Decoupled Heatmaps for 3D Robot Pose Estimation from Depth Maps, RA-L 2022" [6]. See the list of Publications on page 151 for more details.

cases. For example, the robot controller could be designed by third parties that disable or revoke any permission to access the robot encoders. Another use case is the study of the interaction between collaborative robots and humans, *e.g.* focusing on the human prejudice and distrust against robots [191, 183]. In this context, a portable and autonomous setup that does not require any hardware access to the internal states of the robots is preferred. In our experience, such a system has been often accepted by manufacturing companies that are participating in an ongoing study.

Therefore, in this paper, we propose to use non-intrusive and ready-to-use sensors, *i.e.* cameras, to address the 3D *Robot Pose Estimation* (RPE) task and, in this way, to monitor the posture of a given robot through the position of its joints in world coordinates. Different solutions have been explored to this aim, all of them requiring the unpractical application of specific sensors [80] or markers [104] on the robot structure. Differently from these, we propose to use depth cameras [213], which provide light-invariant and precise 3D scene information at a low cost [271], and deep neural networks to directly and accurately predict the 3D location of the robotic joints. Since supervised deep networks usually require a great amount of labeled data, we designed our approach to leverage synthetic data during the training phase and seamlessly work with real cameras and robots at inference. In this way, our method does not need to acquire data in the real world and annotate them, which is known to be an expensive and time-consuming activity.

To this end, we collected and publicly release* a new dataset, namely *SimBa*, that contains synthetic data for training and real-world annotated recordings for evaluation. Regarding the model architecture, we draw knowledge and expertise from the related field of the *Human Pose Estimation* (HPE) [173], being aware of the impressive progress of the computer vision community in that field. Indeed, we present an approach that consists of a novel and effective pose representation, here referred to as *Semi-Perspective Decoupled Heatmaps* (SPDH), that extends the well-known 2D heatmaps to the 3D domain. Existing 2D HPE architectures developed for the RGB domain can be easily adapted to process depth data as

*https://aimagelab.ing.unimore.it/go/rpe

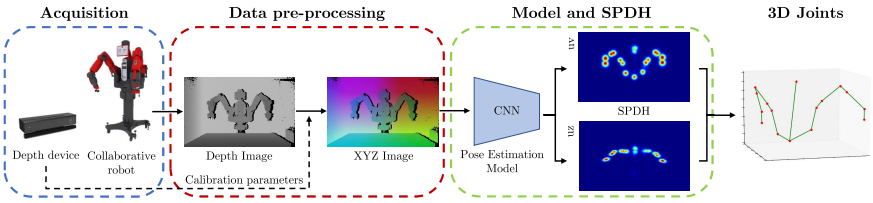


Figure 6.1: Overview of our 3D Robot Pose Estimation (RPE) system. A depth image is converted into an XYZ image which is given as input to a pose estimation deep model. The network predicts the proposed *Semi-Perspective Decoupled Heatmaps* (SPDH) from which the 3D robot pose is computed.

input and predict the proposed SPDH, leading to accurate 3D joint locations in world coordinates.

We demonstrate that this approach overcomes alternative methods in terms of accuracy, adding negligible computational overhead to existing 2D methods. Experimental results confirm the efficacy of the proposed system, paving the way for future research in the field of the 3D Robot Pose Estimation from depth maps.

To sum up, our main contributions are:

- We address the problem of the 3D RPE from depth maps, introducing SPDH, a novel heatmap-based 3D pose representation that can be applied to existing 2D human pose estimation architectures with minimal changes, achieving competitive results.
- We propose an effective and practical training procedure, based on synthetic depth maps, that can be applied to acquire data at scale without expensive and time-consuming manual annotations. We publicly release the simulation parameters and the dataset used for the experimental validation.

6.1 Related work

Robot Pose Estimation (RPE). Only a few works address the RPE task and, to the best of our knowledge, only a minor subset of them make use of depth

data. This is the case of the system proposed by Bohg *et al*. [24], in which, taking inspiration from [219], a random forest classifier is applied to depth images to segment the links of the robot arms, from which the skeleton joints are estimated. In a similar work [256], Widmaier *et al*. propose to directly regress the joint angles without the need to predict robot arm segmentation. However, these methods do not estimate the pose in terms of camera-to-robot coordinates.

Instead of depth maps, the large majority of works focus on RGB images. Labbe *et al*. [123] proposed a method that, given a single RGB image of a known articulated robot, estimates the 6D camera-to-robot pose in terms of rigid translation and rotation through a render-and-compare approach. A reference point and an anchor part are needed to perform the estimation, and their choice significantly affects the performance. In the work by Lee *et al*. [131], the RGB input image is fed to a deep encoder-decoder architecture that outputs one map per keypoint. The final camera-to-robot pose is computed through *Perspective-n-Point* (PnP) [138], assuming that the camera intrinsics and joint configuration of the robot are known. Similarly, in [125] the PnP is used to compute the camera-to-robot pose using a combination of both synthetic and real data. A double system is presented in the work of Tremblay *et al*. [234]: one network predicts the object-to-camera pose while another estimates the robot-to-camera pose. Both networks are trained entirely on synthetic data and the final output is intended to help the robot grasping system, rather than estimating the whole robot pose. Differently from the discussed approaches, our method can be adapted to any heatmap-based 2D pose estimation method to estimate the 3D pose of a robot from a single depth map.

Human Pose Estimation (HPE). The task of estimating the human pose has been extensively investigated in the computer vision community. Similar to the RPE field, the vast majority of works is based on RGB images and outputs only 2D predictions. Recently, several works also addressed the task of 3D HPE from single monocular intensity images, such as in [194] where a coarse-to-fine prediction scheme based on volumetric predictions is exploited to compute both the 2D and then the 3D pose. However, these volumetric methods [101] are often characterized by computational inefficiencies, in terms of complexity and mem-

ory requirement, even though some recent works [56] attempt to address this issue. These considerations drove our choice to focus on HPE architectures that predict the 2D joint positions through heatmaps applied on RGB images, analyzed in the following.

Among the numerous HPE methods based on 2D heatmaps [281], the architecture known as *Stacked Hourglass*, introduced in by Newell *et al*. [175], is developed to process features at different scales and to capture the spatial relationships of the human body, obtaining high accuracy. Recently, Sun *et al*. [227] proposed a multi-scale approach called High-Resolution Network, or simply *HRNet*. HRNet maintains high-resolution representations throughout the entire estimation process. Other works have been proposed to specifically reduce the computational complexity of the existing methods maintaining a satisfactory accuracy. This is the case of the work described in [162], that proposed the *Fast Pose Machine* (FPM) architecture, based on a cascade of detectors with lightweight and efficient CNN structures. The model can employ different backbones as feature extractors such as *Squeezenet* [94] and *MobileNet* [90]. Differently from these methods, the simple architecture proposed by Martinez *et al*. [160] aims to predict the 3D human pose directly from its 2D version. Despite the simplicity of the approach, the reported results reveal a good accuracy independently from the 2D pose detector used during the training phase. Being aware of the high accuracy achieved by these methods, we aim to adapt RGB HPE architectures for depth data and use them as backbone of our method.

Only a limited number of HPE methods are based on depth maps. In the pioneering work of Shotton *et al*. [218], a random forest classifier is used to classify each pixel of the input depth maps and thus to segment the human body. Then, the 3D joint candidates are selected through a weighted density estimator. In [51, 17] authors propose to use a 2D HPE model to predict head position and human poses in depth images. Schnurer *et al*. [214] propose to optimize the Stacked Hourglass [175] architecture reducing its computational load. The predicted 2D pose is then used in combination with a predicted joint-specific depth map in order to obtain the final 3D coordinates of skeleton joints. In other words, the authors proposed a system that combines a heatmap-based prediction for the

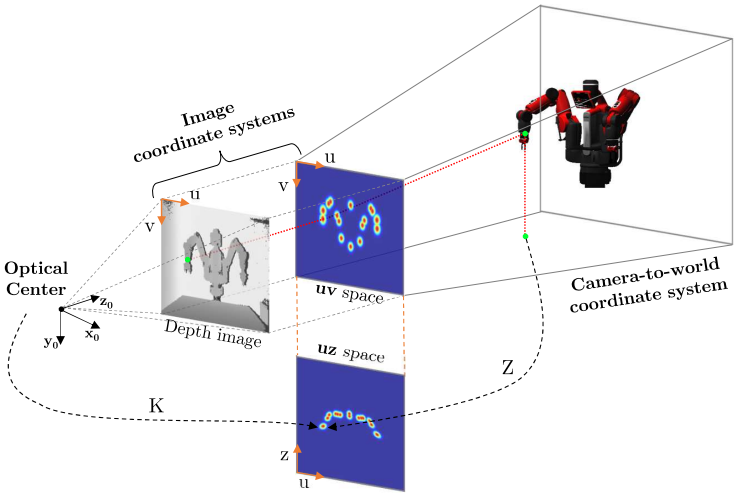


Figure 6.2: Visual representation of the proposed *Semi-Perspective Decoupled Heatmaps* (SPDH). In particular, *uv* and *uz* spaces are depicted in relation to the input depth map and the acquired robot.

2D coordinates and a value regression for the depth value. A Residual Pose Machines is used by Martinez *et al*. [161] to detect only the 2D location of human skeleton joints on the depth images. The depth value of the surface close to a given skeleton joint can be computed by sampling from the depth map using the location of that joint. However, this simple approach is not robust against possible body occlusions and the sensor noise; both of them can significantly alter the depth value in the sampled point. Moreover, this approach can only predict the position on the surface of the robot arm, rather than its center. Finally, we observe that the 3D pose estimation from depth data is not yet deeply investigated in the literature, in particular whether deep learning algorithms are used.

6.2 Semi-Perspective Decoupled Heatmaps

We propose a novel *Semi-Perspective Decoupled Heatmaps* (SPDH) pose representation that relies on projections of the 3D space under the assumption of having a single robot in the image. Each 3D joint location is mapped into two decoupled

bi-dimensional spaces: the *uv* space, *i.e.* the camera image plane, and the *uz* space, composed by quantized Z-values and the u dimension, as depicted in Figure 6.2. The pose estimation algorithm will be trained to generate two heatmaps for each joint, one for each space. The corresponding training heatmaps are Gaussian probability distributions centered on the projections of the joint coordinates.

The *uv* heatmap takes inspiration from the output representation used by most of the recent 2D HPE methods [281]. However, differently from them, each heatmap is constructed using a Gaussian function that has perspective awareness of the joint's distance from the camera.

Formally, given a joint j, the related heatmap $\mathcal{H}_j^{uv}$ is defined in the *uv* space and computed as follows:

$$\begin{aligned} \mathcal{H}_j^{uv}(p) &= \mathcal{N}(p - p_j, \sigma_j) \\ &= \frac{1}{2\pi\sigma_j} e^{-[(p^x - p_j^x)^2 + (p^y - p_j^y)^2]/(2\sigma_j^2)} \\ \sigma_j &= \frac{\sigma^m \cdot f}{Z_j} \end{aligned} \tag{6.1}$$

where f is the focal length of the camera, p_j is the 2D joint location, p is the pixel location, Z_j is the Z coordinate of the 3D joint location and σ^m is the desired standard deviation of the Gaussian distribution in the metric space.

On the other hand, the *uz* heatmap can be seen as the representation of the probability of seeing a joint at the image coordinate u if it were at a distance z from the camera. To generate $\mathcal{H}_j^{uz}$, the following two-step process is applied.

Firstly, we define a restricted depth space $\bar{Z} = \{\bar{Z}_i \in Z;\ \bar{Z}_{min} \leq \bar{Z}_i \leq \bar{Z}_{max}\}$ and we split it into slices of size ΔZ. The number of slices represents the height of the *uz* heatmap and can be computed as follows:

$$z = \frac{\bar{Z}_{max} - \bar{Z}_{min}}{\Delta Z} \tag{6.2}$$

In this way, there is a direct connection between the 2D *uz* space and the depth-aware $\bar{Z}$ space. For simplicity, we sample the space $\bar{Z}$ so that z has the same value of v, *i.e.* the dimension of the two heatmaps is the same.

Secondly, we proceed with the computation of the heatmap. We project each point p^{xy} of the image into the 3D reference frame according to its (x, y) coordinates. We apply the intrinsic parameters of the camera K, *i.e.* the focal length f and the optical center c, and obtain

$$P(p) = \left((p^x - c) \cdot \frac{\bar{Z}^y}{f}, \bar{Z}^y\right) \tag{6.3}$$

where $\bar{Z}^y$ is the corresponding value in camera coordinates of z sampled in p^y. Then, we compute the euclidean distance $d(p) = ||P(p) - P_j||$ between each point P – corresponding to each location p^{xy} on the uz space – and the 3D ground-truth joint location $P_j = [X_j, Z_j]$ – excluding the Y axis. We use this distance to compute the value of the heatmap in each point p as:

$$\begin{aligned} \mathcal{H}_j^{uz}(p) &= \mathcal{N}(P(p) - P_j, \sigma^m) \\ &= \frac{1}{2\pi\sigma^m} e^{-d(p)/(2\sigma^{m2})} \end{aligned} \tag{6.4}$$

A visual representation of the relation between the proposed Semi-Perspective Decoupled Heatmaps and the 3D space is shown in Figure 6.2; examples of SPHD are also reported in Figure 6.3, third and fourth row.

6.3 3D Robot Pose Estimation

The proposed approach for the 3D Robot Pose Estimation is based on the aforementioned SPDH representation. Figure 6.1 depicts a visual overview of each step, detailed in the following subsections, adopted to output the final prediction.

6.3.1 Depth data acquisition

The first step is the data acquisition procedure based on a depth camera, *i.e.* a device capable of acquiring depth maps, and a collaborative robot. Despite the challenges and limitations posed by the usage of depth sensors [199], we observe that depth data acquired through the same depth device do not usually require

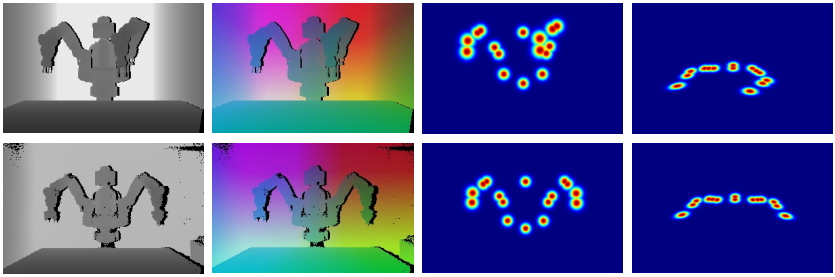


Figure 6.3: Examples for synthetic (first line) and real (second line) depth data. Then, XYZ image and the novel SPDH representation, depicted through both the heatmap in *uv* space and heatmap in *uz* space, are reported.

the adoption of challenging domain randomization techniques [231] typically applied on RGB synthetic images to bridge the gap between real and synthetic data [131, 233]. Indeed, depth data provide robustness to light changes and variations in background textures [213], helping to make the transition from synthetic to real depth data more straightforward and the synthetic data generation easier and less time-consuming. Moreover, depth cameras provide 3D information of the scene that the method can leverage to estimate the 3D pose of the robot.

Formally, we define an acquired depth map as the couple $D_M = \langle D, K \rangle$, composed by the matrix of distances $D = \{d_{ij}\}$, $d_{ij} \in [0, R]$, in which values are between 0 and the maximum depth range R, and K, *i.e.* the perspective projection matrix computed with the intrinsic parameters of the acquisition device. In particular, d_{ij} represents the distance between the optical center and a surface containing the point p_{ij} and parallel to the image plane; then, D can be visualized as a depth image, encoded as one-channel gray-level image I_{D}, as reported in the first line of Figure 6.3.

6.3.2 Data pre-processing

As mentioned, the collected depth map D_M contains information about the distance between the camera and the objects' surfaces of the acquired scene, while our purpose is to have an input with explicit 3D information. To this end, we convert the depth image $I_{\mathrm{D}}^{1\times H\times W}$ into an XYZ image $I_{\mathrm{XYZ}}^{3\times H\times W}$, where each pixel

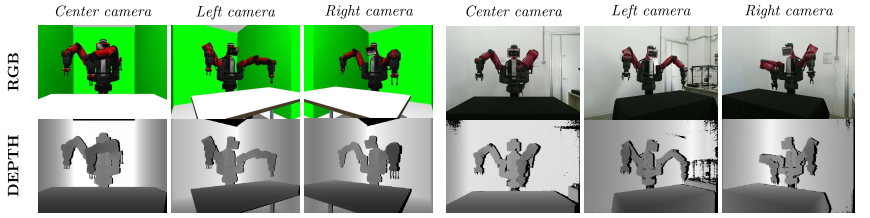


Figure 6.4: Examples of synthetic (first three columns) and real (last three columns) RGBD frames acquired from different camera positions.

$q_{ij} \in \mathbb{R}^3$ corresponds to the projection in the 3D space of the original pixel $d_{ij} \in \mathbb{R}^1$, by applying the inverse of the camera intrinsic matrix K and then multiplying by the corresponding depth value. Thus, each pixel of the resulting image represents the 3D coordinate of that pixel in the depth image: visual samples are shown in the second row of Figure 6.3. The resulting image is then normalized independently along the three axes X, Y, Z, before being processed by the network.

6.3.3 Model architecture

As mentioned above, our method is designed to work with a generic deep learning-based architecture belonging to the 2D HPE field. Thanks to the adopted SPDH representation, it is straightforward to adapt the selected architecture for our 3D RPE task, as detailed in the following. The network takes as input an XYZ image of size $3 \times h \times w$ and outputs a $2n \times h \times w$ tensor, where n is the number of joints. The output represents the uv and uz heatmaps for each keypoint, where each pixel value determines the likelihood that a keypoint lies in that position. To compute the predicted joint $\hat{P}_j = [\hat{X}_j, \hat{Y}_j, \hat{Z}_j]$, we exploit the maximum values of the heatmaps; for the uv map, we get the 2D coordinates $\hat{p}_j$ of the Gaussian peak; for the uz map, we consider just the z coordinate of the Gaussian peak which is then converted into a depth metric value as follows:

$$\hat{Z}_j = (z \cdot \Delta Z) + \bar{Z}_{\text{min}} \tag{6.5}$$

Finally, $\hat{p}_j$ is projected in the 3D space by applying the inverse of the camera intrinsic matrix K and then multiplying by $\hat{Z}_j$, obtaining the final prediction $\hat{P}_j$.

6.4 Experiments

6.4.1 SimBa dataset

To evaluate the proposed approach, we collected a new dataset, namely *SimBa*, composed of both synthetic and real images, which are used respectively for training and testing.

Among several collaborative robots, we choose the *Rethink Baxter*, which has been widely used in the research community. In the collected dataset, the Baxter moves respectively to a set of random pick-n-place locations on a table, assuming realistic poses.

Synthetic data. For the synthetic dataset, we use ROS for interacting with the synthetic robot model, the cameras and the environment, and Gazebo for rendering the simulated world. The dataset consists of a set of sequences recorded from 3 RGB-D cameras (center, left, right) that are randomly moved within a sphere of 1 meter diameter from their anchor. In particular, we collect two simulation runs with different initializations. Each run contains 20 recording sequences, composed of 10 pick-n-place motions which are equally split between the left and right robot arm. Each sequence is recorded at the same time by the three cameras, whose position is randomly changed at the beginning of each sequence. The simulation runs at 10 fps and records RGB (1920×1080) and depth (512×424) frames from each camera, the 16 robot joints positions, the pick-n-place locations, and the camera positions. The synthetic dataset contains a total of 40 sequences with over 350k annotated RGBD frames.

Real data. The real dataset was acquired using the ROS framework. The time-of-flight *Microsoft Kinect One* (second version) was used to record the moving robot from 3 camera positions (center, left, right). In this case, each camera position contains 20 pick-n-place sequences which are split equally between the left and right arm. The recording runs at 15 fps for the RGB (1920×1080) and

depth (512×424) frames and at 40 fps for the 16 robot joints positions. The real dataset contains over 20k annotated RGBD frames.

After the acquisition of the synthetic and the real dataset, we align each depth frame to the RGB sensor using the corresponding extrinsic parameters. Some examples of the frames recorded in both scenarios are depicted in Figure 6.4, in which different levels of precision and noise are visible.

6.4.2 Experimental setup

We split the dataset into 28 train sequences with over 26k frames, 4 validation sequences with over 3.5k frames, and 8 test sequences with over 7k frames. For the training phase, we sampled each synthetic sequence every 10 frames to avoid pose redundancy. The synthetic test set was used only to check the efficacy of the method in the initial phase of the project. To evaluate the method and the competitors, we sampled each sequence of the real dataset every 5 frames obtaining a test set of 4k frames.

To generate the ground truth of the joint positions with SPDH representation, we sampled a space $\bar{Z} = [500 \text{ mm}, 3380 \text{ mm}]$ with a depth step $\Delta Z = 15\text{mm}$. The values of the $\bar{Z}$ space were selected according to the range of the depth sensor that is up to 5 m [213]. Both heatmaps are computed with $\sigma^m = 50$ mm. Visual examples are depicted in Figure 6.3.

We use as input a resized depth image I_D of resolution 384×192 applying a 3D data augmentation during training. We first transform the depth image into a pointcloud and then apply a random 3D rotation of $[-5°, 5°]$ along X or Y axis and a random translation of $[-80\text{mm}, 80\text{mm}]$ along X or Z axis. Then, the pointcloud is converted again into a depth image from which the XYZ image is computed, as explained in Section 6.3.

We adapt the state-of-the-art 2D HPE architecture called HRNet-32 architecture [227] as the pose estimation model to predict our SPDH from which the 3D robot pose is computed. The network is trained from scratch for 30 epochs on our synthetic dataset using the $L2$ loss between the predicted and the ground-truth SPDH.

Approach	Network	mAP (%) ↑ 40mm	60mm	80mm	100mm	ADD (cm) ↓ L1	L2
2D to 3D from depth	Stacked Hourglass (1 HG) [175]	8.98	31.21	49.12	66.11	15.63 ±6.62	11.59 ±5.32
2D to 3D from depth	Stacked Hourglass (2 HG) [175]	10.13	31.94	50.54	67.14	14.88 ±6.10	11.06 ±5.04
2D to 3D from depth	FPM (MobileNet) [162]	9.83	29.09	49.13	66.70	16.25 ±6.66	11.66 ±5.38
2D to 3D from depth	FPM (SqueezeNet) [162]	10.84	32.87	51.58	67.87	15.12 ±6.11	11.22 ±5.07
2D to 3D from depth	HRNet-32 [227]	12.52	33.23	49.57	67.18	14.51 ±5.59	10.86 ±4.64
2D to 3D from depth	HRNet-48 [227]	12.15	32.55	50.83	67.99	14.62 ±5.78	10.99 ±4.81
3D regression	ResNet-18 [82]	9.40	19.99	27.06	44.44	17.10 ±5.43	12.20 ±4.12
2D to 3D lifting	Martinez et al. [160] *	26.96	37.98	48.40	58.33	14.01 ±4.84	10.03 ±3.53
Volumetric heatmaps	Pavlakos et al. [194]	18.15	42.24	61.60	86.15	10.35 ±1.07	7.11 ±0.65
SPDH (ours)	HRNet-32 [227]	**53.75**	**79.75**	**93.90**	**98.12**	**6.62** ±1.53	**4.41** ±1.09

* relative joint positions

Table 6.1: Comparison between ours and literature approaches trained using an XYZ image as input

We used a batch size of 16, *Adam* [113] as optimizer and $1e^{-3}$ as learning rate with 10 as decay factor after 50% and 75% of the training epochs. At test time, the network is evaluated on the real dataset, without using any domain adaptation techniques, differently from [131] which is based on RGB images.

6.4.3 Metrics

For the 3D evaluation of the predicted robot poses, we exploit the average distance metric (ADD) [131, 261], in terms of the average distance of all 3D robot joints to their ground truth positions. ADD metric is useful to merge translation and rotation errors in a single value. In particular, we compute ADD reporting $L1$ and $L2$ average distances expressed in centimeters with standard deviations w.r.t. ground truth positions. Here, lower results represent good performance. In addition, we compute the *mean Average Precision* (mAP), which expresses the percentage of 3D keypoints within a certain threshold. In our experimental validation, we set 4 different thresholds at $40, 60, 80, 100$ mm. Here, higher results are better. We believe this metric shows the performance in a more straightforward manner compared to ADD, highlighting the accuracy of the system at different thresholds.

6.4.4 Competitors

To validate the proposed RPE approach, we compare it with four alternatives, belonging to the HPE domain, that can be adapted to predict the 3D robot pose:

- "2D to 3D from depth" is a two-step approach. Firstly, a state-of-the-art HPE method [175, 162, 227] predicts the 2D robot pose on depth images. Then, the Z value is sampled from the depth to obtain the 3D joint coordinates.
- the "3D regression" method corresponds to an architecture that directly regresses the 3D joint coordinates starting from a depth image. We empirically found that the best performance is obtained using the well-known ResNet [82] architecture, adapted and trained for regressing the 3D robot joints.

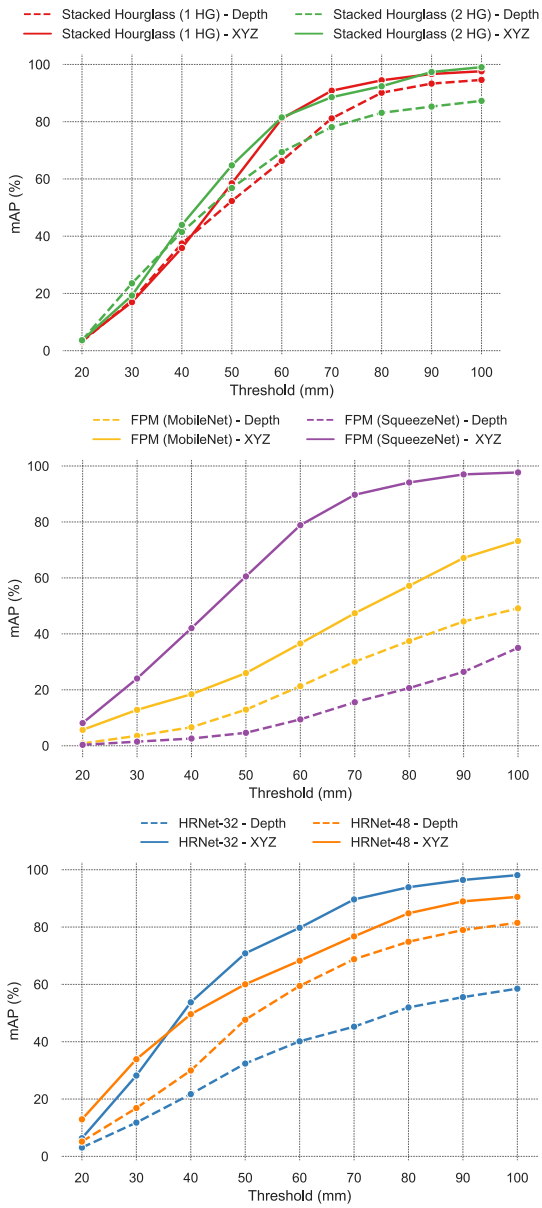


Figure 6.5: Comparison on mAP scores between different networks trained giving as input just a depth image I_{D} or the proposed XYZ image I_{XYZ} (dashed line = input I_{D}, solid line = input I_{XYZ}, same color = same network configuration).

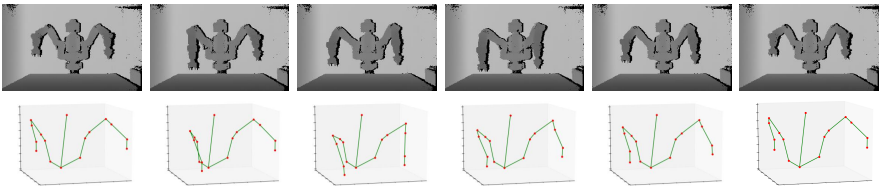


Figure 6.6: Real depth images and final predicted 3D robot pose for a random pick-and-place motion with both arms.

- the "2D to 3D lifting" approach directly converts a set of already predicted 2D joint locations to their 3D counterpart, relative to a root joint. In particular, we select the network proposed by Martinez *et al*. [160].
- a "volumetric heatmap" approach that outputs 3D heatmaps. In our experimental validation, we adopt the state-of-the-art method proposed in [194], which predicts a volume with size $d \times w \times h$ – with $d = 64$ – for each joint and uses its maximum value as the 3D joint location.

6.4.5 Results

As shown in Table 6.1, our method performs better than other competitors in all the metrics, especially in terms of mAP with low distance thresholds. We observe that the 2D to 3D from depth approach leverages the high precision of 2D pose estimation models, but is limited by the depth map; indeed, it samples the depth values at the 2D joint coordinate, resulting in predicting a 3D location on the robot surface rather than onto the inner joint position. The approach based on direct 3D regression does not reach satisfactory results, confirming that the task is not trivial. The 2D to 3D lifting method uses a relative joints' position with regard to a specific root joint – the robot base in our case. Thus, this approach is not directly comparable to our proposal, since it needs a post-processing step in which the camera pose has to be known or predicted and then applied to the 3D coordinates in order to get the correct camera-to-robot values. However, we noticed overfitting phenomena on synthetic training data, resulting in low accuracy when testing on real data.

Network	Δ **Z** (mm)	**mAP** (%) ↑ **40mm**	**60mm**	**80mm**	**100mm**
Stacked HG (2 HG) [175]	7.5	46.83	78.08	89.52	96.07
Stacked HG (2 HG) [175]	15	43.94	81.50	92.39	99.06
Stacked HG (2 HG) [175]	30	44.55	76.04	87.22	99.03
HRNet-32 [227]	7.5	51.80	69.42	79.27	88.24
HRNet-32 [227]	15	53.75	79.75	93.90	98.12
HRNet-32 [227]	30	43.66	69.24	86.38	97.20

Table 6.2: Comparison of results using different values of ΔZ

The approach based on volumetric heatmaps obtains a good level of accuracy, even if still lower than our method. However, the main issue of this approach is the high computational load that predicting 3D volumes requires. Indeed, a volumetric heatmap represents a quantized 3D space that quickly grows in size in order to increase the precision of the method. This problem can be noticed during training when this approach requires almost double the amount of GPU memory (8.3GB) than our method (4.7GB). To summarize, our approach achieves the best results in predicting the inner joints of the robot, which is a challenging task to solve with the alternative approaches.

Finally, some qualitative results of our method are depicted in Figure 6.6, reporting the initial depth image and the final 3D robot skeleton.

6.4.6 Ablation study

To further evaluate our approach, we perform an ablation study to investigate the impact of using different network inputs and of sampling a different $\bar{Z}$ space when computing the *uz* heatmap.

In the first experiment, we adapt three different 2D HPE baselines [175, 162, 227] in order to learn the proposed SPDH output. In particular, they were trained using two different inputs: a depth image I_{D} and the proposed XYZ image I_{XYZ}. As can be seen in Figure 6.5, the results confirm that the network learning process benefits from an input I_{XYZ} with explicit 3D information. Indeed, using only the depth values from I_{D} makes the network independent from

the intrinsic parameters of the camera, which are needed to learn a meaningful 3D representation of the world from a 2D space.

With the hypothesis that a smaller value of ΔZ should take to higher mAP scores, the latter experiment explores different samplings of the $\bar{Z}$ space. We select the networks [175, 227] with the best performances and train them with three different ΔZ values, *i.e.* 7.5, 15 and 30 mm. We observe that using $\Delta Z = 30$ a bigger space $\bar{Z} = [0, 5760]$mm is sampled, while for $\Delta Z = 7.5$mm we adapt the input for our system increasing the size of input images to 384×384 adding upper-lower padding and maintaining the same $\bar{Z}$ space as our main experiment in Section 6.4.2. Experimental results reported in Table 6.2 reveal that the choice of the parameter ΔZ, which can be potentially non-trivial since it changes the size of the *uz* heatmap, does not have a significant impact on the performance of the whole proposed system, tending to avoid the need to ad hoc decrease or increase ΔZ value for different application contexts.

6.5 Limitations

Although our experimental section shows promising results, we observe that the output is limited to a single robot in the acquired scene. In addition, the influence of other objects in the scene on the final prediction needs to be investigated, since these objects could change the visual appearance of the scene or produce occlusions on robot surfaces. The proposed system is one of the first attempts in this field, and it can be improved in many terms, *e.g.* on the temporal smoothness of the pose.

7

D-SPDH:
Improving 3D Robot Pose Estimation in Sim2Real scenario via Depth Data

Vision-based Robot Pose Estimation (RPE), *i.e.* the ability to infer the pose of a robot from an external camera viewpoint, is an enabling technology for many robotics applications, *e.g.* object grasping [21], robot manipulation [22], and motion planning [128]. In particular, the precise estimation of the 3D world coordinates of robot joints in real time has a significant impact on many real-world safety applications, especially regarding human-machine interactions, such as collision detection and avoidance [131] with collaborative robots (cobots) [198].

Indeed, in the context of Industry 4.0 [127], experts agree that cooperation between humans and intelligent agents [254, 255], rather than complete removal of humans, will be a key enabler for the advancement in manufacturing [117]. In this context, safe interaction between humans and cobots is a crucial element to

This Chapter is related to the publication "A. Simoni *et al.*, D-SPDH: Improving 3D Robot Pose Estimation in Sim2Real scenario via Depth Data, Under Review" [3]. See the list of Publications on page 151 for more details.

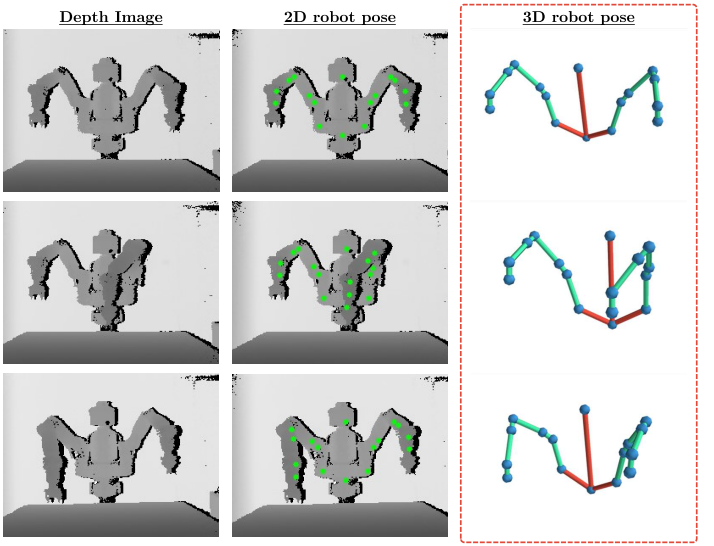


Figure 7.1: Visualization of the tackled task, *i.e.* Robot Pose Estimation (RPE) from depth data. Given as input a depth image, our proposed D-SPDH method is able not only to predict the 2D pose but also to recover the full 3D pose of the robot in order, for instance, to guarantee a safe human-machine interaction.

be investigated [40]. Moreover, in future generations of manufacturing, robots and operators will share the workspace and have physical contact, raising new aspects related to social and physical coordination between coworkers [45, 191]: topics that are analyzed also through the robot's pose.

Therefore, in this paper, we focus on the development of D-SPDH, a vision-based method able to predict the robot joint positions in the 3D world relying only on depth maps as input (see Figure 7.1). In other words, we propose a system that is completely agnostic about the robot's internal state, in terms of encoders, communication interfaces, and other electromechanical components, that outputs 3D coordinates.

Specifically, with the term D-SPDH we refer to a double-branch Convolutional Neural Network (CNN) architecture trained only on depth data. We believe that using depth data only can lead to two main advantages: (i) it provides information for a more accurate estimation of real-world 3D coordinates [278], and, (ii) it avoids illumination issues that typically affect systems based, for in-

stance, on RGB data [271]. Each branch of the model is specialized in one of the Semi-Perspective Decoupled Heatmaps (SPDH) [6] representation, which has been proven to be effective in encoding the information to address the robot pose estimation task.

D-SPDH is developed in the Sim2Real scenario [87], *i.e.* the model is trained only on synthetic depth data, easily obtainable with simulators, and tested on real sequences. We observe that this scenario limits the difficulties in acquiring a large amount of varied and labeled depth data usually required to train deep learning-based systems [46]. Besides, in this manner, the training procedure and then the method deployment are not tied to a specific acquisition depth device and its technology, which can introduce artifacts in the depth data [213] limiting generalization capabilities [199]. Furthermore, the use of depth data tends to reduce the domain gap between synthetic and real data without domain randomization or similar techniques [233].

The proposed method is evaluated on an extended version of the SimBa dataset [6] that we created called SimBa^{++}. SimBa^{++} consists of both synthetic sequences collected through Gazebo simulator [116] and real data acquired through the second version of the Microsoft Kinect depth device. This evolution of the dataset includes new challenging real-world scenes with double-arm movements.

Finally, in this paper, we also seize the opportunity to thoroughly investigate challenges and future research scenarios of 3D RPE from depth data. We believe that such an emerging field has important practical and theoretical implications that have not yet been fully analyzed and explored in the literature.

Summarizing, the contributions of this paper are:

- We propose D-SPDH, a double-branch architecture capable of predicting reliable 3D world locations of robot joints using only depth data. The input depth map is converted into a different depth representation (*i.e.* XYZ image, see Section 7.2.1), which is fed into a backbone connected to two branches for the prediction in two different joint spaces through the SPDH representation (Section 7.2.3). A visual summary of the proposed system is given in Figure 7.2.

- We release SimBa^{++}, an extended version of the previous SimBa dataset [6], which is used for training, experimental evaluations and comparisons. As one of the first datasets in its category featuring both synthetic and real depth data with 3D annotations, we describe and analyze SimBa^{++} in detail in Section 7.3.1.

- We provide a thorough description and experimental comparison of several approaches to the problem of RPE. We start from 2D RPE, move to 2D to 3D projection, and finally to the full 3D pose estimation (Section 7.3.4). This study enables us to draw some conclusions and outline challenges and opportunities for future research in the field of 3D RPE (Section 7.4).

7.1 Related work

Estimating the correct 3D location of an object from the perspective of an external camera is a complex problem, so different approaches have been presented in the last decades, following the new advances in computer vision and deep learning. We present an overview of the current literature related to robot pose estimation dividing the works into two groups: (i) *hand-eye calibration* divided into marker-based and learning-based methods, and (ii) *rendering-based approaches*, which use rendering methods to project a 3D synthetic robot model into the scene and predict the pose. Finally, a section is dedicated to currently available datasets for robot pose estimation.

Hand-eye calibration. In robotics, the common approach to estimate the absolute pose of a robot with respect to the camera is the Hand-Eye Calibration [89, 83]. This approach, for instance, used in [185, 267, 95, 192], consists of attaching a fiducial marker (*e.g.* ArUco [63], ARTag [59], AprilTag [179]) to the end effector that is tracked through multiple frames. These algorithms exploit forward kinematics and multiple frames to solve an optimization problem using 3D-to-2D correspondences to get the camera-to-robot transform. However, these methods require the installation of markers on the manipulator, which is

not always a feasible operation depending on the working scenario.

Nonetheless, with recent advances in human pose estimation [281], many works have been proposed to estimate the camera-to-robot pose using CNNs. We divide these approaches into two groups, depending on the input image type: *depth-based* and *RGB-based*. The first group is a minor subset in which depth data is used to predict the robot's pose. [24], taking inspiration from [219], apply a random forest classifier to the depth images to segment the links of the robot arm from which the skeleton joints are estimated. Similarly, the method described in [256] directly regresses the joint angles without the segmentation prior. However, we observe that these methods retrieve just the joint angles but do not recover the absolute pose with respect to the camera.

On the other hand, RGB-based methods represent the large majority. Lambrecht *et al.* proposes in [124] a method that combines synthetic and real data to train a keypoint localization network that predicts 2D robot joints. Computing the 3D joint configuration from the forward kinematics, a *Perspective-n-Point* (PnP) [134] [138] algorithm retrieves the 3D robot pose in camera coordinates. Similarly, Zuo *et al*. [289] also presents a keypoint detector but trained on synthetic data only. Instead of PnP, they use a non-linear optimization to regress the camera pose and joint angles of a small low-cost manipulator. Recently, the work [131] demonstrates that learning-based approaches could replace classic marker-based calibration also for standard manipulators (*e.g*. Rethink Baxter and Franka Emika Panda). They exploit synthetic data for training, feeding RGB images into an encoder-decoder network that predicts the 2D pixel coordinates of the robot joints. Assuming the camera intrinsics and the configuration of the joint angles are known, the camera-to-robot transform is computed via PnP. Moreover, Tremblay *et al*. [234] extended the previous work to retrieve the camera-to-object transform. Their pipeline consists of two networks, one for the camera-to-robot pose [131] and one for the camera-to-object pose, with the main goal of improving the grasping performance of the robot.

In contrast with the discussed works, our D-SPDH method directly regresses the camera-to-robot pose using a heatmap representation of a 3D pose. In this way, the proposed system can be used with any network predicting heatmaps,

Dataset	Year	Robots	Synth	Real	Data type	Frames	Notes
CRAVES [289]	2019	1	✓	✓	RGB	5.5k	
DREAM [131]	2020	3	✓	✓	RGB	357k	
WIM [176]	2022	7	✓		RGB	140k	
CHICO [211]	2022	1		✓	RGB	≈1M	
SimBa [6]	2022	1	✓	✓	RGB-D	370k	Single-arm only
SimBa^{++}	2023	1	✓	✓	RGB-D	380k	Double-arm

Table 7.1: Datasets available in the literature for the Robot Pose Estimation task. Further details are reported in Section 7.1.

as those developed for 2D human pose estimation. As a result of our direct 3D regression, we do not use any PnP algorithm making the proposed approach agnostic to the robot state, in terms of joints and angles, which sometimes could be unknown.

Rendering-based approach. Going beyond learning-based methods, recent works [123, 176] propose approaches based on rendering. With the growing interest in neural rendering techniques [166, 251], the goal is to use synthetic robot models and optimize the camera-to-robot pose estimation by projecting them into the scene. Labbe *et al*. [123] paves the way to this field of research presenting the first method for robot pose estimation based on the *render&compare* paradigm. This optimization algorithm iteratively refines an initial robot state defined as the joint angles configuration and the pose of an anchor part with respect to the camera. At test time, the approach could work also with unknown joint angles, but the drop in performance is significant [123]. Furthermore, inspired by [251], the work of [176] proposes a self-supervised method exploiting both an explicit rough approximation of the robot body and an implicit refinement of it. To compensate for the lack of 3D pose supervision, the approach uses multi-view sequences of an articulated moving robot with annotated foreground masks.

Our D-SPDH is also a supervised learning-based approach, but differently from the methods described above it works without the knowledge of the robot state and does not need multiple views. Moreover, D-SPDH is considerably faster than methods based on volume rendering (see Section 7.4.2). These elements

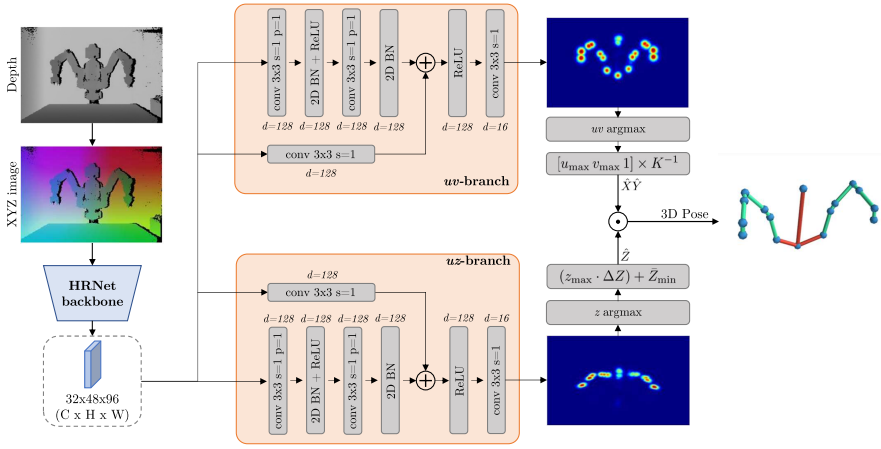


Figure 7.2: Overview of the proposed D-SPDH method: an initial depth map is firstly converted in the XYZ representation (Section 7.2.1) and then used as input for the HRNet-32 [227] backbone that extracts a set of visual features elaborated separately by two branches, *i.e.* *uv*-branch and *uz*-branch (Section 7.2.3). The output of each branch consists of an SPDH representation that is finally converted into the 3D robot skeleton.

simplify deployment to real-world scenarios, where information is not always reliable and speed is crucial.

Datasets for Robot Pose Estimation. Datasets are essential in computer vision, especially for training deep learning architectures. Unfortunately, collecting 3D annotated data for robot pose estimation in the real world is costly. An emerging solution to this problem is the use of simulators to generate synthetic data. As shown in Table 7.1, only four datasets are currently available for the problem of robot pose estimation and they contain exclusively RGB images: CRAVES [289], DREAM [131], WIM [176], and CHICO [211].

CRAVES is a synthetic and real dataset for the pose estimation of an OWI-535 low-cost manipulator. It contains 5k synthetic RGB images generated with Unreal Engine 4 (UE4) and background domain randomization, and 537 real RGB images with annotations for 2D keypoints and visibility. DREAM is a more complex dataset covering three robots, *i.e.* Franka Emika Panda, Kuka LBR with Allegro, and Rethink Baxter. The synthetic data are generated with UE4 using domain randomization [231] and consist of 100k RGB images for each

robot with 2D/3D keypoint locations and robot joint angles as annotations. The real data covers only the Franka Emika Panda and consists of two sets of images: Panda-3Cam containing 17k annotated RGB images collected with 3 different cameras and Panda-Orb that handles a variety of camera poses with 40k annotated RGB images collected from a RealSense camera. WIM is a smaller synthetic-only dataset generated with the python-based renderer NViSII [170] together with PyBullet [41] for animations. It contains 1k RGB frames of a synchronized video with 20 viewpoints for 7 different robots. Finally, CHICO is a real dataset for human-robot collaboration with contact and represents a benchmark for human pose forecasting and collision. It contains 240 RGB HD sequences in which 20 human operators work together with a 7-DoF KUKA LBR robot in a shared workspace.

Our SimBa^{++} is therefore the only dataset that features both RGB and depth data. The dataset contains both synthetic and real sequences of a Rethink Baxter robot, with annotated 3D keypoint locations and camera positions. These elements make our dataset the first suitable for both RGB and depth-based approaches, which can use either synthetic or real data and be also compared for their domain adaptation abilities. SimBa^{++} contains 350k synthetic images and 30k real images.

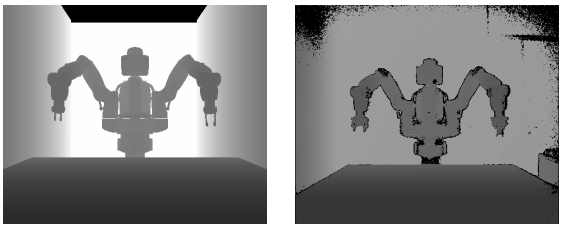
7.2 Proposed method

An overview of the pipeline is depicted in Figure 7.2.

7.2.1 Sim2Real working scenario

The proposed system uses depth data only. Indeed, we observe that the use of depth devices, especially if based on infrared light, represents an effective and low-priced solution to acquire 3D data robust to light changes and variations in background textures [60].

Moreover, we work in the Sim2Real scenario: during training, the input is a synthetic depth map, while in test mode it is acquired by a real depth sensor. A visual comparison between the synthetic and real depth maps is shown in Fig-



(a) Synthetic Depth Image *(b) Real Depth Image*

Figure 7.3: Visual comparison of two depth maps depicting the same scene but acquired in two different ways: on the left is the synthetic image, obtained through the use of the Gazebo simulator, while on the right is the real depth image, acquired through the second version of the Microsoft Kinect device. As shown, these two images are visually different, for instance presenting different levels of noise (black dots) and depth accuracy.

ure 7.3. We aim to work in the challenging Sim2Real scenario in order to avoid the time-consuming acquisition and annotation procedures and develop a system independent of the type of depth sensor. Thus, the quality of depth data (*i.e.*, depth accuracy, format, resolution) is strongly influenced by the acquisition device [213] and negatively affects the performance and generalization capabilities of vision-based systems, especially those based on deep learning architectures, when they are trained and tested on images acquired with different depth devices or technology [199]. A solution consists of acquiring a great variety of depth data with a new depth sensor every time and making an additional effort to finetune a model on the new sequences: this unpractical and time-consuming approach leads us to investigate the Sim2Real scenario.

7.2.2 Processing of input depth data

From a formal point of view, a depth map can be defined as $D_M = \langle D, K \rangle$ where D is the measured matrix of distances d_{ij} between the acquisition device and the points in the scene, and K is the perspective projection matrix, obtained as the intrinsic parameters of the depth camera. It is worth noting that the maximum acquisition range of a real depth map relies on the technology used, mainly based on Structured Light (SL) or Time-of-Flight (ToF) [274], and on the specific sensor quality and resolution. d_{ij} is defined in the range $[r, R]$, where r and R are

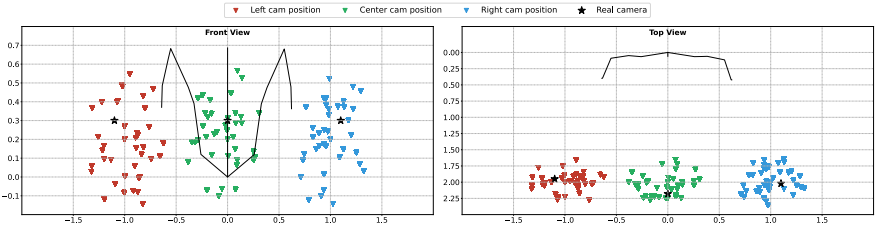


Figure 7.4: Visualization of the camera positions exploited during the acquisition procedure of the SimBa^{++} dataset with respect to the robot location (here represented through its skeleton). Different views, front-view (left) and top-view (right), of the acquisition scenes are reported, highlighting differences between the synthetic and real collection procedures.

respectively the minimum and the maximum measurable ranges.

The input depth map is converted into an XYZ image $I_{\mathrm{D}}^{1\times H\times W} \rightarrow I_{\mathrm{XYZ}}^{3\times H\times W}$, *i.e.* a 2D representation formally defined as follows:

$$I_{\mathrm{XYZ}} = \pi(D \cdot K^{-1}) \tag{7.1}$$

where π is the projection in the 3D space of every value d_{ij} through the inverse of the projection matrix K. The intuition behind XYZ representation is to have an input image that limits the above-mentioned differences between synthetic and real depth data, improving the performance of the adopted model in the Sim2Real scenario. This consideration is confirmed by the experimental results reported in Section 7.3.4.

7.2.3 Intermediate pose representation through D-SPDH

To estimate the 3D pose of the robot, we first regress an intermediate representation referred to as Semi-Perspective Decoupled Heatmaps (SPDH) [6]. This representation decomposes the 3D space into two bidimensional spaces where the robot joint locations, organized as in Figure 7.5, are represented as heatmaps: (i) the *uv* space corresponding to the camera image plane, and (ii) the *uz* space, containing quantized values of the Z dimension of the 3D real world. The first space represents the front view of the scene in which the heatmaps H^{uv} are computed with a perspective awareness of the distance of the joints with respect to

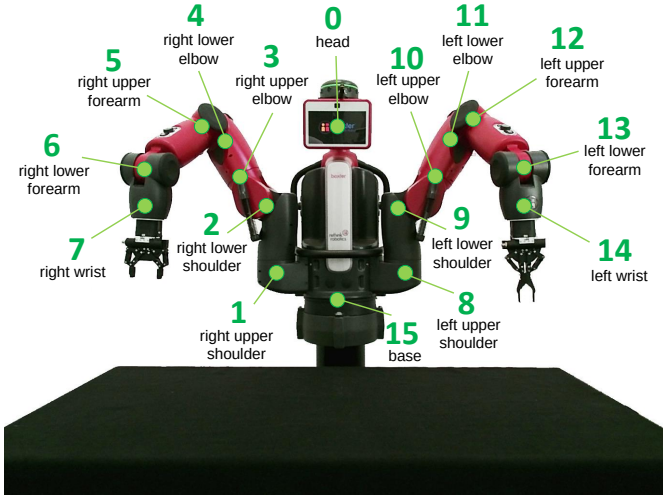


Figure 7.5: Joint locations on the Rethink Baxter robot available in the SimBa^{++} dataset and used in the experimental evaluation.

the camera: we obtain smaller Gaussians for the farthest joints to force the network to focus on those locations that are usually more difficult to predict. On the other hand, the latter space is a bird-eye view of the scene in which the heatmaps H^{uz} are obtained from a quantized portion of the Z plane of size defined as:

$$z = \frac{\bar{Z}_{max} - \bar{Z}_{min}}{\Delta Z} \tag{7.2}$$

where $\bar{Z} = \{\bar{Z}_i \in Z;\ \bar{Z}_{min} <= \bar{Z}_i <= \bar{Z}_{max}\}$.

The proposed method uses an HRNet-32 [227] backbone, specifically exploiting the four stages except for the final convolution. In this way, the output of the backbone consists of a set of visual features that are given as input to the uv and uz branches. Each branch consists of a residual block with 128-dimensional convolutional layers, batch normalization and ReLU activations, and a final convolutional layer that predicts the heatmaps reducing the channel dimension to 16, *i.e.* the number of robot joints, as shown in Figure 7.5.

The output of each branch is finally processed to compute the 3D robot skeleton. For each heatmap H^{uv} in the uv space, we compute the argmax and then the coordinates $\hat{X}\hat{Y}$ multiplying the pixel values of the peak and the inverse of the

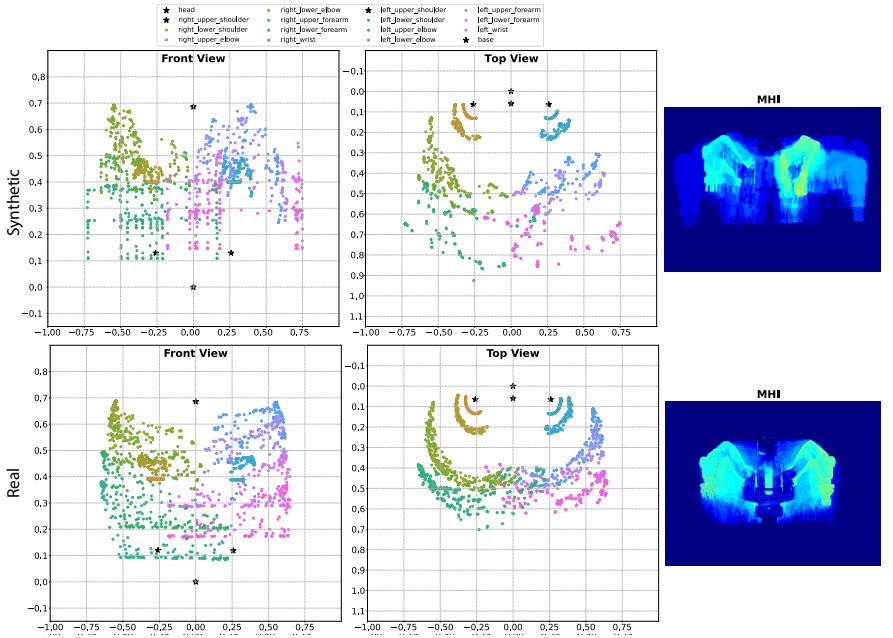


Figure 7.6: Visualization of the joints' movements in synthetic and real sequences contained in the SimBa^{++} dataset with the same camera position. The first two graphs depict each joint location through the sequences from the front and top view of the scene; on the right, a Motion History Image (MHI) [10] of the robot's movements on the same sequences is presented: in this representation, brighter colors denote a high level of motion with respect to blue areas.

camera intrinsics K. On the other hand, for each heatmap H^{uz} in the uz space, we compute the argmax of the z coordinate and convert it into a continuous value in metric space defined as $\hat{Z} = (z_{\max} \cdot \Delta Z) + \bar{Z}_{\min}$. Finally, we multiply the $\hat{X}\hat{Y}$ coordinates from the uv space and the $\hat{Z}$ coordinate from the uz space, obtaining a 3D point $\hat{P}$ for each robot joint.

7.3 Experiments

In this section, we present the training procedure of the proposed method and the metrics used for evaluation.

7.3.1 Dataset

To evaluate the proposed method, we extend the previous version of the SimBa dataset [6] with new real sequences in which the *Rethink Baxter* moves both arms. This new test set represents a challenging scenario since only single-arm movements are available in the synthetic training set. In the following, we give a detailed description of the synthetic and real data that are used respectively for training and testing.

Synthetic data. The synthetic sequences are collected in a virtual environment using Gazebo for physics simulation and ROS for operating the synthetic robot model. The simulation consists of two runs with different random initializations containing 20 different camera poses from which the robot is recorded at 10 fps while performing 10 pick-n-place motions. The recordings are taken from three anchor cameras that are randomly positioned within a sphere of 1m diameter, as depicted in Figure 7.4. The movements cover most of the working space at the front of the robot, as illustrated in Figure 7.6 (top). This guarantees enough variation of the joints' positions for the training phase. The synthetic data contains a total of 400 sequences and 350k RGB-D frames with annotations for 16 joints, pick-n-place locations, and camera positions.

Real data. The real sequences are acquired with the *Microsoft Kinect One* ToF sensor, using ROS for recording the robot movements. The camera is placed in three anchor positions (center, left, right), as depicted in Figure 7.4, so that they are within the space of the synthetic cameras, but not at the same exact location. As an extension to the original dataset [6], we introduce new sequences and divide the dataset into two groups: (i) *single-arm movements* and (ii) *double-arm movements*. The first test set remains the same as in the original dataset, containing 20 sequences at 15 fps from each camera position with pick-n-place motions with either the left or the right arm. The latter is the extension to the original dataset and consists of 10 additional sequences at 15 fps from each camera position with both robot arms moving. Sequences with both robot arms moving are not available in the synthetic dataset, thus incrementing the challenges in the domain shift operation. SimBa^{++} contains a total of 30k real RGB-D frames with

Input	Network	Params (M)	Synthetic test set PCK (%) ↑ 2.5px	Synthetic test set Avg Error (px) ↓	Real test set PCK (%) ↑ 2.5px	Real test set PCK (%) ↑ 5px	Real test set PCK (%) ↑ 10px	Real test set Avg Error (px) ↓
RGB	FPM (MobileNet) [162]	0.16	88.23	1.71	15.97	**55.41**	**92.17**	**10.77**
	FPM (SqueezeNet) [162]	0.36	92.42	1.60	15.36	42.65	81.16	14.17
	SH (1 stack) [175]	14.80	99.44	0.67	0.46	10.66	17.56	68.95
	SH (2 stacks) [175]	26.80	99.41	0.66	0.13	5.59	10.21	95.65
	HRNet-32 [227]	28.50	99.58	0.65	**18.69**	53.51	71.69	22.48
	HRNet-48 [227]	63.60	99.62	0.62	2.67	8.99	17.99	70.87
	TransPose-R-A4 [270]	6.08	99.54	0.63	2.13	15.57	25.63	55.10
	Uniformer-B [136]	53.50	99.18	0.70	11.32	46.43	84.54	11.08
RGB-D	FPM (MobileNet) [162]	0.16	91.26	1.67	1.39	9.58	17.13	84.53
	FPM (SqueezeNet) [162]	0.36	92.17	1.63	**10.23**	**30.88**	**57.94**	**37.84**
	SH (1 stack) [175]	14.80	99.38	0.68	1.01	8.30	16.15	77.75
	SH (2 stacks) [175]	26.80	99.52	0.65	0.41	10.49	14.65	82.88
	HRNet-32 [227]	28.50	99.44	0.67	5.39	14.66	21.39	103.74
	HRNet-48 [227]	63.60	99.66	0.61	2.58	13.09	16.66	118.33
	TransPose-R-A4 [270]	6.08	99.59	0.65	2.21	13.59	25.15	58.92
	Uniformer-B [136]	53.50	99.29	0.68	5.54	22.22	36.66	58.29
DEPTH	FPM (MobileNet) [162]	0.16	88.43	1.75	33.83	72.32	95.51	6.28
	FPM (SqueezeNet) [162]	0.36	91.58	1.62	44.79	87.57	99.59	3.03
	SH (1 stack) [175]	14.80	99.41	0.68	43.85	87.94	92.28	7.35
	SH (2 stacks) [175]	26.80	99.62	0.64	47.99	93.73	98.44	4.02
	HRNet-32 [227]	28.50	99.51	0.67	48.35	88.57	93.31	6.84
	HRNet-48 [227]	63.60	99.65	0.61	50.16	95.37	99.12	2.85
	TransPose-R-A4 [270]	6.08	99.61	0.66	**57.41**	**96.48**	99.15	**2.66**
	Uniformer-B [136]	53.50	99.22	0.70	49.53	94.58	**99.73**	2.68
XYZ	FPM (MobileNet) [162]	0.16	88.37	1.71	37.03	70.29	93.61	6.73
	FPM (SqueezeNet) [162]	0.36	92.40	1.60	49.67	89.64	99.74	2.87
	SH (1 stack) [175]	14.80	99.55	0.66	39.67	91.31	97.15	5.09
	SH (2 stacks) [175]	26.80	99.50	0.69	43.32	90.68	96.29	4.69
	HRNet-32 [227]	28.50	99.54	0.67	50.29	96.96	**99.88**	2.66
	HRNet-48 [227]	63.60	99.61	0.66	49.42	95.23	99.08	2.83
	TransPose-R-A4 [270]	6.08	99.63	0.63	**51.89**	**97.42**	99.84	**2.59**
	Uniformer-B [136]	53.50	99.19	0.69	47.93	94.52	99.61	2.94

Table 7.2: 2D Robot Pose Estimation results (see Section 7.3.4) on SimBa^{++} synthetic and real sequences with single-arm movements

annotations for 16 joints and 3 camera positions.

7.3.2 Model training

The model is trained for 30 epochs on the SimBa^{++} synthetic dataset using $L2$ loss on the heatmaps, batch size 16, *Adam* [113] optimizer, and learning rate $1e^{-3}$ with decay factor 10 at 50% and 75% of training. We follow the same training split of [6] to enable a direct result comparison.

We apply a 3D data augmentation on the point cloud computed from the depth map D_M. In particular, 3D points are rotated of $[-5°, +5°]$ on XY axes and translated of $[-8\text{cm}, +8\text{cm}]$ on XZ axes. We further translate the points on the XZ axes, changing implicitly the camera position. In addition to this geomet-

ric augmentation, we introduce a pixel-wise pepper noise and a random dropout of portions of the depth map, simulating respectively the noise of the real sensor and the holes caused by light on reflective surfaces (*e.g.* metallic objects or screens) that usually produce invalid depth measurements. The pepper noise is introduced for $10 - 15\%$ of the pixels and the random dropout consists of rectangular areas of different dimensions where pixels are set to 0 value.

7.3.3 Metrics

For the quantitative evaluation of the proposed method and the competitors, we used 2D and 3D metrics already introduced in the literature for similar tasks.

For the 2D RPE, we use the Percentage of Correct Keypoints [13] (PCK) metric, *i.e.* the percentage of predicted joints that are within a certain distance threshold with respect to the ground truth. We compute PCK with a confidence threshold of 0.5 and a margin error of 2.5 pixels for the synthetic dataset and $\{2.5, 5, 10\}$ pixels for the real dataset. Moreover, we compute also the average pixel error over all robot joints.

For the 3D RPE, we use the average distance metric (ADD) [261, 131], which is the mean $L2$ distance expressed in centimeters of all 3D robot joints to their ground truth positions. This value (the lower the better) is useful to condense the error related to the translation and rotation in the 3D world. In addition, a *mean average precision* (the higher the better) is used as the accuracy on the ADD using different thresholds of $\{2, 4, 6, 8, 10\}$ centimeters. In this way, results can be evaluated at different distances from the ground truth, giving more interpretability to the actual performance of the methods.

7.3.4 Results

Due to the relative novelty of the 3D Robot Pose Estimation task from depth data, we seize the opportunity to analyze step-by-step the challenges of this research field in combination with the experimental evaluation of the proposed D-SPDH method.

We start our investigation from the 2D domain, focusing on the possibility

to use models and techniques introduced in the 2D human pose estimation literature. Moreover, we assess the challenge of working in a Sim2Real scenario, in which training data differs from the real ones, testing methods on both synthetic and real data. Then, we move our analysis toward the estimation of the 3D pose, starting from the simple solution based on sampling the depth value directly from the depth data, to the regression of the full 3D pose in world coordinates.

2D Robot Pose Estimation. In this task, we compare several literature approaches explicitly developed for the human pose estimation task, ranging from lightweight models [162] to recent Transformer-based architectures [270, 136]. These methods, originally based on the RGB domain, are tested on different input modalities, *i.e.* RGB, RGB-D (channel-wise stacked), depth, and XYZ (as presented in Section 7.2.1) images, belonging to both synthetic and real data.

Experimental results are shown in Table 7.2, in terms of PCK and average pixel error. As expected, good performances are visible on the synthetic data, while the difference between each input modality rises when testing on the more challenging real sequences. In particular, without applying any domain adaptation technique during training, depth and XYZ inputs overcome the RGB and RGB-D modalities, probably since RGB data introduces a significant visual gap between the synthetic and real domains. On the other hand, since the depth and XYZ image representations contain fewer visual details (in particular no details about textures), the trained models tend to better generalize to the real domain, proving that the domain gap on these input types is reduced.

2D to 3D Projection from Depth Data. Once experimentally defined the depth-based inputs for the 2D estimation in the previous analysis, a simple approach to obtain a 3D joint prediction would be to take the 2D predicted coordinates and project them into the 3D space using the camera intrinsics and the corresponding depth value. However, we observe this projection would always lay on the surface of the robot, and therefore be incorrect, as the goal is to predict the central location of the robotic joint. Besides, the magnitude of the error would depend on the robot's model, shape, and pose (*e.g.* self-occlusions).

To mitigate this issue, we still sample the Z value from the depth map, but

		Synthetic test set mAP (%) ↑						Real test set mAP (%) ↑					
	Network	2cm	4cm	6cm	8cm	10cm	ADD (cm) ↓	2cm	4cm	6cm	8cm	10cm	ADD (cm) ↓
DEPTH	FPM (MobileNet)	21.36	62.35	75.99	78.16	80.65	10.29 ±6.18	7.30	24.02	55.07	74.28	81.93	13.49 ±10.93
	FPM (SqueezeNet)	23.10	63.38	75.81	78.15	80.49	10.35 ±6.34	6.73	32.98	67.76	81.14	85.16	8.74 ±5.85
	SH (1 stack)	32.73	68.35	75.17	77.71	80.01	10.45 ±6.29	8.78	36.44	68.66	77.31	79.80	11.37 ±7.72
	SH (2 stacks)	33.48	68.57	75.59	78.18	80.60	9.46 ±5.41	9.62	40.69	72.35	82.42	84.69	8.17 ±5.27
	HRNet-32	33.57	68.56	75.64	78.02	80.38	9.83 ±5.68	9.01	37.31	68.41	78.33	80.02	11.88 ±8.85
	HRNet-48	33.34	68.79	75.64	78.23	80.50	9.46 ±5.44	9.81	39.36	70.74	83.08	85.99	7.19 ±4.32
	TransPose-R-A4	33.17	68.51	75.38	77.96	80.49	9.91 ±5.87	9.27	43.46	75.56	83.44	85.46	7.02 ±4.23
	Uniformer-B	33.95	68.36	75.27	77.79	80.27	9.73 ±5.59	9.79	39.42	73.52	82.67	85.60	7.43 ±4.71
XYZ	FPM (MobileNet)	21.70	62.67	75.49	77.83	80.28	10.86 ±6.57	4.83	24.21	49.39	68.64	78.99	17.32 ±14.15
	FPM (SqueezeNet)	23.38	63.50	75.79	78.12	80.49	10.52 ±6.28	7.67	37.89	72.86	83.04	86.96	7.91 ±5.26
	SH (1 stack)	33.05	68.48	75.43	77.97	80.30	9.67 ±5.55	8.91	39.34	71.61	80.81	84.42	8.99 ±5.63
	SH (2 stacks)	34.05	68.61	75.61	78.13	80.37	9.69 ±5.58	6.79	40.04	72.02	80.42	83.06	8.70 ±5.40
	HRNet-32	33.02	68.24	75.55	77.97	80.37	9.53 ±5.38	8.71	39.55	72.55	83.17	86.98	7.03 ±4.50
	HRNet-48	33.43	68.78	75.60	78.13	80.38	9.47 ±5.39	9.27	40.84	73.52	82.59	85.13	7.04 ±4.20
	TransPose-R-A4	32.59	68.48	75.35	77.88	80.20	9.95 ±5.91	9.02	45.50	76.57	84.62	87.47	7.24 ±5.00
	Uniformer-B	32.91	68.03	75.03	77.65	80.22	10.14 ±5.94	8.78	39.38	73.44	83.10	85.91	7.59 ±4.95

Table 7.3: 3D Robot Pose Estimation results (see Section 7.3.4) on SimBa^{++} synthetic and real sequences with single-arm movements, exploiting 2D to 3D projection from depth data considering the surface-to-joint displacement

		mAP (%) ↑					
Method	Network	2cm	4cm	6cm	8cm	10cm	ADD (cm) ↓
3D regression	ResNet-18 [82]	0.57	9.40	19.99	27.06	44.44	12.20 ±4.12
2D to 3D lifting	[160] *	**13.70**	26.96	37.98	48.40	58.33	10.03 ±3.53
Volumetric heatmaps	[194]	3.35	18.15	42.24	61.60	86.15	7.11 ±0.65
SPDH	TransPose-R-A4	2.58 ±0.77	43.45 ±3.00	73.56 ±1.95	89.15 ±1.24	93.99 ±0.52	5.89 ±1.69
SPDH	Uniformer-B	11.58 ±0.98	40.48 ±1.80	68.90 ±1.97	85.01 ±1.11	91.43 ±0.45	5.61 ±1.79
SPDH	HRNet-32	7.31 ±2.48	48.61 ±5.33	79.88 ±8.76	91.65 ±7.97	96.79 ±3.55	4.65 ±1.00
D-SPDH	HRNet-32	10.62 ±5.69	**53.82** ±9.46	**85.43** ±4.36	**96.17** ±3.82	**98.88** ±1.36	**4.14** ±0.77

* relative joint positions

Table 7.4: 3D Robot Pose Estimation results (see Section 7.3.4) on SimBa^{++} real sequences with single-arm movements

		mAP (%) ↑					
Method	Network	2cm	4cm	6cm	8cm	10cm	ADD (cm) ↓
SPDH	TransPose-R-A4	3.88 ±0.94	43.51 ±2.86	72.47 ±0.47	87.33 ±1.23	93.39 ±0.57	5.89 ±1.80
SPDH	Uniformer-B	**11.58** ±1.03	43.29 ±1.86	71.85 ±1.55	88.00 ±0.84	93.21 ±0.56	5.36 ±1.97
SPDH	HRNet-32	6.71 ±1.38	49.54 ±6.84	80.39 ±8.16	91.75 ±7.84	96.73 ±3.73	4.65 ±0.96
D-SPDH	HRNet-32	10.26 ±4.97	**54.58** ±8.28	**85.04** ±4.76	**95.69** ±3.77	**98.86** ±1.41	**4.14** ±0.69

Table 7.5: 3D Robot Pose Estimation results (see Section 7.3.4) on SimBa^{++} real sequences with double-arm movements

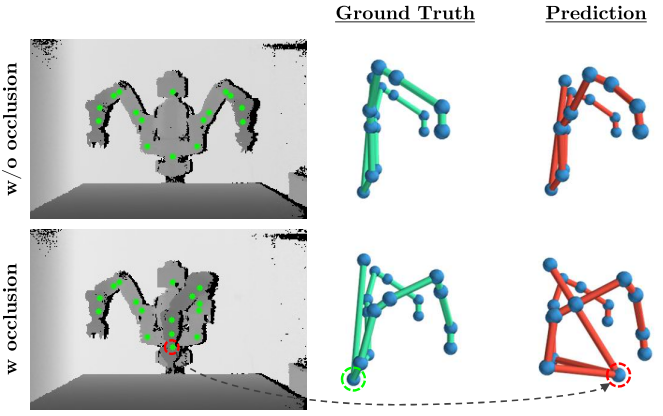


Figure 7.7: Example of the influence of self-occlusions on the predicted 3D pose using the 2D to 3D projection approach. With respect to a frame with all visible robot joints (first row), the occlusion caused by the left arm (second row) results in a large error for the robot base prediction.

introducing also a fixed displacement (computed through the robot model), to reduce the distance between the prediction and the ground truth joint location. In other words, this displacement tries to move the sampled point from the surface to the proper position of the joint inside the robot. In this experiment, we compare the same networks trained on the 2D pose estimation in terms of mAP and ADD.

As shown in Table 7.3, the performance in the 3D domain looks satisfying, but especially at low mAP thresholds, the limitations of this approach arise. Indeed, since using the sampled Z coordinate from the depth produces ADD errors higher than 8 centimeters, the mAP scores at low thresholds become unreliable. In addition, when testing on the real domain, the method is highly influenced by the quality and accuracy of the depth sensor since Z is sampled at a specific point. Another problem is the presence of self-occlusions, which leads to a sampled Z coordinate that is too distant from the inner joint of the robot (Figure 7.7). Moreover, we report the results considering only the projected XY coordinates of the 3D space. As shown in Figure 7.8, it is worth noting that both mAP scores and ADD metric drop significantly when considering the Z values, proving that the sampling from the depth map is not reliable enough for precise 3D joint location.

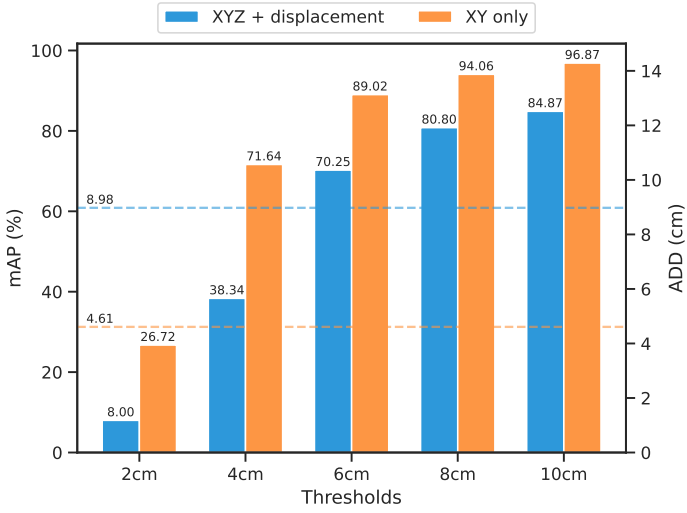


Figure 7.8: Evaluation comparison of the 2D to 3D projection from depth data (see Section 7.3.4) in terms of mAP (barplot) and ADD (horizontal lines), considering XYZ with displacement or XY axes only. The trend is computed as an average over all the networks trained for the 2D pose estimation.

3D Robot Pose Estimation. Given the shortcomings of the 2D to 3D projection analyzed in the previous section, we now consider the 3D robot pose estimation as a direct prediction from the input images. As shown in Table 7.4, we compare the proposed D-SPDH method with the 3D pose estimation literature.

- Direct 3D regression. One of the most common approaches is to regress directly the 3D joint coordinates from an image using CNNs. We empirically select a ResNet-18 [82] backbone that is adapted and trained on the synthetic data to regress the 3D robot joint positions. However, as widely demonstrated for the human pose estimation case [281], this approach does not lead to good results, proving that estimating the 3D absolute pose of an articulated object with respect to the camera is not trivial.

- 2D to 3D lifting. Another widely used approach is predicting the 3D pose starting from a 2D pose. The main feature of this approach is the need for a relative joint position with respect to a specific root (*e.g.* the

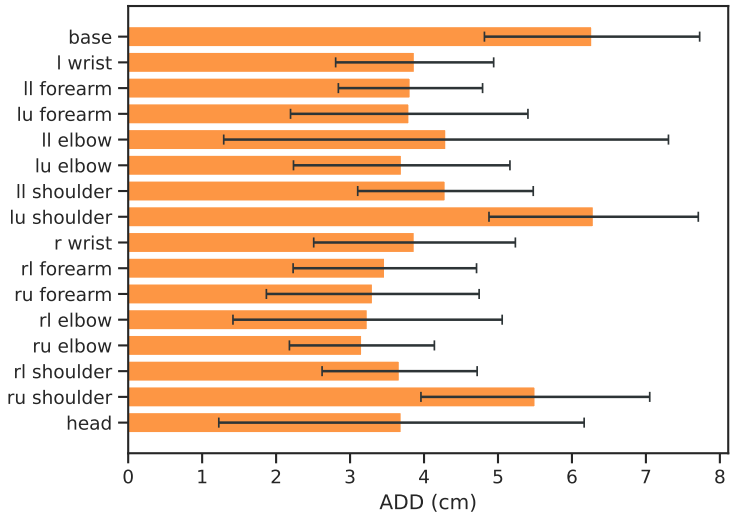


Figure 7.9: Results in terms of ADD metric (mean and std) for each robot joint on the real sequences with double-arm movements (l = left, r = right, ll = left-lower, lu = left-upper, rl = right-lower, ru = right-upper).

robot base), so the absolute 3D pose is computed with a post-processing fitting of the pose with respect to the camera position. For the comparison, we evaluate the method proposed by [160], in which a sequence of different Multi-Layer Perceptron (MLP) networks are trained to predict the 3D joints relying on their 2D positions. From the reported results, we observe that this method is prone to overfitting on the synthetic data obtaining low results on the prediction of the relative 3D pose.

- VOLUMETRIC HEATMAPS. The third solution is based on volumetric heatmaps, a specific representation to encode 3D joint locations in a sampled 3D volume. We train the state-of-the-art method of [194] which outputs a volume of size $d \times w' \times h'$, with $d = 64$, $w' = \frac{w}{4}$, and $h' = \frac{h}{4}$. We observe that this method obtains good results but does not perform as well as all SPDH-based approaches. Moreover, the main problem with volumetric heatmaps is memory usage, especially if the goal is to obtain precise 3D joint locations. Indeed, the memory footprint increases exponentially

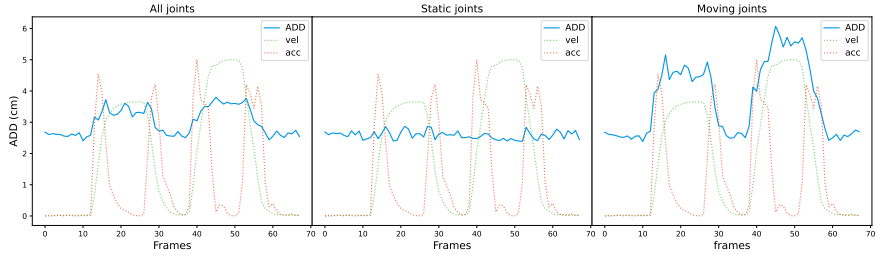


Figure 7.10: Temporal analysis of a real sequence with single-arm movement in terms of ADD and bones length (blue) with respect to velocity (green) and acceleration (red) of joints (see Section 7.4.1).

Method	Network	Limbs Error (cm)
3D regression	ResNet-18 [82]	1.22 ±1.45
2D to 3D lifting	[160]	0.67 ±0.96
Volumetric heatmaps	[194]	2.04 ±1.94
SPDH	TransPose-R-A4	2.15 ±5.44
SPDH	Uniformer-B	1.26 ±1.90
SPDH	HRNet-32	1.00 ±1.23
D-SPDH	HRNet-32	0.84 ±0.73

Table 7.6: Pose plausibility (see Section 7.4.1), *i.e.* the ability of the system to predict realistic joint locations, in terms of robot's limbs mean length error.

with the size of the volumetric heatmap, limiting its resolution and leading to quantization errors. In our experiments, this approach leads to a heavy GPU memory requirement of ≈ 16GB, which is considerably higher than all other methods.

- SPDH vs D-SPDH representation. We take the top three baselines from the 2D pose estimation experiments, *i.e.* HRNet-32, TransPose, and Uniformer, and adapt them to predict the SPDH. Among the baselines, HRNet-32 is the best-performing one on the majority of mAP thresholds and on the ADD metric, so we use it as the backbone for D-SPDH. As stated by the results in Table 7.4, our approach outperforms SPDH by leveraging the double branch architecture and data augmentation. More-

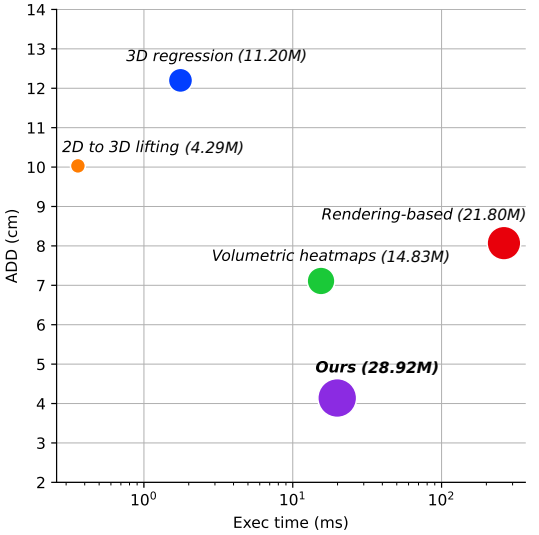


Figure 7.11: Performance comparison (see Section 7.4.2) of different approaches for 3D RPE in terms of execution time (expressed in milliseconds) and ADD metric (low is better). The circle size refers to the number of parameters which is specified next to each method. Thus, small circles close to the plot origin represent the most desirable solution.

over, as shown in Table 7.5, we report the results on the new test set with double-arm movements. This evaluation proves that our method obtains good results even though during training only single-arm movements are seen, outperforming the SPDH approach also in this scenario. Finally, as depicted in Figure 7.9, we analyze the performance of D-SPDH reporting the ADD metric for each robot joint. In this case, it is worth noting that the average error is similar for all the joints and the standard deviation (black line) is relatively low.

7.4 Discussion

7.4.1 Movement-error correlation and pose plausibility

To complete the experimental evaluation, we also explore the effect of the robot's movements on the accuracy of the final prediction. As depicted in Figure 7.10, we analyze the trend of the ADD metric with respect to the joints' movement in

terms of acceleration and velocity. We split the graph into three sections considering the error for (i) all the joints, (ii) the static joints, and (iii) the moving joints. Indeed, the correlation between movement and error is present in most of the sequences suggesting that some actions generate a higher joint error. Moreover, the plots outline that the moving joints contribute the most to the error rate, so the static joints' location, *i.e.* the robot position, is preserved by the network over time. These elements suggest the possibility of including the temporal information in the pipeline to smooth the error caused by the movement of the robot arm.

As a second analysis of the results, we assess the problem of pose plausibility in terms of the robot's physical constraints. In particular, the goal is to prove that the length of the robot's limbs is preserved in the pose prediction, maintaining a realistic robot skeleton. We compute the limbs of the Rethink Baxter robot from its joints, obtaining a total of 15 limbs, where 4 are static. As shown in Table 7.6, D-SPDH obtains competitive results with a low average limb length error, demonstrating that the physical proportions of the robot are preserved while outperforming the competitors in the absolute 3D pose.

7.4.2 Performance analysis

In the last part of our investigation, we analyze the impact of the proposed D-SPDH on the computational requirement. Specifically, we compare our system and the competitors in terms of execution time (expressed in milliseconds) against the ADD error, which well summarizes the performance of the system. For a fair comparison, all experiments are run on the same workstation with an Intel Core i7-7700K and an Nvidia GeForce GTX 1080 Ti, and performance is averaged over multiple input samples. Results of the performance analysis are graphically summarized in Figure 7.11. The two main axes of the figure represent the ADD and the execution time, and the radius of the circles represents the number of parameters. Interestingly, our D-SPDH achieves the lowest error and a very competitive execution time, despite featuring the largest number of parameters. The execution time of D-SPDH enables real-time operation, *i.e.* the

proposed system can achieve $\sim$ 50 frame per second. Unfortunately, solutions based on 2D to 3D lifting and 3D regression present faster execution time, but at the cost of reduced accuracy.

7.5 Takeaways

Several considerations can be made following our experimental evaluation:

- 2D HPE models can be effectively used for the 2D RPE task, not only using RGB as input but also with different input modalities, *e.g.* depth maps as in our case. In other words, there is no need to create new specific backbones for the RPE task, as confirmed by our adoption of HRNet-32 developed for the human pose estimation scenario.

- The proposed D-SPDH double-branch solution represents a major improvement of the SPDH representation, in which each branch is specialized in extracting and predicting a specific heatmap. D-SPDH achieves better accuracy and real-time performance, enabling the development of possible future collision-avoiding systems in the industrial context.

- The Sim2Real scenario simplifies the acquisition of new and accurate labeled data but still represents a challenge for the RPE task. In particular, the performance gap between the use of synthetic and real depth data in input is significant: we observe that this research field is not yet fully explored and needs further investigation. At the time of writing, the possibility (not always practicable) of acquiring and annotating real depth data for training is still an effective solution to improve accuracy.

- To predict 3D robot pose, the 2D to 3D projection is a straightforward technique to retrieve the 3D joint location directly using the Z value available in the depth map. However, this solution is limited in that it always predicts points on the surface of objects and thus is negatively affected in the case of body occlusions.

- The use of a model that directly regresses the 3D world coordinates of the robot joints performs well only on synthetic data, showing that the domain shift negatively influences the final performance and that the regression tends to overfit the training data.

8

3D Pose Nowcasting:
Forecast the Future to Improve the Present

We are increasingly approaching an era in which humans and robots will share different spaces and moments of the day, both in social and working scenarios [198].

Non-invasive camera monitoring combined with specific computer vision algorithms, such as Robot and Human Pose Estimators [281, 131], are key and enabling technologies for safe interaction between humans and robots [40]. For instance, in the Industry 4.0 setting [127], in which the same workplace is shared between workers and cobots [117], the ability to detect poses and avoid collisions is fundamental for safety. Furthermore, recent investigations [254, 255] confirm that – rather than the complete removal of humans – future generations of manufacturing will support the coexistence of humans and cobots, stressing the urgency for new investigations related to physical and social coworker coordination [45]. Another possible application setting is represented by home au-

This Chapter is related to the publication "A. Simoni *et al*., 3D Pose Nowcasting: Forecast the Future to Improve the Present, Under Review" [5]. See the list of Publications on page 151 for more details.

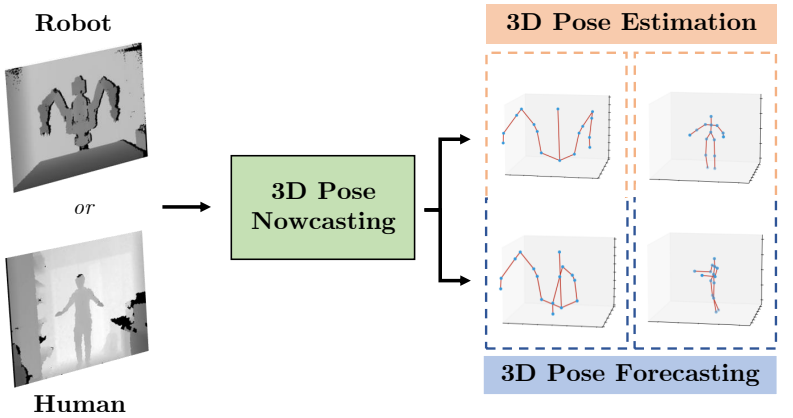


Figure 8.1: Estimating current and future poses through 3D Pose Nowcasting, using depth images as input data, is a fundamental technology for safe interaction between workers and collaborative machines in indoor scenarios, such as the Industry 4.0 setting.

tomation, in which robots can autonomously perform actions but also interact with humans.

In both cases, technologies based on non-invasive sensors that are agnostic with respect to the state of the robot's encoders, are highly desirable. A variety of collision detection systems, especially for the industrial environment, has been proposed but, unfortunately, they often require the use of specific sensors [80], markers [104] or access to the robot's proprietary software [65], which is not always possible.

Therefore, in this paper, we propose a vision-based system able to accurately estimate the 3D poses by learning to forecast the near future as an auxiliary task. In particular, we show how the knowledge about the future at training time improves the model's performance in the present.

Given the similarities with the weather forecasting [28], we refer to this novel paradigm as **3D Pose Nowcasting**, characterized by the following elements: i) the forecasting regards a brief time window (around a few seconds); ii) we are not required to access specific physical models or additional sensors other than the input data (in our case, depth images); iii) forecasting, in addition to enhancing present estimation, is important to raise alarms about imminent and unexpected

events (*e.g.* collisions, hazards).

The proposed method for 3D Pose Nowcasting, outlined in Figure 8.1, is based on addressing the task from two different research fields, *i.e.* 3D Pose Estimation (PE) and 3D Pose Forecasting (PF), jointly learned during training. In particular, the model is trained end-to-end to estimate the 3D pose at the current timestep and the 3D poses at the next future timesteps.

Our approach is based on depth data enabling the development of a vision-based system robust to varying or absent environmental light sources [213], usually common in indoor scenarios such as workplaces. Besides, depth acquisition devices nowadays are inexpensive, yet accurate [274]. Moreover, in the Sim2Real setting [87], the use of depth reduces the domain gap between synthetic and real scenarios [6], thus enabling the usage of large-scale datasets without the time-consuming collecting and labeling procedures required with real data.

From an architectural point of view, PE and PF are tackled through two double-branch CNNs, each specialized in estimating and forecasting joints in 3D world coordinates. The first branch is composed of a backbone originally developed for Human Pose Estimation [13] (HPE), while the second one is obtained by exploiting a motion encoder based on a recurrent neural network, that processes a sequence of past joint locations. The 3D world-coordinate locations of each joint are given in output in real-time, leveraging the recent Semi-Perspective Decoupled Heatmaps (SPDH) [6] as an intermediate representation of poses. To train the model, a double loss is used to optimize both the current pose and the future poses. This is justified by the fact that we want the forecasting loss to influence and improve the estimate at the current timestep.

Summarizing, the main contributions of our paper are:

- We introduce the novel paradigm of 3D Pose Nowcasting, a combination of 3D Pose Estimation and 3D Pose Forecasting in a joint optimization framework. By learning to predict the future, our model improves its pose estimation accuracy in the present.

- We demonstrate the robustness of our approach in the Sim2Real scenario, enabling effective exploitation of synthetic data at training time, and also

domain transfer capabilities from synthetic to real.

- We obtain state-of-the-art performance in estimating the current robot's pose, also providing reliable future predictions. In addition, we show that 3D Pose Nowcasting can be easily exploited for estimating human body joints.

8.1 Related work

Robot Pose Estimation from Depth. Only a limited amount of research addresses the task of pose estimation from depth data. Bohg *et al*. [24] proposed to use a random forest classifier to classify and then group depth maps pixels, obtaining skeleton joints. A similar approach is reported in [256], in which joint angles are directly regressed without any segmentation prior. However, these methods are unable to infer real-world 3D poses, limiting their estimates to joint angles. The large majority of literature works for robot pose estimation are developed for the RGB domain. In general, there are two main approaches: hand-eye calibration-based and rendering-based. In the former, methods are based on fiducial markers (*e.g*. ArUco [63]) placed on the robot's end effector, tracked through multiple cameras. Then, a 3D-2D correspondence problem is solved by relying on forward kinematics or the PnP [134] approach. Unfortunately, these methods are invasive since they require the physical application of markers on the robot, which is not always feasible or practicable. Differently, rendering-based methods [123, 176] use the render&compare paradigm, where an optimization algorithm iteratively refines the pose projected to the image with respect to the camera.

Human Pose Estimation from Depth. Shotton et al. [219] introduced a pioneering approach based on a random forest classifier to classify pixels enabling the segmentation of the human body. The 3D joint candidates are then identified through a weighted density estimator. Using similar features, in [273] the authors proposed to use a regression tree to predict the probability distribution of the direction of a specific joint. Entering the deep learning-based field,

some works introduce the use of NNs in combination with a single depth frame. In [249], a specific memory module referred to as Convolutional Memory Block is introduced, merging the power of CNNs and a memory mechanism used to handle depth data. More recently, [61] introduced a capsule autoencoder network based on fast Variational Bayes capsule routing, focusing on improving viewpoint generalization both on intensity and depth data. Other works are based on point clouds sampled from depth data. In particular, the method described in [278] is based on a point clouds proposal module followed by a 3D pose regression module. Similarly, the same authors in [279] introduced a sequential pose estimation module based on a window of different frames, improving the general performance at the cost of increasing computational complexity. Finally, some literature works have been developed originally for the hand pose estimation task [169, 265, 76] and then adapted to tackle also the human pose estimation task.

Pose Forecasting. Recently, Sampieri *et al*. [211] proposed a graph convolutional neural network to jointly model robot arms and human operators from RGB images. Their goal is to anticipate human-robot collisions. In this work, we follow this research direction and we leverage a trajectory forecasting architecture to improve the current 3D robot pose estimate while also providing information about the future locations of robots and humans. From a general point of view, a large crop of literature has addressed motion forecasting tasks, especially in automotive [130, 159, 99, 153] and human behavior understanding [196, 232, 36, 32, 49]. The task can be framed as an encoder-decoder problem, where past motion is projected into a latent state and then decoded into a plausible future [130, 11]. Interestingly, most approaches formulate the forecasting task as a multimodal prediction task, due to the intrinsic uncertainty of the problem [245, 130, 210, 75]. More recently, several works have addressed the task of forecasting human poses. Compared to the automotive setting, this is a much more complex scenario, since body joints can move erratically and the position of the whole skeleton must be predicted at every timestep. Here, graph-based representations play an important role, since body joints can be naturally represented as connected nodes [200, 140, 8, 222]. Unlike these methods, Mangalam *et al*.

[157] fused 3D skeletons, camera ego-motion and monocular depth estimates to forecast body poses. In a similar way, we propose a depth-based approach for pose estimation and forecasting. Differently from [157], we focus on robot poses and, instead of observing a full sequence of depth and joints, we blend the current depth with an encoding of autoregressively generated past joints.

Depth-based datasets for Pose Estimation and Forecasting. We observe a substantial lack of datasets that can be used for robot pose estimation and forecasting starting from depth data. Recently, four different datasets have been introduced in the literature, but totally based on RGB data. Released in 2019, the CRAVES [289] dataset consists of synthetic and real acquisitions of a single type of robotic arm, for a total of about 5k frames. DREAM [131] and WIM [176], introduced in 2020 and 2022, contain 350k and 140k intensity frames, respectively, depicting different types of robots. One of the most recent datasets is referred to as CHICO [211]. Expressively introduced for collision detection in human-robot interaction, it collects more than 1 million frames acquired with multiple RGB cameras *. Therefore, the only dataset exploitable to test our method is the recent SimBa [6], consisting of more than 370k frames depicting the Rethink Baxter robot performing pick-and-place operations in random locations. This dataset has been acquired in the Sim2Real [87] scenario, *i.e.* the training and testing frames belong to two different domains: synthetic (generated through ROS and Gazebo [116] simulator) and real (acquired through the time-of-flight Microsoft Kinect v2 depth device). SimBa is suitable for our task due to the presence of video sequences, collected at 30 fps.

With regard to the estimation of human poses, we adopt the ITOP dataset [78], which has been used as a benchmark by several prior works [278, 61, 279, 249, 62]. Also in this case, we observe a substantial lack of depth-based datasets in the literature, suitable for our method, for different motivations. Human3.6M [97] dataset contains very low-quality depth images, acquired through the MESA Imaging SR4000 device. The NTU dataset [235], originally developed for the human action recognition task, contains good quality depth data, but unfortunately,

*This dataset presents corrupted 3D joint annotations on images not yet fixed by the authors, making it impossible for us to adopt it.

the human pose annotations are automatically provided through the method described in [219], reducing their accuracy. The mRI dataset [12] appears to be an interesting dataset but depth data have yet to be released, at the time of writing.

8.2 Proposed method

An overview of the proposed framework is depicted in Figure 8.2. It is organized in an encoder-decoder fashion that is split into two input branches and two output branches. The encoder extracts visual and temporal embeddings, while the decoder consists of the *Pose Nowcasting* block, which is made of two SPDH [6] branches dedicated to pose estimation and pose forecasting.

From a formal point of view, the encoder can be viewed as a single frame 2D depth input branch $\Pi(\cdot)$ and a temporal 3D joint recurrent input branch $\Gamma(\cdot)$. For a depth image D and a sequence of $t = 1, ..., M$ poses $P_j^t = [X_j^t, Y_j^t, Z_j^t]$ with $j = 1, ..., J$ 3D joints, two same-size feature maps $\Pi(D)$ and $\Gamma(\mathbf{P})$ are computed and concatenated. The output branches of the nowcasting decoder then independently generate current and future pose predictions.

8.2.1 Depth and past pose input processing

As mentioned, the first input branch is responsible for extracting the features related to the current pose. In this case, the input is represented by a depth image that is converted into an XYZ image, formally defined as follows:

$$I_{XYZ} = \pi(D \cdot K^{-1}) \tag{8.1}$$

where π is the projection in the 3D space, D is the matrix of distances used to create the depth image and K is the projection matrix. This kind of depth representation has been proved to have better generalization capabilities across different domains with respect to common depth images [6]. Being aware of the recent and significant advances in HPE [42], we exploit the well-known HRNet-32 architecture [227], specifically the randomly initialized first four stages without the last convolution, as the backbone to extract pose-related features.

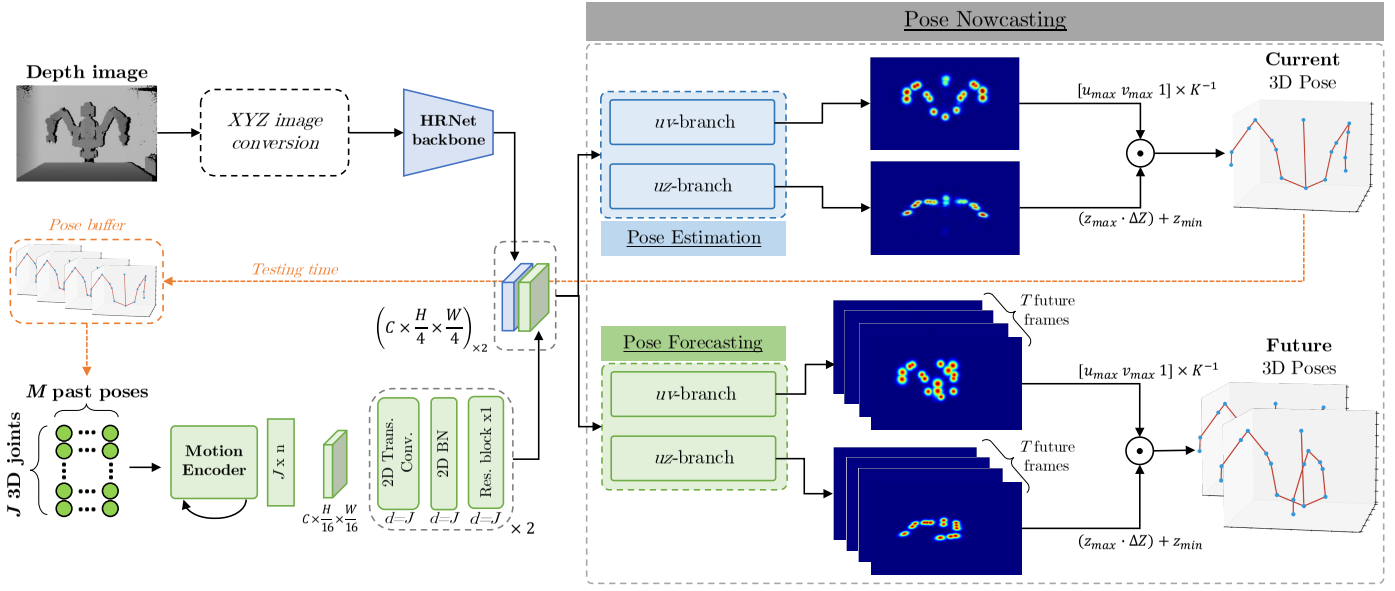


Figure 8.2: Overview of the proposed 3D Pose Nowcasting framework. First, features related to the depth map and the past poses are extracted. These features are then concatenated and fed to two different branches, *i.e.* the Pose Estimation and Pose Forecasting ones. Finally, the framework outputs the current and the near-future 3D poses. For the sake of visualization, heatmaps are stacked channel-wise.

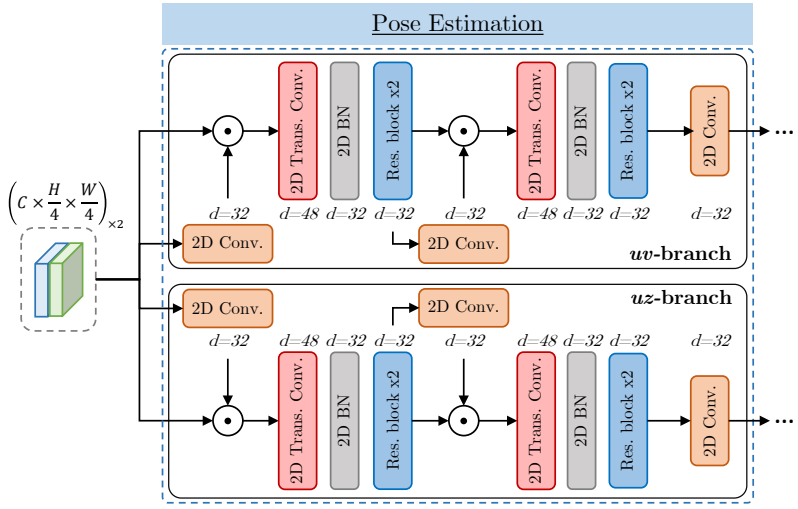


Figure 8.3: Architecture of the Pose Estimation branch. The input is represented by the concatenation of features extracted from depth maps and past joints. Each *uv*/*uz* sub-branch generates the heatmap-based SPDH [6] representation of 3D joint locations.

These features are then concatenated with the ones extracted through the other branch, described as follows.

The second input branch incorporates temporal information obtained from previously estimated 3D joint positions: this information becomes available as soon as a buffer of poses of length M is filled by storing the outputs of the pose estimation block. This branch uses a motion encoder, implemented as a GRU[†], to process higher dimensional embeddings of each pose $P^{\star}_{j}$. Its output is organized into a $C \times \frac{H}{16} \times \frac{W}{16}$ shaped feature map, which is then processed with two layers of residual transposed convolutions with BatchNorm. This architecture is both responsible for processing temporal information stored in previously estimated joints and for adapting the 3D representation to a 2D map that can be fused with the feature map extracted by $\Pi(\cdot)$ from depth images.

[†]Potentially any kind of recurrent architecture such as LSTMs or Transformers could be used. Since our focus is on Nowcasting, we adopt GRUs as commonly done in the trajectory forecasting literature, leaving the investigation of different architectures to future research.

8.2.2 Pose estimation and forecasting branches

Our framework is completed by the nowcasting block with two output branches jointly solving pose estimation and forecasting. Both branches exploit the same SPDH [6] representation, in which the 3D space is decomposed into two bi-dimensional spaces where skeleton joint locations are expressed through heatmaps. In particular, the *uv* space corresponds to the camera image plane (the front view of the acquired scene), while the *uz* space contains the quantized values of the depth dimension, *i.e.* a sort of birds-eye view of the scene with discretized information about the distance of the joints.

In the *pose estimation branch*, the SPDH representation is obtained through the architecture detailed in Figure 8.3, consisting of two residual transposed convolution layers followed by a BatchNorm and ReLU activation function. The estimated pose is represented by a set of $J \times 2$ heatmaps, one pair for each joint in the *uv* and *uz* spaces.

In the *pose forecasting branch*, we adopt a lighter architecture to deal with the multiple SPDH representations that aim to model the near-future joint locations. In particular, we use two 2D convolutional layers, with a size of 32, interspersed with a BatchNorm and ReLU activation function. The forecasted poses are represented as $T \times (J \times 2)$ future heatmaps, where T is the forecasting horizon.

For both output branches, final predictions are obtained as follows: we compute the argmax of the *uv* heatmaps and we multiply the resulting values (u_{max}, v_{max}) with the inverse of the camera intrinsics K^{-1} to obtain the final 3D coordinates. Differently, with *uz* heatmaps, we transform the result of the argmax operation into a continuous value in the metric space multiplying it with the quantization step (ΔZ) computed in the defined depth range (z_{min}, z_{max}).

8.2.3 Losses

To train the model, we directly optimize the *uv*/*uz* heatmaps, before they are converted into 3D coordinates. The system is trained end-to-end optimizing the Mean Squared Error (MSE) loss function $\mathcal{L}$ between generated and ground truth

Input	Model	mAP (%) ↑ 2cm	4cm	6cm	8cm	10cm	ADD (cm) ↓
Depth	ResNet-18 [82]	0.57	9.40	19.99	27.06	44.44	12.20 ±4.12
2D joints	Martinez *et al.* [160] *	13.70	26.96	37.98	48.40	58.33	10.03 ±3.53
Depth	Pavlakos *et al.* [194]	3.35	18.15	42.24	61.60	86.15	7.11 ±0.65
Depth	Simoni *et al.* [6]	6.33	53.75	79.75	93.90	98.12	4.41 ±1.09
Depth	*Ours w/o forecasting*	16.25	57.51	89.81	**99.26**	**99.81**	3.77 ±0.98
Depth + M past poses	*Ours*	**30.68**	**66.90**	**92.69**	98.02	98.38	**3.52** ±1.30

Table 8.1: Robot pose estimation results on SimBa. The proposed framework is tested by taking as input a single depth image ("*Ours w/o forecasting*") or a depth image with the previously predicted 3D joints ("*Ours*"). Method marked with * uses a relative joint representation.

heatmaps:

$$\mathcal{L}_{PE} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} ||H_j^t - \widehat{H}_j^t||_2 \tag{8.2}$$

$$\mathcal{L}_{PF} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \frac{1}{T} \sum_{k=1}^{t+T} ||H_j^{t+k} - \widehat{H}_j^{t+k}||_2 \tag{8.3}$$

$$\mathcal{L} = \mathcal{L}_{PE} + \mathcal{L}_{PF} \tag{8.4}$$

where $\mathcal{L}_{PE}$ is the pose estimation loss between the estimated pose $\widehat{H}_j^t$ and the ground truth H_j^t at the current timestep t; $\mathcal{L}_{PF}$ is the auxiliary pose forecasting loss between the sequence of $k = 1, ..., T$ generated future poses H_j^{t+k} and their corresponding ground truths $\widehat{H}_j^{t+k}$; and $\mathcal{J}$ is the set of skeleton joints in both the uv and uz views. Note that $\widehat{H}_j^t$ is generated by the pose estimation branch whether $\widehat{H}_j^{t+k}$ are generated by the pose forecasting branch.

8.3 Experiments

8.3.1 Datasets

SimBa [6] is a recent dataset specifically acquired for the robot pose estimation task from depth data. It presents unique features such as the presence of synthetic and real depth data, acquired with Gazebo and the Microsoft Kinect v2 sensor. Both domains consist of several sequences of random pick-and-place operations,

Input	Model	Horizon	mAP (%) ↑ 2cm	4cm	6cm	8cm	10cm	ADD (cm) ↓
M past poses	*Linear*	t	0.31	6.02	15.81	25.78	41.23	16.89 ±5.73
M past poses	*Linear*	t+0.5s	0.42	5.54	15.34	25.40	40.58	17.54 ±6.20
M past poses	*Linear*	t+1s	0.29	4.78	14.76	23.44	38.08	19.25 ±6.20
M past poses	*Linear*	t+1.5s	0.32	4.34	14.11	22.76	36.72	19.75 ±6.17
M past poses	*Linear*	t+2s	0.37	3.98	13.72	21.84	35.96	20.04 ±6.10
M past poses	*Ours*	t	5.33	22.77	37.42	57.96	78.05	8.38 ±3.88
M past poses	*Ours*	t+0.5s	4.77	20.74	37.31	55.63	76.53	8.61 ±4.07
M past poses	*Ours*	t+1s	4.41	19.65	35.58	53.16	73.58	9.09 ±4.04
M past poses	*Ours*	t+1.5s	4.12	19.34	33.40	51.65	72.08	9.73 ±4.23
M past poses	*Ours*	t+2s	4.02	18.81	32.56	50.32	70.21	10.41 ±4.59
Depth + *M* past poses	*Ours*	t	**30.68**	**66.90**	**92.69**	**98.02**	**98.38**	**3.52** ±1.30
Depth + *M* past poses	*Ours*	t+0.5s	**31.32**	**66.04**	**91.71**	**97.66**	**98.33**	**3.57** ±1.33
Depth + *M* past poses	*Ours*	t+1s	**28.89**	**59.67**	**84.39**	**91.04**	**92.65**	**4.50** ±2.25
Depth + *M* past poses	*Ours*	t+1.5s	**26.41**	**55.99**	**78.14**	**85.93**	**87.93**	**5.71** ±3.48
Depth + *M* past poses	*Ours*	t+2s	**25.04**	**53.43**	**73.52**	**81.27**	**83.39**	**6.85** ±4.38

8

Table 8.2: Results on both robot pose estimation and forecasting on SimBa. The proposed method is compared to a linear model and our model without the depth-based input branch, while tested in an autoregressive manner.

acquired through randomly placed cameras (left, right and center). The acquired depth data leverages the Time-of-Flight technology and has a spatial resolution of 510×424. This dataset has challenges due to different domains for training and testing (Sim2Real scenario) and different positions of the acquisition devices.

ITOP [78] consists of 20 subjects performing 15 different complex actions, for a total of 50k frames (40k training and 10k testing, as reported in the original paper). Two Structured Light (SL) depth sensors (Asus Xtion Pro) are used to acquire data, one placed in front of the subject, and one placed on the top: in this paper, we focus on the side view, in which human joints are not fully occluded by the head and shoulders of the subject. Annotations consist of 2D and 3D joint coordinates, manually refined to lie inside the body to address human pose estimation from depth data. Unfortunately, not all annotations are valid, thus limiting the length of temporally consistent sequences. The challenges of this dataset are related to the limited quality of depth data, in terms of spatial resolution (320×240), depth accuracy (SL technology [213]), and action complexity, with several occlusions produced during movements.

The proposed system has been trained and tested on the SimBa dataset [6], specifically created for the estimation of robotic joints from depth images. In addition, we demonstrate the generalization capabilities of our approach by testing the system on the ITOP [78] dataset, which has characteristics similar to the context of our interest, albeit applied to human poses.

8.3.2 Metrics

For the 3D pose estimation and forecasting tasks, we exploit standard literature metrics, *i.e.* Average Distance metric (ADD) and mean Average Precision (mAP). The first, that is the L_2 distance expressed in centimeters of all 3D robot joints to their ground truth positions, conveys the error related to the translation and rotation in the 3D world (the lower the better). The second metric is defined as:

$$\text{mAP} = \frac{1}{|N|} \sum_{j \in N} \left(\|\mathbf{v}_j - \widehat{\mathbf{v}}_j\|_2 < \eth \right) \tag{8.5}$$

where N is the number of skeleton joints, $\mathbf{v}_j$ is the predicted joint and $\widehat{\mathbf{v}}_j$ is the ground truth. This metric is intended as the accuracy of the ADD using different thresholds ($\eth = \{2, 4, 6, 8, 10\}$ centimeters in our experiments and it improves the interpretability of results.

8.3.3 Training

The proposed model is trained for 30 epochs by exploiting the MSE loss for the heatmaps produced by both the branches for the current and future poses. We use the Adam optimizer, with an initial learning rate of 10^{-3}, a decay factor of 10^{-1} at 50% and 75% of the training procedure, and a batch size of 16. In all experiments, we use the original dataset splits to train and test the model.

During the training on both datasets, we applied data augmentation on the point clouds computed from the input depth maps. Specifically, 3D points are randomly translated with a maximum range of $[-20\text{cm}, +20\text{cm}]$ and $[-30\text{cm}, +30\text{cm}]$ for XY and Z axes, respectively. Moreover, the points are rotated with a range of $[-5°, +5°]$ for the XZ axes. In terms of visual appearance, we introduce

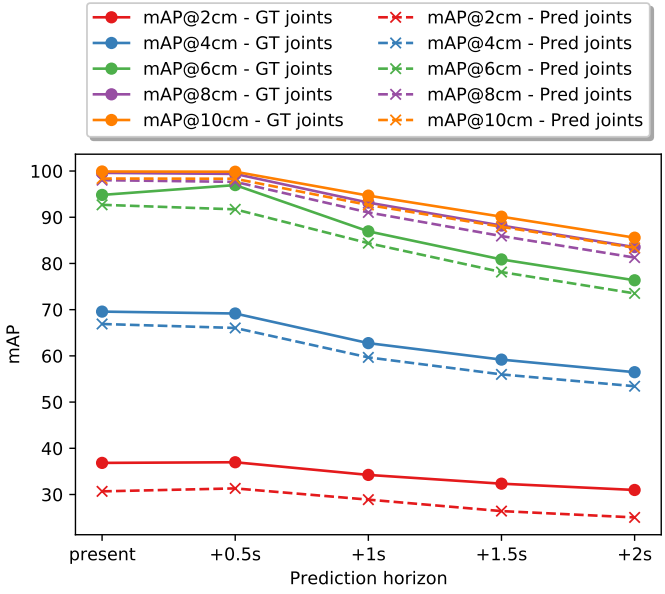


Figure 8.4: Comparison on Simba in terms of mAP using ground truth and predicted 3D joints as input to pose forecasting branch.

a pepper noise on about 15% of the pixels and a random dropout, consisting in setting with the value 0 several small portions of the input image: in this manner, we simulate the presence of depth noise, usually found in real-world depth sensors, and the presence of non-reflecting surfaces (on which the depth value is not valid) in the acquired scene.

8.3.4 Results

We report results on SimBa and ITOP, both with our full pipeline and with a baseline not leveraging the nowcasting paradigm. In all experiments, when the model is optimized to forecast the future, past poses are fed at 10Hz for a duration of 1s. In output instead, we sample poses at 2Hz with a temporal horizon of 2s maximum.

Results on SimBa. Table 8.1 shows results on the SimBa dataset, reporting mean Average Precision (mAP) using different thresholds ($\eth = \{2, 4, 6, 8, 10\}$ cm) as well as ADD. We report results using only the depth image (*Ours w/o fore-*

	mAP (%) at 10cm ↑												
Joint	RF [219]	IEF [31]	VI [78]	RTW [273]	CMB [249]	REN-9x6x6 [76]	A2J [265]	V2V* [169]	DECA-D3 [62]	WSM [278]	AdaPose [279]	*Ours w/o forecasting*	*Ours*
Head	63.8	96.2	98.1	97.8	97.7	98.7	98.5	98.3	93.9	98.1	98.4	98.9	98.6
Neck	86.4	85.2	97.5	95.8	98.5	99.4	99.2	99.1	97.9	99.5	98.7	99.0	99.4
Shoulders	83.3	77.2	96.5	94.1	75.9	96.1	96.2	97.2	95.2	94.7	95.4	97.5	97.6
Elbows	73.2	45.4	73.3	77.9	62.7	74.7	78.9	80.4	84.5	82.8	90.7	84.4	84.4
Hands	51.3	30.9	68.7	70.5	84.4	55.2	68.3	67.3	56.5	69.1	82.1	76.8	77.4
Torso	65.0	84.7	85.6	93.8	96.0	98.7	98.5	98.7	99.0	99.7	99.7	98.7	98.8
Hips	50.8	83.5	72.0	80.3	87.9	91.8	90.8	93.2	97.4	95.7	96.4	87.6	90.4
Knees	65.7	81.8	69.0	68.8	84.4	89.0	90.7	91.8	94.6	91.0	94.4	86.8	89.7
Feet	61.3	80.9	60.8	68.4	83.8	81.1	86.9	87.6	92.0	89.9	92.8	75.3	88.0
Upper body	70.7	61.0	84.0	84.8	80.6	-	-	-	83.0	-	-	90.3	90.4
Lower body	59.3	82.1	67.3	72.5	86.5	-	-	-	95.3	-	-	85.5	90.7
Total body	65.8	71.0	77.4	80.5	83.4	84.9	88.0	88.7	88.7	89.6	**93.4**	88.0	<u>90.6</u>

Table 8.3: Per-joint results on human pose estimation on ITOP side-view test set. The best result is reported in bold, while the second best is underlined. As shown, the proposed framework achieves a significant accuracy on the total body, even though not expressively developed for the HPE task. Method marked with * uses 10 models ensemble.

casting) and with the additional input of past predicted 3D joints (*Ours*). Following [6], we test the same competitors to predict the 3D poses reporting the results in Table 8.1. In particular, we train a ResNet-18 [82] to directly regress 3D coordinates from depth maps. We then evaluate the method proposed in [160], a sequence of MLPs trained to estimate 3D joint coordinates relying on their 2D positions. This approach only provides relative joint locations with respect to a specific root (the robot base). The third competitor, is based on the volumetric heatmap approach [194], a representation for encoding 3D locations in a sampled 3D volume. This approach, in addition to a limited accuracy, leads to a significant video memory occupation of about 16GB, considerably higher than all the other methods (approximately 9 times higher than ours, see Section 8.3.5). Finally, [6] uses the SPDH representation with a standard CNN. Even without the use of the GRU input our approach yields the state of the art on SimBa. Interestingly, when exploiting past joints' locations with a recurrent network and adding the pose forecasting branch, results are improved further especially at low spatial thresholds, almost doubling mAP at the 2cm mark.

Then, we show in Figure 8.4 the results for 3D Pose Forecasting by comparing mAP at different future timestamps. As an upper bound, we report results relying on ground truth past joints' locations. Interestingly, even when autore-

	mAP (%) ↑					
Horizon	**2cm**	**4cm**	**6cm**	**8cm**	**10cm**	**ADD** (cm) ↓
t	10.19	38.76	64.32	79.12	86.57	6.49
t+0.5s	1.94	9.61	21.48	33.91	44.75	17.66
t+1s	1.20	6.72	16.39	27.78	38.56	18.94

Table 8.4: Results on human pose estimation and forecasting on ITOP side-view test set. The model takes as input both depth and past poses.

gressively feeding back estimated joints as input, the performance drop is limited with a maximum difference of 6% for the 2cm threshold. Finally, as shown in Table 8.2, it must be noted that at 1s ADD is roughly 1cm higher than the ADD at the current timestep prediction, making the approach suitable for collision detection. Table 8.2 also shows a comparison between a simple baseline made of a linear regressor trained with SGD and our model with only the encoder-decoder for the forecasting branch. In the latter, the HRNet backbone extracting information from depth images is not used. In both configurations, we obtain much worse results, indicating the non-triviality of the task. In Figure 8.5 (right) we show qualitative results for poses predicted by our model with and without the forecasting branch, highlighting its importance.

Results on ITOP. We show in Table 8.3 our results compared to the state-of-the-art. Overall results for all methods on ITOP are generally worse than on SimBa, due to the fact human movements are more erratic and complex with respect to robot arm motion. Moreover, training is made more challenging by the presence of invalid joints, *i.e.* joints without any manual annotation in the dataset. Nonetheless, on average considering the total body, our approach using a single depth frame is on par with most competing methods. Adding the supervision on future timesteps we rank above all methods except for AdaPose [279], an approach expressively developed for the HPE task (differently from ours) which obtains a slightly higher mAP metric.

Furthermore, it is interesting to notice which joints benefit the most from nowcasting, *i.e.* adding the forecasting branch. In general, the lower body registers a considerable improvement between the two variants of our approach. Hips

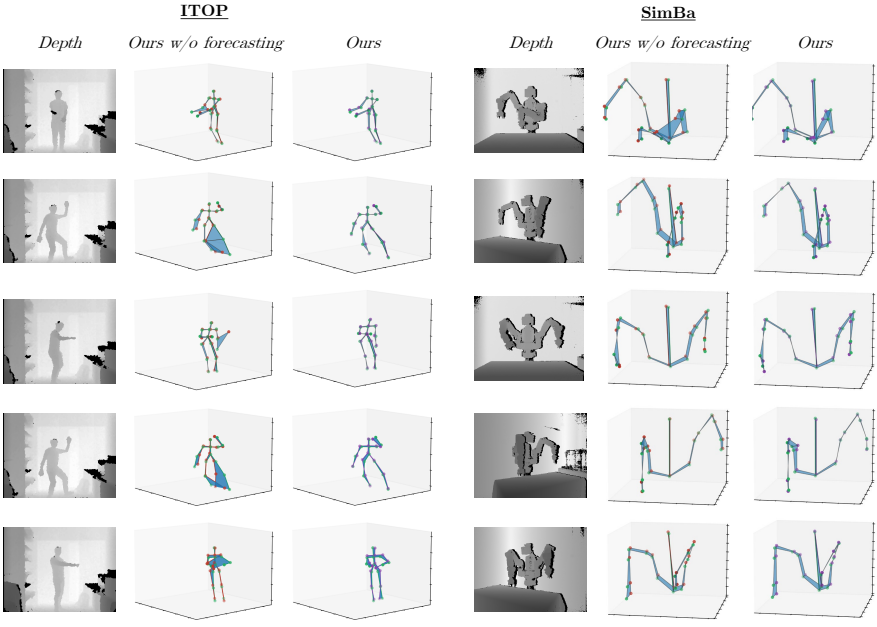


Figure 8.5: Qualitative examples for both ITOP and SimBa datasets where it can be appreciated the improvement in the pose estimation using the proposed approach. Green joints represent the ground truth pose, whereas red and violet represent respectively the poses estimated by our method without future and our full method. Blue regions connect ground truth skeletons and predictions, highlighting errors.

and knees report a gain of approximately +3% mAP, whereas feet even +13% mAP. Given that feet demonstrate greater dynamism in comparison to other body joints, they manifest behavior that is comparatively less erratic than, for instance, hands, wherein the advantageous outcome is less apparent.

In Table 8.4 we show the performance of the framework addressing the forecasting task, which is more challenging in the presence of wide movements performed by humans. These results can be a useful baseline reference for future works that address the forecasting task on ITOP. In Figure 8.5 (left) we show qualitative results on ITOP, comparing the model with the present-only baseline.

8.3.5 Performance Analysis

Our model must be deployable in a work environment, thus must be efficient for safety applications, *e.g*. avoiding collisions and hazards. We measured inference time on an Intel i7 (2.90 GHz) CPU and Nvidia Titan XP GPU. The pose estimation branch alone runs at 20 FPS. Adding the forecasting branch, observing autoregressively generated poses and estimating future ones, the overall inference time is around 11 FPS with a video memory occupation of about 1.8GB. Since we feed to the architecture 1 second of 3D poses sampled at 10Hz and estimated by the model itself, we can run the whole framework in real-time without delays. The reaction time after observing the present frame before estimating the current and future poses is 90ms.

9

Unsupervised Detection
of Dynamic Hand Gestures from Leap Motion Data

Nowadays, *Natural User Interfaces* (NUIs) [146], *i.e.* interfaces in which the interaction relies on free movements of the user body instead of the adoption of mechanical tools (such as mouse and keyboard), represent a powerful solution in the *Human Computer Interaction* (HCI) field to build intuitive and user-friendly applications. Dynamic hand gestures are one of the most-used ways to interact [197], along with voice commands [27, 16] and gaze [126, 112]. In this context, the growing interest in dynamic hand gestures has been supported by the recent introduction of affordable devices that are capable of acquiring both 2D and 3D data. Moreover, some devices can also provide additional semantic information, such as hand keypoints [220] or skeleton joints of the human body [218], with high accuracy and real-time performance. In this work, we assume to acquire information on the user's hand using the *Leap*

This Chapter is related to the publication "D. D'Eusanio *et al.*, Unsupervised Detection of Dynamic Hand Gestures from Leap Motion Data, ICIAP 2021" [1]. See the list of Publications on page 151 for more details.

Motion Controller device [*], an infrared stereo camera capable of estimating the 3D positions of the hand joints in real time through its proprietary software tools.

Unfortunately, NUIs are still limited in real-world applications, due to the lack of effective and robust methods that correctly and quickly detect and classify the gestures. In addition, most of the existing methods, especially the deep learning-based ones, require a large amount of labeled training data, in which the class and the temporal boundaries of each gesture have to be annotated. However, the annotation of the temporal segmentation of each gesture is a time-consuming and error-prone procedure, particularly in the case of long sequences. Moreover, the use of specific datasets collected for a given use case or application requires the user to label new data if the method is applied in a different scenario or if a new gesture is added.

To address these issues, in this paper, we propose a *Transformer*-based [242] model and a specific training approach that, in an unsupervised manner, allows the network to learn to detect the presence of dynamic hand gestures within an input sequence. Indeed, during the training phase, we assume to have access to a large set of single dynamic gestures (*i.e.* a set of sequences, each containing only one gesture) and their gesture class. In this scenario, we propose to exploit the *Connectionist Temporal Classification* (CTC) loss [72]: using this objective, a neural network can learn to temporally segment (*i.e.* detect) an element without explicit segmentation labels, while requiring only sequences of multiple gestures and the associated list of gesture classes. Thus, we apply this loss to "synthetic" gesture sequences, generated by combining several single gestures, and we show that this training approach leads to a learned model that is capable of successfully segmenting and classifying dynamic hand gestures. Moreover, during the testing procedure, we assume to have a continuous data stream that can contain none, one, or multiple dynamic gestures. We show that, even in this challenging case, the network trained with "synthetic" sequences successfully segments and classifies the gestures.

[*] https://www.ultraleap.com/product/leap-motion-controller

9.1 Related work

In the literature, several approaches have been proposed to tackle the detection of hand gestures. Low-level motion parameters such as acceleration, velocity, and trajectory curvature [106] or, in general, body activity [103] can be used to detect the gesture boundaries. In addition, methods based on *Continuous-time Dynamic Programming* (CDP) [178], *Dynamic Time Warping* (DTW) [44], *Hidden Markov Models* (HMMs) [226, 33] and *Conditional Random Fields* (CRF) [203] have been presented. Predicted likelihood scores are compared with a given threshold to detect the gesture boundaries even though, in terms of generalization capabilities, defining a fixed threshold is hard. Indeed, some methods [129, 268] propose to compute an adaptive threshold at inference.

A great variety of methods tackles the classification of detected hand gestures, either static or dynamic. The task is commonly addressed through the use of machine learning-based methods, such as HOG and SVMs [204, 58], HMMs [25, 129], and neural networks. The latter can be split, based on the used model, in recurrent networks, such as RNNs [91, 132], LSTMs [30, 275], and CNNs (2D [35, 52] or 3D [286, 158]). Moreover, recent works propose the combination of multiple features extracted by CNNs [144] or GNNs [277]. Recently, Vaswani *et al.* [242] proposed the Transformer model, an effective self-attention mechanism that has rapidly replaced recurrent methods in many natural language processing and computer vision tasks, including gesture classification [48].

As mentioned, the input of the framework is represented by the 3D hand joints provided by the Leap Motion Controller. In order to improve the usability of application scenarios, many efforts have been conducted by researchers in order to accurately detect 3D hand joints [137] even from single RGB [26] and infrared images [186] and depth maps [272].

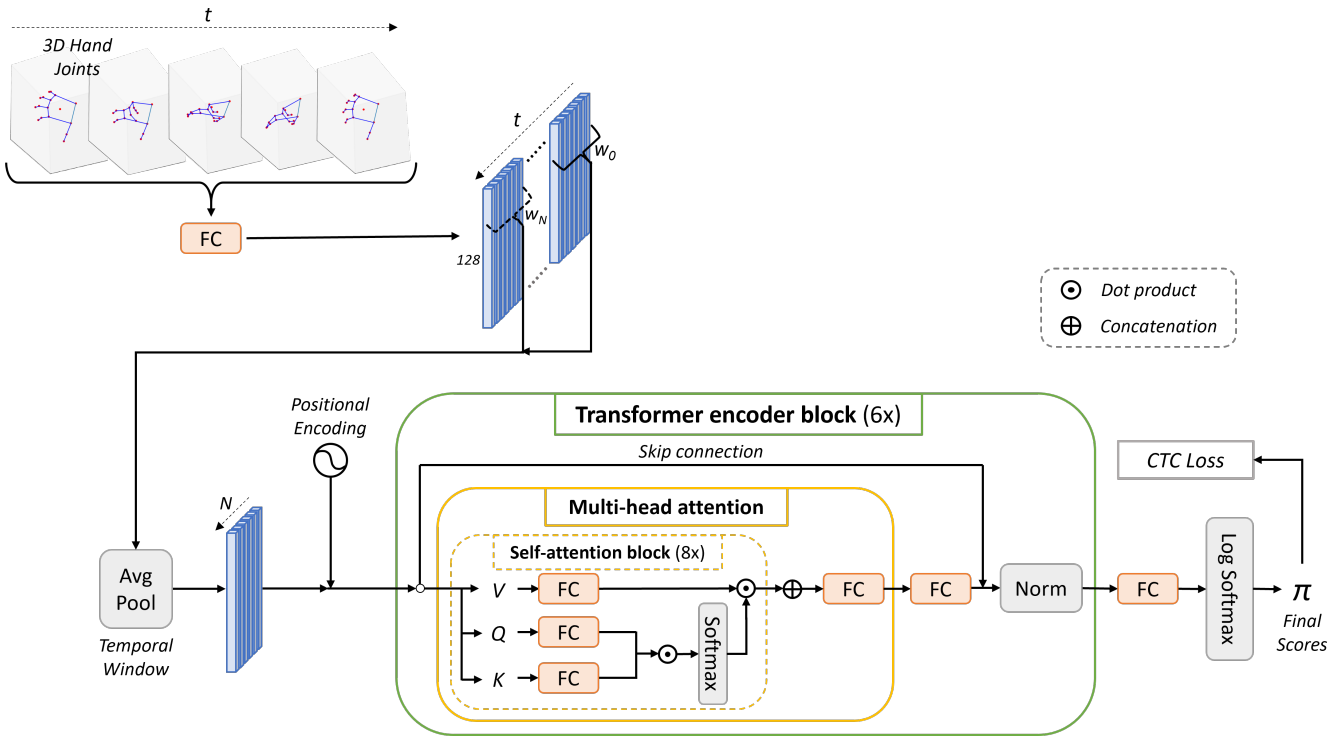


Figure 9.1: Overview of the proposed method. A sequence of 3D hand joint features, provided by the *Leap Motion*, are fed into a fully connected (FC) layer and temporally pooled. Then, a Transformer-based network, based on self-attention blocks (see Section 9.2.1) and optimized with the CTC loss, (see Section 9.2.2), learns to detect and classify dynamic gestures.

9.2 Proposed method

In this section, we present the architectural details of our method and the proposed training procedure that solves the detection and classification tasks in an unsupervised and supervised manner, respectively.

9.2.1 Network architecture

The proposed gesture detection and classification architecture is defined by a Transformer-based model [242] which enables the temporal analysis of the input data. A schematic overview is shown in Figure 9.1.

The input consists of a series of M feature vectors v_j representing the dynamic hand gesture at time j. The first module of the framework is represented by a fully connected layer with a ReLu activation function that remaps each input vector into a 128-d feature. These features are then subdivided along the time dimension into a set of N temporal windows $W_i = \{v_j \,|\, (i-1)\frac{M}{N} < j \leq i\frac{M}{N}\}$ and passed through an average pooling layer that extracts an embedding e of aggregated features. This preliminary feature mapping operations can be defined as:

$$e = \text{AvgPool}(\text{FC}(v)) \tag{9.1}$$

The temporal analysis of the embedding e is then performed by a Transformer-based network which is composed of 6 consecutive encoder blocks E. Each block E contains a set of 8 self-attention blocks followed by two fully connected layers, one with a ReLu activation function and the other one with a normalization layer. An encoder is defined by:

$$E_i(e) = \text{Norm}(e + \text{MultiHead}(e)) \tag{9.2}$$

where MultiHead represents a multi-head attention block with 8 self-attention blocks. In details, a single self-attention operator is represented as:

$$\text{Att}(e) = \text{softmax}\left(\frac{QK}{\sqrt{d_k}}\right) V \tag{9.3}$$

where Q, K, and V are linear projections of e into a 32-dimensional feature space, $d_k = 32$ is the scaling factor corresponding to the size of K. The multi-head attention module is a combination of multiple attention operators:

$$\text{MultiHead} = (\text{Att}_1 \oplus \ldots \oplus \text{Att}_8)\, W^O \tag{9.4}$$

where $\oplus$ is the concatenation operator and W^O is a linear projection from and to a 128-d feature space.

The final part of our network is composed of a linear classifier that predicts $N + 1$ classes (corresponding to N different hand gestures and a "no gesture" label). The output features are then passed through a log-softmax that generates the final scores π.

9.2.2 Proposed training procedure

Our training procedure is built around the *Connectionist Temporal Classification* (CTC) loss function [72] and the generation of synthetic gesture sequences during training.

Synthetic sequence generation. The CTC loss can be used to learn an additional "None" class in an unsupervised manner. However, it requires sequences of multiple gestures split by a "no gesture" action, but obtaining this kind of data is hard. Indeed, these sequences are more complex to collect compared to single-gesture clips and the gestures must not always be performed in the same order. In addition, their annotation is more expensive, due to the need for a temporal segmentation of each gesture. To address these issues, we propose an alternative approach to construct synthetic gesture sequences from single gestures and show that they can be successfully used for training the proposed model. In detail, we randomly combine single-gesture annotated sequences in longer sequences composed of multiple gestures and create a ground truth vector as a list of annotations of the single gestures. Then, without the need for any temporal annotation other than the ordered list of gesture classes, we train our network with the CTC loss.

Connectionist Temporal Classification. We employ the CTC loss func-

tion [72] to optimize the neural model during the training procedure. In particular, we adopt this loss for learning to segment and label gesture sequences from unsegmented data streams. In detail, the model is forced by the CTC loss to predict one of the ground truth gesture classes or an additional label "no gesture" for each input window W_i. The result is a 1-d vector or path π, which maps the input to a sequence of class labels. Moreover, a function $\beta(\pi) = y$ removes the "no gesture" labels and collapses the sequentially-repeated class labels in single instances. For example, giving an input sequence of 7 frames and an output path such as $\pi = [-, 3, 3, -, -, 2, -, 2]$ (where $-$ is the "no gesture" class), the decoded output is $y = [3, 2, 2]$.

Gesture detection and classification. Given the predicted path π, we consider each switch from a "no gesture" label to any other gesture class as a detection of a new gesture. Similarly, the switch from a gesture class to another gesture label or the "no gesture" one is used to identify the end of the current gesture. That is, we use the class change in the prediction from/to the "no gesture" status or from/to another gesture as the beginning and the end of a gesture. Given that the model predicts a gesture class for each temporal window W_i, the gesture classification is simply given by this prediction. It is worth noting that, in this way, both the detection and the classification of the gestures are computed by the same model in a single pass.

9.3 Experiments

In this section, we present the experimental setting in terms of exploited data and model results. Finally, we analyze the performance of the proposed approach.

9.3.1 Experimental setup

As input of the proposed model, we chose to use high-level 3D data, giving the extraction of this information as granted. Indeed, many sensor SDKs and existing neural networks are capable of computing high-level hand features in real time. In detail, we use the location and rotation of the 3D hand joints retrieved by the Leap Motion SDK. In addition, we compute the speed and the acceleration of

Model	Detection metrics			
	Jaccard Index (%)	FPR (%)	Δ Start (s)	Δ End (s)
LSTM [86]	22.93	64.44	0.42 ± 0.47	0.08 ± 0.43
GRU [37]	42.30	45.50	0.51 ± 0.80	0.29 ± 0.77
Ours	**53.42**	**39.25**	$\mathbf{0.40 \pm 0.72}$	$\mathbf{-0.06 \pm 0.69}$

Table 9.1: Experimental results for the hand gesture detection task, obtained on the Briareo dataset [158].

each joint using the joint locations in the previous time steps. An input gesture v_j can be defined as multiple hand joint features g^i:

$$g^i = \left([x, y, z], [\alpha, \beta, \gamma], [\mathbf{s}_x, \mathbf{s}_y, \mathbf{s}_z], [\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z] \right) \tag{9.5}$$

where $[x, y, z]$ are the 3D coordinates of the hand joint i, $[\alpha, \beta, \gamma]$ are its rotation as Euler angles, $[\mathbf{s}_x, \mathbf{s}_y, \mathbf{s}_z]$ and $[\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z]$ are respectively the speed and acceleration vectors computed with regard to the previous two frames. Since the Leap Motion device collects 16 hand joints, each joint information g^i is concatenated to the others obtaining a 192-d feature vector v_j.

During training, we optimize the network parameters using the *Adam* [113] optimizer with learning rate 10^{-4}, weight decay 10^{-4} and dropout with probability $p = 10^{-1}$ within the Transformer block. The model is developed using the PyTorch [189] framework. The code will be published online[†].

Given that we are not bound to predefined multiple-gesture sequences, we test using different numbers of gestures within the synthetic sequences. Similarly, we test multiple values of N by fixing the number of time steps within each window W_i. Thanks to the designed architecture, we do not have to set a fixed number of time steps M per each gesture sequence. In other words, we directly give the gesture sequence to the network regardless of its length. We report the results of our experiments in Section 9.3.3.

[†] https://aimagelab.ing.unimore.it/go/unsupervised-gesture-segmentation

9.3.2 Dataset

We train and evaluate our method on a publicly released dataset, namely *Briareo*, following official train and test splits. Briareo [158] is a hand gesture dataset recorded in a realistic car simulator using multiple devices including the Leap Motion sensor, placed in the tunnel console looking upwards. Briareo was conceived for the automotive context, in which the infrared capabilities of the acquisition devices can be used to develop light-invariant vision-based solutions. Gestures from 12 classes are performed by 40 different subjects (33 males and 7 females). Each gesture is performed by each subject 3 times and the dataset contains an additional recording containing all the gestures in sequence. While the single-gesture sequences are used for training, the all-gesture sequence is used for testing. To evaluate the performance of the proposed approach, we create a small validation set sampling from the training data and manually annotate the all-gesture sequences.

9.3.3 Results

To evaluate our method, we select a set of metrics for both the detection and the classification task. In particular, we use the *Jaccard Index* as the main metric for the detection, expressed as the intersection over union (IoU) between the predicted path π and the ground truth path π^{GT}. This metric rewards a correct detection if there is at least one overlapping frame between the predicted and the ground truth gesture. All frames that are detected as gestures but do not have a correspondence in the ground truth are considered False Positives. In addition, we assess the temporal delay of the predictions as two time intervals Δ, one at the start and one at the end of the gesture, between the predicted and the target gesture. To evaluate the classification task, we use the F1 score, the classification accuracy, and the recall metric.

We report the results on Briareo both for the detection (Table 9.1) and the classification (Table 9.2) tasks. We compare the proposed Transformer-based method with two methods, based on LSTM [86] and GRU [37], and literature competitors. As shown in the left part of the tables, our method, trained on syn-

Model	Classification metrics		
	F1 Score (%)	Accuracy (%)	Recall (%)
Manganaro *et al.* [158]	-	94.40	-
LSTM [86]	89.87	98.98	82.29
GRU [37]	92.36	99.13	86.46
Ours	**95.99**	**99.52**	**92.71**

Table 9.2: Experimental results, split between detection and classification metrics, obtained on the Briareo dataset [158].

thetic gesture sequences, obtains promising results on the detection metrics and outperforms the recurrent-based approaches. In the classification task, all the networks reach similar results, but our method is still able to outperform other approaches. Indeed, the transformer module can elaborate long sequences without recurrent structures, which is beneficial for the overall performance.

9.3.4 Ablation Study

In this section, we assess the performance of our method using different hyper-parameter values.

In Figure 9.2 (left), we evaluate how the number of gestures that compose the training sequence impacts the classification accuracy during the test phase (when the number is fixed to 12). The experiment shows that the network achieves the best classification metrics when using 7 or more gestures and that results are very similar for higher values.

Similarly, we evaluate how the temporal pooling affects the model and report the results in the right part of Figure 9.2. While there is a huge gap between the smallest pooling sizes (*i.e.* 1, 4) and the highest ones (*i.e.* 8, 12, 16), there is not a substantial difference within the highest group, showing that the pooling operation is beneficial to the classification accuracy.

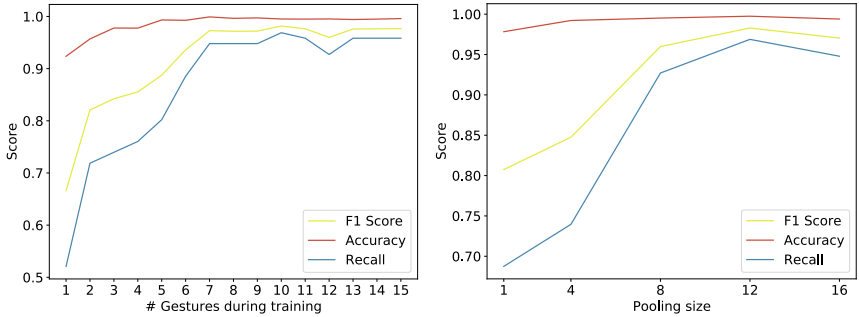


Figure 9.2: F1 score, accuracy, and recall changing the number of gestures in the synthetic training sequences (left) and varying the number of time steps (which are average pooled) within each window W_i (right).

Model	**Parameters** (M)	**GPU** (ms)	**CPU** (ms)	**VRAM** (GB)
LSTM	0.9	**1.67 ± 0.23**	7.12 ± 0.64	0.60
GRU	**0.7**	1.75 ± 0.31	6.58 ± 0.66	0.60
Ours	1.0	4.65 ± 0.39	**4.72 ± 1.13**	**0.59**

Table 9.3: Performance on GPU and CPU. Results are averaged on 100 different runs.

9.3.5 Performance Analysis

We test the computational load of our architecture in terms of the number of parameters and the required GPU VRAM in the testing phase. The performance analysis is conducted on a workstation equipped with a *Intel Core i7-7700K* and a *NVidia GeForce GTX 1080Ti*. Results are reported in Table 9.3. As shown, our method obtains comparable results on both GPU and CPU during inference.

9.3.6 Limitations

In this section, we analyze the difficulties encountered using the framework. First, we note that the training procedure based on the CTC loss function is unstable: the loss value has high variance – from low to high values and vice versa – even within consecutive training steps. Second, due to the unsupervised nature of the proposed method, we obtain a relatively high false positive rate during the testing phase, as shown in Table 9.1: that may require a post-processing phase. Finally,

as reported in Table 9.3, the VRAM required at inference is low, while a great amount may be required for the training of the framework, depending on the length of the sequences and the pooling size W_i (see Section 9.2.1).

10 Conclusions

This dissertation seeks to make a valuable contribution to the 3D Computer Vision research domain by leveraging semantic keypoints for different tasks. This final chapter provides a summary of the contributions made in each study, offering insights into potential future research directions. To wrap up this thesis, we present final remarks and list the activities carried out throughout the Ph.D. program.

10.1 Summary of contributions

Novel view sythesis. In Chapter 3, we presented a novel pipeline for predicting the visual future appearance of an urban scene. We propose a novel approach as an alternative to end-to-end solutions, where human interpretable information is included in the loop and every actor is modeled independently. Existing state-of-the-art methods or the user can both be sources for that information. Furthermore, the final visual output is conditioned onto that by design. We demonstrate the performance superiority of our pipeline with respect to traditional end-to-end baselines through an extensive experimental section. More-

over, we visually illustrate how our method can generate diverse realistic futures starting from the same input by varying the provided interpretable information. With reliable trajectory predictions, our method can be used as a synthetic data augmentation of datasets for object detection or tracking.

VEHICLE CLASSIFICATION. In Chapter 4, we show how visual and pose features can be merged in the same framework in order to improve the car model classification task. Specifically, we leverage the ResNext-101 architecture, for the visual part, and Stacked-Hourglass, for the car keypoint localization, to design a combined architecture. Experimental results confirm the accuracy and the feasibility of the presented method for real-world applications. Moreover, the performance analysis confirms the limited inference time and the low amount of video memory required to run the system.

VEHICLE RECONSTRUCTION. In Chapter 5, we show how the 3D mesh reconstruction of objects can be learned jointly on multiple classes using only foreground masks and coarse camera poses as supervision. The proposed approach discerns between different object categories and learns meaningful category-level meanshapes, which were initialized as spheres, in an unsupervised manner. In addition, a novel approach to predict the instance-specific deformation at the vertex level is presented. The network produces smooth deformations and is independent of the number of mesh vertices, allowing the dynamic subdivision of the mesh during training. Quantitative and qualitative results on two public datasets show the effectiveness of the proposed method.

3D POSE ESTIMATION THROUGH HEATMAPS. In Chapter 6, we present a depth-based 3D Robot Pose Estimation approach that can be trained on fully synthetic data and evaluated on real data with promising results. Leveraging from a novel heatmap-based output representation, namely *Semi-Perspective Decoupled Heatmaps* (SPDH), the proposed method takes an XYZ image obtained from a depth map as input and predicts two bi-dimensional heatmaps that are then converted to 3D joint locations. We also present and publicly release the *SimBa* dataset, that we use to evaluate the proposed system in both synthetic and real environments. A thorough experimental section compares the proposed method to al-

ternative approaches derived from the HPE domain, confirming its promising performance.

3D ROBOT POSE ESTIMATION. In Chapter 7, we have proposed the D-SPDH architecture to estimate the 3D pose of a robot, investigating the challenging Sim2Real scenario, *i.e.* relying only on synthetic depth data as input during the training while testing the method on real sequences. We also have introduced SimBa^{++}, an extended version of the SimBa dataset including new challenging and real sequences with double-arm movements, on which the proposed system is tested and compared with literature competitors. The experimental evaluation confirms the suitability of the presented approach, in terms of both accuracy and real-time performance.

3D POSE NOWCASTING. In Chapter 8, we introduced the paradigm of 3D Pose Nowcasting, using depth data. The proposed framework jointly optimizes pose estimation and forecasting, exploiting two branches and the SPDH intermediate representation. We obtain state-of-the-art results in predicting current and near-future robot poses. The framework is also able to work with humans, achieving performance comparable with the current literature competitors on ITOP.

HAND GESTURE RECOGNITION. In Chapter 9, we propose a method to detect and classify dynamic hand gestures. The model, based on the Transformer architecture, is trained in an unsupervised manner for the temporal segmentation task and in a traditional supervised setting for the classification one. The CTC loss is exploited to learn the temporal segmentation without explicit labels. Experimental results, obtained on the Briareo dataset, reveal that the proposed method achieves satisfying accuracy scores with limited computational load.

10.2 FUTURE DIRECTIONS

3D VEHICLE RECONSTRUCTION. Given the remarkable achievements demonstrated by neural radiance fields (NeRFs) [166, 19, 172, 20, 111] in 3D reconstruction, the future of 3D object reconstruction involves leveraging this approach to generate authentic synthetic representations of vehicles. This differs from the

studies conducted in this thesis, as it introduces additional challenges, including accommodating multiple viewpoints of the scene and achieving precise camera pose estimation.

In future work, we aim to explore the application of NeRF-based methodologies to obtain more photorealistic 3D models of vehicles within real-world scenarios. Notably, this goal involves addressing the complexities associated with capturing various perspectives of the scene and accurately estimating the camera poses. Additionally, we propose utilizing semantic keypoints specific to vehicles to enhance the initial camera pose estimation typically computed by conventional structure-from-motion algorithms [215]. These semantic keypoints provide consistent visual information across multiple viewpoints, serving as a global constraint to refine and improve the accuracy of the camera poses.

Within the automotive context, a notable challenge arises from handling light reflections on surfaces, such as car windows and bodies. To tackle this issue, we propose enhancing NeRF's capability by making it aware of the material composition of each component of a vehicle. In this way, NeRF can treat each part differently based on the radiance field interacting with a particular material. By incorporating this material-awareness aspect, we aim for a more realistic representation of vehicles, especially in scenarios where reflections and material properties play a crucial role in the visual fidelity of the reconstructed 3D model.

3D POSE ESTIMATION. In the robotics world, we remark on the need for a dataset featuring 3D annotations of both human and robot skeletons engaged in collaborative activities within the same environment. Acknowledging the impracticality of acquiring real-world data to cover all the potential scenarios, the adoption of a high-fidelity simulator with robust physics emerges as a viable solution. This approach facilitates the gathering of extensive data for training a pose estimation algorithm, aligning with the methodology proposed in this thesis.

For human-centric scenarios, our focus is on leveraging a diffusion-based technique that has recently demonstrated state-of-the-art performance in single-person contexts. Current literature [68, 217, 38] addresses this challenge through the utilization of 2D-to-3D lifting methods [195, 282, 139], treating them as denoiser networks. These networks take a noisy sequence of 3D poses, conditioned

by a sequence of corresponding 2D poses, and output a denoised 3D pose for the central frame. The objective is to extend this methodology to scenarios involving multiple individuals within a scene, introducing complexities such as substantial occlusions and tracking.

To enhance the robustness of the approach, we propose incorporating visual context features into the diffusion model. These features are typically overlooked by 2D-to-3D lifting approaches that solely consider the predicted 2D pose. Additionally, alongside context features, our strategy involves extracting social embeddings for each person present in the scene. This incorporation aims to enhance the awareness of potential occlusions between individuals while they are interacting with each other or with the environment.

10.3 Final remarks

Some of the works presented in this thesis have been successfully published in international conferences and journals. In particular, the work on 3D vehicle reconstruction in Chapter 5 has been published in the International Conference on 3D Vision, while the paper on the heatmap-based representation for 3D pose estimation has been published in IEEE Robotics and Automation Letters and presented at IEEE/RJS International Conference on Intelligent Robots and Systems (IROS 2022). Following the studies presented in this dissertation, we hope that the 3D Computer Vision community will find them useful for their future research.

10.4 Ph.D. Activities

This final section presents a list of the main activities carried out by the candidate during the Ph.D. program in Information and Communication Technologies.

10.4.1 Teaching Activities

2021: Laboratory Lecturer of "3D Computer Vision" for "School in AI: Deep Learning, Vision and Language for Industry";

10.4.2 Conference Attendances

10 - 15 January 2021: IAPR 25th International Conference on Pattern Recognition (ICPR), 2020, *Milan, Italy (Remote)*;

8 February - 10 February 2021: International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), 2021, *Vienna, Austria (Remote)*;

1 - 3 December 2021: International Conference on 3D Vision (3DV), 2021, *London, UK (Remote)*.

23 - 27 May 2022: International Conference on Image Analysis and Processing (ICIAP), 2022, *Lecce, Italy*.

23 - 27 October 2022: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, *Kyoto, Japan*.

2 - 6 October 2023: IEEE/CVF International Conference on Computer Vision (ICCV), 2023, *Paris, France*.

10.4.3 Seminars and Workshops

November 2020: Attendance at "Deep Scene Perception without Labeled Data" seminar, speaker: Prof. Luigi Di Stefano;

February 2021: Attendance at "The Machine Learning of Time: Past and Future" seminar, speaker: Prof. Efstratios Gavves;

March 2021: Attendance at "There will be Artificial Emotional Intelligence" seminar, speaker: Prof. Björn Schuller;

March 2021: Attendance at "Towards Robust End-to-End Driving" seminar, speaker: Prof. Andreas Geiger;

June 2021: Attendance at "Research in videogames: use of deep learning for saliency estimation and cheating prevention" seminar, speaker: Dr. Iuri Frosio;

October 2021: Attendance at "Safe, Interaction-Aware Decision Making and Control for Robot Autonomy" seminar, speaker: Prof. Marco Pavone;

October 2021: Attendance at "Brain Inspired Computing Workshop: from Neuroscience to Artificial Intelligence";

April 2022: Attendance at "Research challenges in Leonardo Labs: applied Deep Learning in the Industry" seminar, speaker: Prof. Alessandro Nicolosi;

April 2022: Attendance at "Domain Adaptation & Generalization" seminar, speakers: Prof. Vittorio Murino and Dott. Pietro Morerio;

November 2022: Attendance at "Digital Humanities and Artificial Intelligence for humans in today society" seminar, speakers: Prof. Rita Cucchiara;

November 2022: Attendance at "Graph Signal Processing for Machine Learning: Challenges and Use cases" seminar, speakers: Prof. Laura Toni;

December 2022: Attendance at "From Handcrafted to End-to-End Learning, and Back: a Journey far Multi-Object Tracking" seminar, speakers: Prof. Laura Leal-Taixé;

December 2022: Attendance at "3D Computer Vision for animals" seminar, speakers: Prof. Silvia Zuffi;

10.4.4 Schools

19 - 23 July 2021: Attendance and completion of the "4th Advanced Course on Data Science and Machine Learning - ACDL 2021" summer school.

9 - 15 July 2022: Attendance and completion of the "International Computer Vision Summer School - ICVSS 2022" summer school.

18 - 22 September 2023: Attendance and completion of the "ELLIS Summer School on Large-Scal AI for Research and Industry 2023" summer school.

10.4.5 Reviewing Service

Conferences.

IEEE International Conference on Robotics and Automation (ICRA);

IEEE/CVF Winter Conference on Applications of Computer Vision (WACV);

IEEE International Conference on Pattern Recognition (ICPR);

Journals.

IEEE Robotics and Automation Letters (RA-L);

Pattern Recognition;

Workshops.

Towards a Complete Analysis of People: From Face and Body to Clothes (T-CAP);

International Workshop and Challenge on People Analysis (WCPA);

List of Publications

[1] D'Eusanio, A., Pini, S., Borghi, G., Simoni, A., and Vezzani, R. (2022). Unsupervised detection of dynamic hand gestures from leap motion data. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 414–424. Springer.

[2] Simoni, A., Bergamini, L., Palazzi, A., Calderara, S., and Cucchiara, R. (2021a). Future urban scenes generation through vehicles synthesis. In *Proceedings of the International Conference on Pattern Recognition*, pages 4552–4559. IEEE.

[3] Simoni, A., Borghi, G., Garattoni, L., Francesca, G., and Vezzani, R. (2023a). D-spdh: Improving 3d robot pose estimation in sim2real scenario via depth data. *Under Review*.

[4] Simoni, A., D'Eusanio, A., Pini, S., Borghi, G., Vezzani, R., et al. (2021b). Improving car model classification through vehicle keypoint localization. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 354–361.

[5] Simoni, A., Marchetti, F., Borghi, G., Becattini, F., Seidenari, L., Vezzani, R., and Del Bimbo, A. (2023b). Robot pose nowcasting: Forecast the future to improve the present. *Under Review*.

[6] Simoni, A., Pini, S., Borghi, G., and Vezzani, R. (2022). Semi-perspective decoupled heatmaps for 3d robot pose estimation from depth maps. *IEEE Robotics and Automation Letters*, 7(4):11569–11576.

[7] Simoni, A., Pini, S., Vezzani, R., and Cucchiara, R. (2021c). Multi-category mesh reconstruction from image collections. In *Proceedings of the International Conference on 3D Vision*, pages 1321–1330. IEEE.

Bibliography

[8] Adeli, V., Ehsanpour, M., Reid, I., Niebles, J. C., Savarese, S., Adeli, E., and Rezatofighi, H. (2021). Tripod: Human trajectory and pose dynamics forecasting in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13390–13400.

[9] Afifi, A. J., Hellwich, O., and Soomro, T. A. (2018). Simultaneous object classification and viewpoint estimation using deep multi-task convolutional neural network. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 177–184.

[10] Ahad, M., Rahman, A., Tan, J., Kim, H., and Ishikawa, S. (2012). Motion history image: its variants and applications. *Machine Vision and Applications*, 23(2):255–281.

[11] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 961–971.

[12] An, S., Li, Y., and Ogras, U. (2022). mri: Multi-modal 3d human pose estimation dataset using mmwave, rgb-d, and inertial sensors. *Advances in Neural Information Processing Systems*, 35:27414–27426.

[13] Andriluka, M., Pishchulin, L., Gehler, P., and Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3686–3693.

[14] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning*, pages 214, 223.

[15] Aumentado-Armstrong, T., Levinshtein, A., Tsogkas, S., Derpanis, K. G., and Jepson, A. D. (2020). Cycle-consistent generative rendering for 2d-3d

modality translation. In *Proceedings of the International Conference on 3D Vision*, pages 230–240.

[16] Bala, A., Kumar, A., and Birla, N. (2010). Voice command recognition system based on mfcc and dtw. *International Journal of Engineering Science and Technology*, 2(12):7335–7342.

[17] Ballotta, D., Borghi, G., Vezzani, R., and Cucchiara, R. (2018). Head detection with depth images in the wild. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 56–63.

[18] Bansal, M., Krizhevsky, A., and Ogale, A. (2019). Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems*.

[19] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479.

[20] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2023). Zip-nerf: Anti-aliased grid-based neural radiance fields. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705.

[21] Bicchi, A. and Kumar, V. (2000). Robotic grasping and contact: A review. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 348–353. IEEE.

[22] Billard, A. and Kragic, D. (2019). Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414.

[23] Black, M. J., Patel, P., Tesch, J., and Yang, J. (2023). Bedlam: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8726–8737.

[24] Bohg, J., Romero, J., Herzog, A., and Schaal, S. (2014). Robot arm pose estimation through pixel-wise part classification. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3143–3150.

[25] Borghi, G., Vezzani, R., and Cucchiara, R. (2016). Fast gesture recognition with multiple stream discrete hmms on 3d skeletons. In *Proceedings of the International Conference on Pattern Recognition*, pages 997–1002. IEEE.

[26] Boukhayma, A., Bem, R. d., and Torr, P. H. (2019). 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10843–10852.

[27] Brenon, A., Portet, F., and Vacher, M. (2016). Preliminary study of adaptive decision-making system for vocal command in smart home. In *Proceedings of the International Conference on Intelligent Environments*, pages 218–221. IEEE.

[28] Browning, K. and Collier, C. (1989). Nowcasting of precipitation systems. *Reviews of Geophysics*, 27(3):345–370.

[29] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7291–7299.

[30] Caputo, F. M., Burato, S., Pavan, G., Voillemin, T., Wannous, H., Vandeborre, J.-P., Maghoumi, M., Taranta, E., Razmjoo, A., LaViola Jr, J., et al. (2019). Online gesture recognition. In *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association.

[31] Carreira, J., Agrawal, P., Fragkiadaki, K., and Malik, J. (2016). Human pose estimation with iterative error feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4733–4742.

[32] Chao, Y.-W., Yang, J., Price, B., Cohen, S., and Deng, J. (2017). Forecasting human dynamics from static images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 548–556.

[33] Chen, F.-S., Fu, C.-M., and Huang, C.-L. (2003). Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745–758.

[34] Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., and Fidler, S. (2019). Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in Neural Information Processing Systems*, 32:9609–9619.

[35] Cheng, W., Sun, Y., Li, G., Jiang, G., and Liu, H. (2019). Jointly network: a network based on cnn and rbm for gesture recognition. *Neural Computing and Applications*, 31(1):309–323.

[36] Chiu, H.-k., Adeli, E., Wang, B., Huang, D.-A., and Niebles, J. C. (2019). Action-agnostic human pose forecasting. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1423–1432. IEEE.

[37] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[38] Choi, J., Shim, D., and Kim, H. J. (2023). Diffupose: Monocular 3d human pose estimation via denoising diffusion probabilistic model. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3773–3780.

[39] Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S. (2016). 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision*. Springer.

[40] Colgate, E., Bicchi, A., Peshkin, M. A., and Colgate, J. E. (2008). Safety for physical human-robot interaction. In *Springer Handbook of Robotics*, pages 1335–1348. Springer.

[41] Coumans, E. and Bai, Y. (2016–2019). Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

[42] Dang, Q., Yin, J., Wang, B., and Zheng, W. (2019). Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676.

[43] Darcis, M., Swinkels, W., Güzel, A. E., and Claesen, L. (2018). Poselab: A levenberg-marquardt based prototyping environment for camera pose estimation. In *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*, pages 1–6. IEEE.

[44] Darrell, T. J., Essa, I. A., and Pentland, A. P. (1996). Task-specific gesture analysis in real-time using interpolated views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1236–1242.

[45] Dautenhahn, K. and Saunders, J. (2011). *New frontiers in human robot interaction*, volume 2. John Benjamins Publishing.

[46] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[47] Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494.

[48] D'Eusanio, A., Simoni, A., Pini, S., Borghi, G., Vezzani, R., and Cucchiara, R. (2020). A transformer-based network for dynamic hand gesture recognition. In *Proceedings of the International Conference on 3D Vision*.

[49] Diller, C., Funkhouser, T., and Dai, A. (2022). Forecasting characteristic 3d poses of human actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15914–15923.

[50] Dinesh Reddy, N., Vo, M., and Narasimhan, S. G. (2018). Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1906–1915.

[51] D'Eusanio, A., Pini, S., Borghi, G., Vezzani, R., and Cucchiara, R. (2019). Manual annotations on depth maps for human pose estimation. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 233–244.

[52] D'Eusanio, A., Simoni, A., Pini, S., Borghi, G., Vezzani, R., and Cucchiara, R. (2020). Multimodal hand gesture classification for the human–car interaction. In *Informatics*, volume 7, page 31. Multidisciplinary Digital Publishing Institute.

[53] Esser, P., Sutter, E., and Ommer, B. (2018). A variational u-net for conditional appearance and shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8857–8866.

[54] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*.

[55] Fabbri, M., Brasó, G., Maugeri, G., Cetintas, O., Gasparini, R., Ošep, A., Calderara, S., Leal-Taixé, L., and Cucchiara, R. (2021). Motsynth: How can synthetic data help pedestrian detection and tracking? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10849–10859.

[56] Fabbri, M., Lanzi, F., Calderara, S., Alletto, S., and Cucchiara, R. (2020). Compressed volumetric heatmaps for multi-person 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7204–7213.

[57] Fabbri, M., Lanzi, F., Calderara, S., Palazzi, A., Vezzani, R., and Cucchiara, R. (2018). Learning to detect and track visible and occluded body joints in a virtual world. In *Proceedings of the European Conference on Computer Vision*, pages 430–446.

[58] Feng, K.-p. and Yuan, F. (2013). Static hand gesture recognition based on hog characters and support vector machines. In *Proceedings of the International Symposium on Instrumentation and Measurement, Sensor Network and Automation*, pages 936–938. IEEE.

[59] Fiala, M. (2005). Artag, a fiducial marker system using digital techniques. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, volume 2, pages 590–596. IEEE.

[60] Frigieri, E., Borghi, G., Vezzani, R., and Cucchiara, R. (2017). Fast and accurate facial landmark localization in depth images for in-car applications. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 539–549. Springer.

[61] Garau, N., Bisagno, N., Bródka, P., and Conci, N. (2021). Deca: Deep viewpoint-equivariant human pose estimation using capsule autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11677–11686.

[62] Garau, N. and Conci, N. (2023). Capsules as viewpoint learners for human pose estimation. *arXiv preprint arXiv:2302.06194*.

[63] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292.

[64] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE.

[65] Geravand, M., Flacco, F., and De Luca, A. (2013). Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4000–4007.

[66] Girdhar, R., Fouhey, D. F., Rodriguez, M., and Gupta, A. (2016). Learning a predictable and generative vector representation for objects. In *Proceedings of the European Conference on Computer Vision*, pages 484–499. Springer.

[67] Goel, S., Kanazawa, A., and Malik, J. (2020). Shape and viewpoint without keypoints. In *Proceedings of the European Conference on Computer Vision*.

[68] Gong, J., Foo, L. G., Fan, Z., Ke, Q., Rahmani, H., and Liu, J. (2023). Diffpose: Toward more reliable 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13041–13051.

[69] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

[70] Goodrich, M. A., Schultz, A. C., et al. (2008). Human–robot interaction: a survey. *Foundations and Trends® in Human–Computer Interaction*, 1(3):203–275.

[71] Grabner, A., Roth, P. M., and Lepetit, V. (2018). 3d pose estimation and 3d model retrieval for objects in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3022–3031.

[72] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 369–376.

[73] Grigorescu, S., Trasnea, B., Cocias, T., and Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386.

[74] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. (2018). A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 216–224.

[75] Guimard, Q., Sassatelli, L., Marchetti, F., Becattini, F., Seidenari, L., and Bimbo, A. D. (2022). Deep variational learning for multiple trajectory prediction of 360° head movements. In *Proceedings of the ACM International Conference on Multimedia*, pages 12–26.

[76] Guo, H., Wang, G., Chen, X., and Zhang, C. (2017). Towards good practices for deep 3d hand pose estimation. *arXiv preprint arXiv:1707.07248*.

[77] Gwak, J., Choy, C. B., Chandraker, M., Garg, A., and Savarese, S. (2017). Weakly supervised 3d reconstruction with adversarial constraint. In *Proceedings of the International Conference on 3D Vision*. IEEE.

[78] Haque, A., Peng, B., Luo, Z., Alahi, A., Yeung, S., and Fei-Fei, L. (2016). Towards viewpoint invariant 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 160–177. Springer.

[79] Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. (2011). Semantic contours from inverse detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 991–998.

[80] Hasegawa, H., Mizoguchi, Y., Tadakuma, K., Ming, A., Ishikawa, M., and Shimojo, M. (2010). Development of intelligent robot hand using proximity, contact and slip sensing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 777–784.

[81] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2961–2969.

[82] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[83] Heller, J., Havlena, M., Sugimoto, A., and Pajdla, T. (2011). Structure-from-motion based hand-eye calibration using l∞ minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3497–3503. IEEE.

[84] Henderson, P. and Ferrari, V. (2018). Learning to generate and reconstruct 3d meshes with only 2d supervision. In *Proceedings of the British Machine Vision Conference*.

[85] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637.

[86] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[87] Höfer, S., Bekris, K., Handa, A., Gamboa, J. C., Mozifian, M., Golemo, F., Atkeson, C., Fox, D., Goldberg, K., Leonard, J., et al. (2021). Sim2real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2):398–400.

[88] Hong, J., Sapp, B., and Philbin, J. (2019). Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8454–8462.

[89] Horaud, R. and Dornaika, F. (1995). Hand-eye calibration. *The International Journal of Robotics Research*, 14(3):195–210.

[90] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[91] Hu, Y., Wong, Y., Wei, W., Du, Y., Kankanhalli, M., and Geng, W. (2018). A novel attention-based hybrid cnn-rnn architecture for semg-based gesture recognition. *PloS one*, 13(10):e0206049.

[92] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4700–4708.

[93] Huang, X., Cheng, X., Geng, Q., Cao, B., Zhou, D., Wang, P., Lin, Y., and Yang, R. (2018). The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 954–960.

[94] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

[95] Ilonen, J. and Kyrki, V. (2011). Robust robot-camera calibration. In *Proceedings of the International Conference on Advanced Robotics*, pages 67–74. IEEE.

[96] Insafutdinov, E. and Dosovitskiy, A. (2018). Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems*, pages 2807–2817.

[97] Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2013). Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339.

[98] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1125–1134.

[99] Ivanovic, B. and Pavone, M. (2019). The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2375–2384.

[100] Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025.

[101] Ji, X., Fang, Q., Dong, J., Shuai, Q., Jiang, W., and Zhou, X. (2020). A survey on monocular 3d human pose estimation. *Virtual Reality & Intelligent Hardware*, 2(6):471–500.

[102] Joo, H., Simon, T., Li, X., Liu, H., Tan, L., Gui, L., Banerjee, S., Godisart, T. S., Nabbe, B., Matthews, I., Kanade, T., Nobuhara, S., and Sheikh, Y. (2017). Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[103] Kahol, K., Tripathi, P., and Panchanathan, S. (2004). Automated gesture segmentation from dance sequences. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 883–888. IEEE.

[104] Kalaitzakis, M., Cain, B., Carroll, S., Ambrosi, A., Whitehead, C., and Vitzilaios, N. (2021). Fiducial markers for pose estimation. *Journal of Intelligent & Robotic Systems*, 101(4):1–26.

[105] Kanazawa, A., Tulsiani, S., Efros, A. A., and Malik, J. (2018). Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision*, pages 371–386.

[106] Kang, H., Lee, C. W., and Jung, K. (2004). Recognition-based gesture spotting in video games. *Pattern Recognition Letters*, 25(15):1701–1714.

[107] Kar, A., Tulsiani, S., Carreira, J., and Malik, J. (2015). Category-specific object reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1966–1974.

[108] Kato, H. and Harada, T. (2019a). Learning view priors for single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[109] Kato, H. and Harada, T. (2019b). Self-supervised learning of 3d objects from natural images. *arXiv preprint arXiv:1911.08850*.

[110] Kato, H., Ushiku, Y., and Harada, T. (2018). Neural 3d mesh renderer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3907–3916.

[111] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions of Graphics*, 42(4).

[112] Ki, J. and Kwon, Y.-M. (2008). 3d gaze estimation and interaction. In *Proceedings of the 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video*, pages 373–376. IEEE.

[113] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[114] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[115] Kirillov, A., Wu, Y., He, K., and Girshick, R. (2020). Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9799–9808.

[116] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154. IEEE.

[117] Kolbeinsson, A., Lagerstedt, E., and Lindblom, J. (2019). Foundation for a classification of collaboration levels for human-robot cooperation in manufacturing. *Production & Manufacturing Research*, 7(1):448–471.

[118] Kortylewski, A., He, J., Liu, Q., and Yuille, A. L. (2020). Compositional convolutional neural networks: A deep architecture with innate robustness to partial occlusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8940–8949.

[119] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

[120] Kulkarni, N., Gupta, A., Fouhey, D. F., and Tulsiani, S. (2020). Articulation-aware canonical surface mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 452–461.

[121] Kulkarni, N., Gupta, A., and Tulsiani, S. (2019). Canonical surface mapping via geometric cycle consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2202–2211.

[122] Kwon, Y.-H. and Park, M.-G. (2019). Predicting future frames using retrospective cycle gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1811–1820.

[123] Labbé, Y., Carpentier, J., Aubry, M., and Sivic, J. (2021). Single-view robot pose and joint angle estimation via render & compare. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1654–1663.

[124] Lambrecht, J. (2019). Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection. In *Proceedings of the International Conference on Robot Intelligence Technology and Applications*, pages 136–141. IEEE.

[125] Lambrecht, J. and Kästner, L. (2019). Towards the usage of synthetic data for marker-less pose estimation of articulated robots in rgb images. In *Proceedings of the IEEE International Conference on Advanced Robotics*, pages 240–247.

[126] Lander, C., Gehring, S., Krüger, A., Boring, S., and Bulling, A. (2015). Gazeprojector: Accurate gaze estimation and seamless gaze interaction across multiple displays. In *Proceedings of the Annual ACM Symposium on User Interface Software & Technology*, pages 395–404.

[127] Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242.

[128] Latombe, J.-C. (2012). *Robot motion planning*, volume 124. Springer Science & Business Media.

[129] Lee, H.-K. and Kim, J.-H. (1999). An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973.

[130] Lee, N., Choi, W., Vernaza, P., Choy, C. B., Torr, P. H., and Chandraker, M. (2017). Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 336–345.

[131] Lee, T. E., Tremblay, J., To, T., Cheng, J., Mosier, T., Kroemer, O., Fox, D., and Birchfield, S. (2020). Camera-to-robot pose estimation from a single image. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 9426–9432.

[132] Lefebvre, G., Berlemont, S., Mamalet, F., and Garcia, C. (2013). Blstm-rnn based 3d gesture classification. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 381–388. Springer.

[133] Lei, J., Sridhar, S., Guerrero, P., Sung, M., Mitra, N., and Guibas, L. J. (2020). Pix2surf: Learning parametric 3d surface models of objects from images. In *Proceedings of the European Conference on Computer Vision*.

[134] Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: Efficient perspective-n-point camera pose estimation. *International Journal of Computer Vision*, 81(2):155–166.

[135] Li, H., Ye, W., Zhang, G., Zhang, S., and Bao, H. (2020a). Saliency guided subdivision for single-view mesh reconstruction. In *Proceedings of the International Conference on 3D Vision*, pages 1098–1107. IEEE.

[136] Li, K., Wang, Y., Peng, G., Song, G., Liu, Y., Li, H., and Qiao, Y. (2022a). Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *Proceedings of the International Conference on Learning Representations*.

[137] Li, R., Liu, Z., and Tan, J. (2019). A survey on 3d hand pose estimation: Cameras, methods, and datasets. *Pattern Recognition*, 93:251–272.

[138] Li, S., Xu, C., and Xie, M. (2012). A robust o (n) solution to the perspective-n-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1444–1450.

[139] Li, W., Liu, H., Tang, H., Wang, P., and Van Gool, L. (2022b). Mhformer: Multi-hypothesis transformer for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13147–13156.

[140] Li, X. and Li, D. (2021). Gpfs: a graph-based human pose forecasting system for smart home with online learning. *ACM Transactions on Sensor Networks*, 17(3):1–19.

[141] Li, X., Liu, S., De Mello, S., Kim, K., Wang, X., Yang, M.-H., and Kautz, J. (2020b). Online adaptation for consistent mesh reconstruction in the wild. In *Advances in Neural Information Processing Systems*.

[142] Li, X., Liu, S., Kim, K., De Mello, S., Jampani, V., Yang, M.-H., and Kautz, J. (2020c). Self-supervised single-view 3d reconstruction via semantic consistency. In *Proceedings of the European Conference on Computer Vision*.

[143] Lin, C.-H., Wang, O., Russell, B. C., Shechtman, E., Kim, V. G., Fisher, M., and Lucey, S. (2019). Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 969–978.

[144] Liu, J., Liu, Y., Wang, Y., Prinet, V., Xiang, S., and Pan, C. (2020). Decoupled representation learning for skeleton-based gesture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5751–5760.

[145] Liu, S., Li, T., Chen, W., and Li, H. (2019). Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717.

[146] Liu, W. (2010). Natural user interface-next mainstream product user interface. In *Proceedings of the IEEE International Conference on Computer-Aided Industrial Design & Conceptual Design*, volume 1, pages 203–205. IEEE.

[147] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37. Springer.

[148] Long, J. L., Zhang, N., and Darrell, T. (2014). Do convnets learn correspondence? In *Advances in Neural Information Processing Systems*, pages 1601–1609.

[149] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics*, 34(6).

[150] Loper, M. M. and Black, M. J. (2014). Opendr: An approximate differentiable renderer. In *Proceedings of the European Conference on Computer Vision*, pages 154–169. Springer.

[151] Lotter, W., Kreiman, G., and Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*.

[152] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

[153] Luc, P., Couprie, C., Lecun, Y., and Verbeek, J. (2018). Predicting future instance segmentation by forecasting convolutional features. In *Proceedings of the European Conference on Computer Vision*, pages 584–599.

[154] Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems*, pages 698–707.

[155] Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., and Van Gool, L. (2017). Pose guided person image generation. In *Advances in Neural Information Processing Systems*, pages 406–416.

[156] Mandikal, P., KL, N., and Venkatesh Babu, R. (2018). 3d-psrnet: Part segmented 3d point cloud reconstruction from a single image. In *Proceedings of the European Conference on Computer Vision Workshops*.

[157] Mangalam, K., Adeli, E., Lee, K.-H., Gaidon, A., and Niebles, J. C. (2020). Disentangling human dynamics for pedestrian locomotion forecasting with noisy supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2784–2793.

[158] Manganaro, F., Pini, S., Borghi, G., Vezzani, R., and Cucchiara, R. (2019). Hand gestures for the human-car interaction: the briareo dataset. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 560–571. Springer.

[159] Marchetti, F., Becattini, F., Seidenari, L., and Del Bimbo, A. (2020). Multiple trajectory prediction of moving agents with memory augmented networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[160] Martinez, J., Hossain, R., Romero, J., and Little, J. J. (2017). A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[161] Martínez-González, A., Villamizar, M., Canévet, O., and Odobez, J.-M. (2018). Real-time convolutional networks for depth-based human pose estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 41–47.

[162] Martínez-González, A., Villamizar, M., Canévet, O., and Odobez, J.-M. (2019). Efficient convolutional neural networks for depth-based multi-person pose estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(11):4207–4221.

[163] Mathieu, M. F., Zhao, J. J., Zhao, J., Ramesh, A., Sprechmann, P., and LeCun, Y. (2016). Disentangling factors of variation in deep representation

using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048.

[164] Mehta, D., Rhodin, H., Casas, D., Fua, P., Sotnychenko, O., Xu, W., and Theobalt, C. (2017). Monocular 3d human pose estimation in the wild using improved cnn supervision. In *Proceedings of the International Conference on 3D Vision*, pages 506–516. IEEE.

[165] Mehta, D., Sotnychenko, O., Mueller, F., Xu, W., Sridhar, S., Pons-Moll, G., and Theobalt, C. (2018). Single-shot multi-person 3d pose estimation from monocular rgb. In *Proceedings of the International Conference on 3D Vision*, pages 120–130. IEEE.

[166] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106.

[167] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[168] Mitsunaga, N., Smith, C., Kanda, T., Ishiguro, H., and Hagita, N. (2008). Adapting robot behavior for human–robot interaction. *IEEE Transactions on Robotics*, 24(4):911–916.

[169] Moon, G., Chang, J. Y., and Lee, K. M. (2018). V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5079–5088.

[170] Morrical, N., Tremblay, J., Lin, Y., Tyree, S., Birchfield, S., Pascucci, V., and Wald, I. (2021). Nvisii: A scriptable tool for photorealistic image generation. *arXiv preprint arXiv:2105.13962*.

[171] Mottaghi, R., Xiang, Y., and Savarese, S. (2015). A coarse-to-fine model for 3d pose estimation and sub-category recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 418–426.

[172] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions of Graphics*, 41(4).

[173] Munea, T. L., Jembre, Y. Z., Weldegebriel, H. T., Chen, L., Huang, C., and Yang, C. (2020). The progress of human pose estimation: a survey and taxonomy of models applied in 2d human pose estimation. *IEEE Access*, 8:133330–133348.

[174] Nazeri, K., Ng, E., Joseph, T., Qureshi, F. Z., and Ebrahimi, M. (2019). Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*.

[175] Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 483–499. Springer.

[176] Noguchi, A., Iqbal, U., Tremblay, J., Harada, T., and Gallo, O. (2022). Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3677–3687.

[177] Ofli, F., Chaudhry, R., Kurillo, G., Vidal, R., and Bajcsy, R. (2013). Berkeley mhad: A comprehensive multimodal human action database. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 53–60. IEEE.

[178] Oka, R. (1998). Spotting method for classification of real world data. *The Computer Journal*, 41(8):559–565.

[179] Olson, E. (2011). Apriltag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE.

[180] Palazzi, A., Bergamini, L., Calderara, S., and Cucchiara, R. (2018). End-to-end 6-dof object pose estimation through differentiable rasterization. In *Proceedings of the European Conference on Computer Vision*.

[181] Palazzi, A., Bergamini, L., Calderara, S., and Cucchiara, R. (2020). Warp and learn: Novel views generation for vehicles and other objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):2216–2227.

[182] Palazzi, A., Borghi, G., Abati, D., Calderara, S., and Cucchiara, R. (2017). Learning to map vehicles into bird's eye view. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 233–243. Springer.

[183] Palazzi, A., Calderara, S., Bicocchi, N., Vezzali, L., Di Bernardo, G. A., Zambonelli, F., and Cucchiara, R. (2016). Spotting prejudice with nonverbal behaviours. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 853–862.

[184] Park, E., Yang, J., Yumer, E., Ceylan, D., and Berg, A. C. (2017). Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3500–3509.

[185] Park, F. C. and Martin, B. J. (1994). Robot sensor calibration: solving ax= xb on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721.

[186] Park, G., Kim, T.-K., and Woo, W. (2020). 3d hand pose estimation with a single infrared camera via domain transfer learning. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, pages 588–599. IEEE.

[187] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019a). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174.

[188] Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. (2019b). Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346.

[189] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS Autodiff Workshop*.

[190] Patel, P., Huang, C.-H. P., Tesch, J., Hoffmann, D. T., Tripathi, S., and Black, M. J. (2021). Agora: Avatars in geography optimized for regression analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13468–13478.

[191] Pauliková, A., Gyurák Babeľová, Z., and Ubárová, M. (2021). Analysis of the impact of human–cobot collaborative manufacturing implementation on the occupational health and safety and the quality requirements. *International Journal of Environmental Research and Public Health*, 18(4):1927.

[192] Pauwels, K. and Kragic, D. (2016). Integrated on-line robot-camera calibration and object pose estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2332–2339. IEEE.

[193] Pavlakos, G., Zhou, X., Chan, A., Derpanis, K. G., and Daniilidis, K. (2017a). 6-dof object pose from semantic keypoints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2011–2018. IEEE.

[194] Pavlakos, G., Zhou, X., Derpanis, K. G., and Daniilidis, K. (2017b). Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7025–7034.

[195] Pavllo, D., Feichtenhofer, C., Grangier, D., and Auli, M. (2019). 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7753–7762.

[196] Pavllo, D., Grangier, D., and Auli, M. (2018). Quaternet: A quaternion-based recurrent model for human motion. In *Proceedings of the British Machine Vision Conference*.

[197] Pavlovic, V. I., Sharma, R., and Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695.

[198] Peshkin, M. A., Colgate, J. E., Wannasuphoprasit, W., Moore, C. A., Gillespie, R. B., and Akella, P. (2001). Cobot architecture. *IEEE Transactions on Robotics*, 17(4):377–390.

[199] Pini, S., Borghi, G., Vezzani, R., Maltoni, D., and Cucchiara, R. (2021). A systematic comparison of depth map representations for face recognition. *Sensors*, 21(3):944.

[200] Plizzari, C., Cannici, M., and Matteucci, M. (2021). Spatial temporal transformer network for skeleton-based action recognition. In *Pattern Recognition*, pages 694–701.

[201] Qi, X., Liu, Z., Chen, Q., and Jia, J. (2019). 3d motion decomposition for rgbd future dynamic scene synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7673–7682.

[202] Qian, R., Lai, X., and Li, X. (2022). 3d object detection for autonomous driving: A survey. *Pattern Recognition*, 130:108796.

[203] Quattoni, A., Wang, S., Morency, L.-P., Collins, M., and Darrell, T. (2007). Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1848–1852.

[204] Ren, Y. and Gu, C. (2011). Hand gesture recognition based on hog characters and svm. *Bulletin of Science and Technology*, 2:011.

[205] Rhodin, H., Salzmann, M., and Fua, P. (2018). Unsupervised geometry-aware representation for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 750–767.

[206] Richter, S. R. and Roth, S. (2018). Matryoshka networks: Predicting 3d geometry via nested shape layers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1936–1944.

[207] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention*, pages 234–241. Springer.

[208] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2564–2571. Ieee.

[209] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

[210] Salzmann, T., Pavone, M., and Ryll, M. (2022). Motron: Multimodal probabilistic human motion forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6457–6466.

[211] Sampieri, A., di Melendugno, G. M. D., Avogaro, A., Cunico, F., Setti, F., Skenderi, G., Cristani, M., and Galasso, F. (2022). Pose forecasting in industrial human-robot collaboration. In *Proceedings of the European Conference on Computer Vision*, pages 51–69. Springer.

[212] Sárándi, I., Hermans, A., and Leibe, B. (2023). Learning 3d human pose estimation from dozens of datasets using a geometry-aware autoencoder to

bridge between skeleton formats. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 2956–2966.

[213] Sarbolandi, H., Lefloch, D., and Kolb, A. (2015). Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding*, 139:1–20.

[214] Schnürer, T., Fuchs, S., Eisenbach, M., and Groß, H.-M. (2019). Real-time 3d pose estimation from single depth images. In *Proceeding of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 5, pages 716–724.

[215] Schonberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4104–4113.

[216] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 618–626.

[217] Shan, W., Liu, Z., Zhang, X., Wang, Z., Han, K., Wang, S., Ma, S., and Gao, W. (2023). Diffusion-based 3d human pose estimation with multi-hypothesis aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14761–14771.

[218] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1297–1304.

[219] Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., et al. (2012). Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2821–2840.

[220] Simon, T., Joo, H., Matthews, I., and Sheikh, Y. (2017). Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1145–1153.

[221] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[222] Sofianos, T., Sampieri, A., Franco, L., and Galasso, F. (2021). Space-time-separable graph convolutional network for pose forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11209–11218.

[223] Song, X., Wang, P., Zhou, D., Zhu, R., Guan, C., Dai, Y., Su, H., Li, H., and Yang, R. (2019). Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5452–5462.

[224] Sorkine, O. (2006). Differential representations for mesh processing. In *Computer Graphics Forum*.

[225] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

[226] Starner, T., Weaver, J., and Pentland, A. (1998). Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375.

[227] Sun, K., Xiao, B., Liu, D., and Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703.

[228] Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., Tenenbaum, J. B., and Freeman, W. T. (2018). Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[229] Tang, Z., Naphade, M., Liu, M.-Y., Yang, X., Birchfield, S., Wang, S., Kumar, R., Anastasiu, D., and Hwang, J.-N. (2019). Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8797–8806.

[230] Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2016). Multi-view 3d models from single images with a convolutional network. In *Proceedings of the European Conference on Computer Vision*, pages 322–337. Springer.

[231] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30.

[232] Toyer, S., Cherian, A., Han, T., and Gould, S. (2017). Human pose forecasting via deep markov models. In *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications*, pages 1–8. IEEE.

[233] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977.

[234] Tremblay, J., Tyree, S., Mosier, T., and Birchfield, S. (2020). Indirect object-to-robot pose estimation from an external monocular rgb camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4227–4234.

[235] Trivedi, N., Thatipelli, A., and Sarvadevabhatla, R. K. (2021). Ntu-x: an enhanced large-scale dataset for improving pose-based recognition of subtle human actions. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 1–9.

[236] Trumble, M., Gilbert, A., Malleson, C., Hilton, A., and Collomosse, J. (2017). Total capture: 3d human pose estimation fusing video and inertial sensors. In *Proceedings of the British Machine Vision Conference*, pages 1–13.

[237] Tulsiani, S., Efros, A. A., and Malik, J. (2018). Multi-view consistency as supervisory signal for learning shape and pose prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2897–2905.

[238] Tulsiani, S., Kulkarni, N., and Gupta, A. (2020). Implicit mesh reconstruction from unannotated image collections. *preprint arXiv:2007.08504*.

[239] Tulsiani, S. and Malik, J. (2015). Viewpoints and keypoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1510–1519.

[240] Tulsiani, S., Zhou, T., Efros, A. A., and Malik, J. (2017). Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2626–2634.

[241] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 109–117.

[242] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[243] Villani, V., Pini, F., Leali, F., and Secchi, C. (2018). Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266.

[244] Villegas, R., Pathak, A., Kannan, H., Erhan, D., Le, Q. V., and Lee, H. (2019). High fidelity video prediction with large stochastic recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 81–91.

[245] Vondrick, C., Pirsiavash, H., and Torralba, A. (2016). Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 98–106.

[246] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

[247] Wang, G., Wang, Y., Zhang, H., Gu, R., and Hwang, J.-N. (2019). Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the ACM International Conference on Multimedia*, pages 482–490.

[248] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al. (2020). Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[249] Wang, K., Lin, L., Ren, C., Zhang, W., and Sun, W. (2018a). Convolutional memory blocks for depth data representation learning. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 2790–2797.

[250] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G. (2018b). Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision*, pages 52–67.

[251] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 34:27171–27183.

[252] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. (2018c). High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807.

[253] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

[254] Weiss, A., Buchner, R., Tscheligi, M., and Fischer, H. (2011). Exploring human-robot cooperation possibilities for semiconductor manufacturing. In *Proceedings of the International Conference on Collaboration Technologies and Systems*, pages 173–177. IEEE.

[255] Weiss, A., Wortmeier, A.-K., and Kubicek, B. (2021). Cobots in industry 4.0: A roadmap for future practice studies on human–robot collaboration. *IEEE Transactions on Human-Machine Systems*, 51(4):335–345.

[256] Widmaier, F., Kappler, D., Schaal, S., and Bohg, J. (2016). Robot arm pose estimation by pixel-wise regression of joint angles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 616–623.

[257] Wiles, O. and Zisserman, A. (2017). Silnet : Single- and multi-view reconstruction by learning from silhouettes. In *Proceedings of the British Machine Vision Conference*.

[258] Wu, S., Rupprecht, C., and Vedaldi, A. (2020). Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[259] Xiang, Y., Kim, W., Chen, W., Ji, J., Choy, C., Su, H., Mottaghi, R., Guibas, L., and Savarese, S. (2016). Objectnet3d: A large scale database for 3d object recognition. In *Proceedings of the European Conference on Computer Vision*, pages 160–176. Springer.

[260] Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 75–82. IEEE.

[261] Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems*.

[262] Xiao, M., Kortylewski, A., Wu, R., Qiao, S., Shen, W., and Yuille, A. (2019). Tdapnet: Prototype network with recurrent top-down attention for robust object classification under partial occlusion. In *Proceedings of the European Conference on Computer Vision Workshops*.

[263] Xie, H., Yao, H., Sun, X., Zhou, S., and Zhang, S. (2019). Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[264] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1492–1500.

[265] Xiong, F., Zhang, B., Xiao, Y., Cao, Z., Yu, T., Zhou, J. T., and Yuan, J. (2019). A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 793–802.

[266] Yan, X., Yang, J., Yumer, E., Guo, Y., and Lee, H. (2016). Perspective transformer nets: learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*.

[267] Yang, D. and Illingworth, J. (1994). Calibrating a robot camera. In *Proceedings of the British Machine Vision Conference*, pages 1–10.

[268] Yang, H.-D., Park, A.-Y., and Lee, S.-W. (2006). Robust spotting of key gestures from whole body motion sequence. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 231–236. IEEE.

[269] Yang, J., Reed, S. E., Yang, M.-H., and Lee, H. (2015). Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Advances in Neural Information Processing Systems*, pages 1099–1107.

[270] Yang, S., Quan, Z., Nie, M., and Yang, W. (2021). Transpose: Keypoint localization via transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11802–11812.

[271] Ye, M., Zhang, Q., Wang, L., Zhu, J., Yang, R., and Gall, J. (2013). A survey on human motion analysis from depth data. In *Time-of-Flight and Depth Imaging Sensors, Algorithms and Applications*, pages 149–187. Springer.

[272] Yuan, S., Garcia-Hernando, G., Stenger, B., Moon, G., Chang, J. Y., Lee, K. M., Molchanov, P., Kautz, J., Honari, S., Ge, L., et al. (2018). Depth-based 3d hand pose estimation: From current achievements to future goals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2636–2645.

[273] Yub Jung, H., Lee, S., Seok Heo, Y., and Dong Yun, I. (2015). Random tree walk toward instantaneous 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2467–2474.

[274] Zanuttigh, P., Marin, G., Dal Mutto, C., Dominio, F., Minto, L., and Cortelazzo, G. M. (2016). Time-of-flight and structured light depth cameras. *Technology and Applications*, pages 978–3.

[275] Zhang, L., Zhu, G., Mei, L., Shen, P., Shah, S. A. A., and Bennamoun, M. (2018a). Attention in convolutional lstm for gesture recognition. In *Advances in Neural Information Processing Systems*, pages 1953–1962.

[276] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018b). The unreasonable effectiveness of deep features as a perceptual metric. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[277] Zhang, W., Lin, Z., Cheng, J., Ma, C., Deng, X., and Wang, H. (2020a). Sta-gcn: two-stream graph convolutional network with spatial–temporal attention for hand gesture recognition. *The Visual Computer*, 36(10):2433–2444.

[278] Zhang, Z., Hu, L., Deng, X., and Xia, S. (2020b). Weakly supervised adversarial learning for 3d human pose estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):1851–1859.

[279] Zhang, Z., Hu, L., Deng, X., and Xia, S. (2021). Sequential 3d human pose estimation using adaptive point cloud sampling strategy. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1330–1337.

[280] Zhao, B., Wu, X., Cheng, Z.-Q., Liu, H., Jie, Z., and Feng, J. (2018). Multi-view image generation from a single-view. In *Proceedings of the ACM International Conference on Multimedia*, pages 383–391.

[281] Zheng, C., Wu, W., Chen, C., Yang, T., Zhu, S., Shen, J., Kehtarnavaz, N., and Shah, M. (2023). Deep learning-based human pose estimation: A survey. *ACM Computing Surveys*, 56(1):1–37.

[282] Zheng, C., Zhu, S., Mendieta, M., Yang, T., Chen, C., and Ding, Z. (2021). 3d human pose estimation with spatial and temporal transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11656–11665.

[283] Zhou, Q.-Y., Park, J., and Koltun, V. (2018a). Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*.

[284] Zhou, T., Tulsiani, S., Sun, W., Malik, J., and Efros, A. A. (2016). View synthesis by appearance flow. In *Proceedings of the European Conference on Computer Vision*, pages 286–301. Springer.

[285] Zhou, X., Karpur, A., Luo, L., and Huang, Q. (2018b). Starmap for category-agnostic keypoint and viewpoint estimation. In *Proceedings of the European Conference on Computer Vision*, pages 318–334.

[286] Zhu, G., Zhang, L., Shen, P., and Song, J. (2017a). Multimodal gesture recognition using 3-d convolution and convolutional lstm. *IEEE Access*, 5:4517–4524.

[287] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017b). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2223–2232.

[288] Zhu, R., Kiani Galoogahi, H., Wang, C., and Lucey, S. (2017c). Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[289] Zuo, Y., Qiu, W., Xie, L., Zhong, F., Wang, Y., and Yuille, A. L. (2019). Craves: Controlling robotic arm with a vision-based economic system. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4214–4223.

Acknowledgments

I would like to thank all the people who were capable of leaving a good memory during my Ph.D. journey.

Starting from my personal life, I can not find words for my family who has always been supportive of my choices and sometimes has to stand for me during tough times. We are used to taking those things for granted, but having such support is crucial for anybody. A special thanks goes to my parents who have struggled a lot to raise and understand me. They have always been interested in what I was doing at work even though it was tough for them to understand at first, so they asked me many times and I had to repeat. In addition, I would mention my grandmother who has been the person to whom I could have always spoken freely about everything and she has always been available and believed in me. Of course, I thank the rest of my family from my grandparents to my aunts and my cousin; everyone has positively contributed to my life. Finally, I would thank all of my friends who always tried to make me happy and celebrate all my achievements as soon as I told them.

Going into my academic life, I would like to thank my advisor, Prof. Roberto Vezzani, who allowed me to begin this Ph.D. journey and has always been available for any technical or theoretical problem that has to be solved. Moreover, I thank my colleagues Stefano, Guido, Luca, and Andrea who guided me since the beginning of my academic research adventure and helped me develop and write my research works. I would mention also my collaborators from the University of Florence with whom we recently worked hard on an interesting research activity.

Last but not least, I would like to thank all the AImageLab people at my uni-

versity. I can consider those guys as my second family. I got in the lab in 2019 during my MSc thesis and we were like 10 people; now the family is even bigger with more than 30 people. I could not connect to everybody, but all of them left something in me that I will take forever. I hope you appreciated the time spent together especially when I tried to spread positivity by going around the lab, making jokes, and sometimes bothering you at work (sorry for that!). Among all of them, I would mention some people that I do not consider only colleagues, but also great friends with whom I shared many great moments outside the academic sphere. So, thanks a lot to Samuele, Davide M., Vittorio, Silvia, Monica, Fedy, Nello, Davide D., Luca, and Roberto.