Enhancing Human-Robot Interaction with Computer Vision

Yuming Du

École des Ponts ParisTech

THÈSE DE DOCTORAT
de l'École des Ponts ParisTech

Enhancing Human-Robot Interaction with Computer Vision

École doctorale N°532, MSTIC

Signal, Image, Automatique

Thèse préparée au sein du LIGM-IMAGINE / École des Ponts ParisTech

Thèse soutenue le 19 Septembre 2023, par
Yuming DU

Composition du jury:

Marie-Odile BERGER Directrice de recherche, Inria	*Présidente*
Christian, WOLF Principal Scientist, Naver Labs	*Rapporteur*
Karteek, ALAHARI Chargé de Recherche, Inria	*Rapporteur*
Eric, MARCHAND Professeur, Université de Rennes	*Examinateur*
Federica BOGO Research Scientist, Meta	*Examinateur*
Vincent, LEPETIT Professeur, Ecole des Ponts ParisTech	*Directeur de thèse*

Abstract

In recent years, the field of robotics has undergone substantial progress, particularly in the pursuit of creating robots capable of effortlessly interacting with humans in intricate environments. Central to this aim is the need for robots to comprehend their surroundings, foresee human actions, and adjust their movements in response. In this thesis, we explore the challenge of improving a robot's abilities in scene understanding, human motion forecasting and synthesis, and the application of acquired human motion knowledge to robot movements.

In the opening part of this thesis, we explore the enhancement of a robot's ability to understand its surroundings on multiple levels. We introduce an innovative framework that enables robots to autonomously identify and segment objects in open-world environments using self-training (published at CVF/IEEE ICCV2021). By leveraging deep learning techniques, our approaches enable robots to efficiently learn from their surroundings and recognize previously unseen objects. In addition to object discovery, we introduce a method to improve the robot's monocular depth estimation capabilities (published at CVF/IEEE CVPR2020). This enhancement further refines the robot's understanding of its environment by providing a more accurate representation of depth information from a single-camera viewpoint. Together, these advancements strengthen the robot's adaptability and performance in navigating complex and dynamic situations.

In the second part, we focus on improving the robot's ability to understand and predict human motion. We present two distinct methods that investigate either historical human motions (published at CVF/IEEE WACV2023) or observed partial joint movements (published at CVF/IEEE CVPR2023) as a basis for accurately anticipating human motion. These enhancements enable robots to collaborate more effectively with humans by foreseeing their motions, thereby playing a crucial role in fostering safe and efficient human-robot interactions.

In the third and concluding part of this thesis, we tackle the challenge of converting learned human motion into robot movements. We introduce a method designed to adapt human grasp demonstrations for use with any multi-fingered grippers, allowing robots to intuitively and effectively manipulate objects (published at IEEE IROS 2022). By integrating kinematic mapping and optimization techniques, our approach guarantees that the adapted grasps are both physically viable and resilient, empowering robots to carry out intricate manipulation tasks in environments centered around human interaction.

By combining and integrating the three components proposed in this thesis, we seek to substantially enhance a robot's capacity to interact with humans in complex environments. Our proposed

enhancements, including improved scene understanding, accurate human motion prediction, and effective grasp adaptation capabilities, are aimed at empowering robots to engage in more seamless, safe, and efficient human-robot collaborations across various domains and applications. Collectively, these advancements lay the foundation for developing more sophisticated and intuitive robotic systems that can adapt to the dynamic and evolving nature of human environments.

Résumé

Le domaine de la robotique a connu récemment des progrès considérables, notamment dans la recherche visant à créer des robots capables d'interagir sans effort avec les humains dans des environnements complexes. Au cœur de cet objectif se trouve la nécessité pour les robots de comprendre leur environnement, d'anticiper les actions humaines et d'ajuster leurs mouvements en conséquence. Dans cette thèse, nous explorons le défi d'améliorer les capacités d'un robot en matière de compréhension de scène, de prévision et de synthèse du mouvement humain, et de l'application des connaissances acquises sur le mouvement humain aux mouvements des robots.

Dans la première partie de cette thèse, nous explorons l'amélioration de la capacité d'un robot à comprendre son environnement à plusieurs niveaux. Nous présentons un cadre novateur qui permet aux robots d'identifier et de segmenter les objets de manière autonome dans des environnements ouverts en utilisant l'auto-apprentissage (publié à CVF/IEEE ICCV2021). En tirant parti des techniques d'apprentissage profond, nos approches permettent aux robots d'apprendre efficacement de leur environnement et de reconnaître des objets jamais vus auparavant. En plus de la découverte d'objets, nous introduisons une méthode pour améliorer les capacités d'estimation de profondeur monoculaire du robot (publiée à CVF/IEEE CVPR2020). Cette amélioration affine davantage la compréhension de l'environnement par le robot en fournissant une représentation plus précise des informations de profondeur à partir d'un point de vue à une seule caméra. Ensemble, ces avancées renforcent l'adaptabilité et les performances du robot dans la navigation de situations complexes et dynamiques.

Dans la deuxième partie, nous nous concentrons sur l'amélioration de la capacité du robot à comprendre et à prédire le mouvement humain. Nous présentons deux méthodes distinctes qui étudient soit les mouvements humains historiques (publiée à CVF/IEEE WACV2023), soit les mouvements articulaires partiels observés (publiée à CVF/IEEE CVPR2023) comme base pour anticiper avec précision le mouvement humain. Ces améliorations permettent aux robots de collaborer plus efficacement avec les humains en prévoyant leurs mouvements, jouant ainsi un rôle crucial dans la promotion d'interactions sûres et efficaces entre humains et robots.

Dans la troisième et dernière partie de cette thèse, nous abordons le défi de convertir les mouvements humains appris en mouvements de robots. Nous introduisons une méthode conçue pour adapter les démonstrations de préhension humaine à l'utilisation avec n'importe quel préhenseur à plusieurs doigts, permettant aux robots de manipuler intuitivement et efficacement les objets (publié à IEEE IROS 2022). En intégrant les techniques de cartographie cinématique et d'optimisation, notre approche garantit que les préhensions adaptées sont à la fois physiquement

viables et résilientes, permettant aux robots d'effectuer des tâches de manipulation complexes dans des environnements axés sur l'interaction humaine.

En combinant et en intégrant les trois composants proposés dans cette thèse, nous cherchons à améliorer considérablement la capacité d'un robot à interagir avec les humains dans des environnements complexes. Nos améliorations proposées, notamment une meilleure compréhension de la scène, une prédiction précise du mouvement humain et des capacités d'adaptation de préhension efficaces, visent à donner aux robots la possibilité de s'engager dans des collaborations homme-robot plus fluides, sûres et efficaces dans divers domaines et applications. Collectivement, ces avancées posent les bases pour le développement de systèmes robotiques plus sophistiqués et intuitifs capables de s'adapter à la nature dynamique et en constante évolution des environnements humains.

ACKNOWLEDGMENT

I would like to express my heartfelt gratitude to all those who have contributed to the completion of this doctoral thesis. This journey has been both challenging and rewarding, and I am immensely grateful for the support and encouragement I have received.

First and foremost, I extend my deepest appreciation to my advisor, Dr. Vincent Lepetit, whose guidance, expertise, and unwavering support have been instrumental throughout this research endeavor. Your mentorship has not only shaped the trajectory of this thesis but has also been a source of inspiration for my academic and personal growth.

I am also indebted to the members of my doctoral committee, Dr. Marie-Odile Berger, Dr. Christian Wolf, Dr. Karteek Alahari, Dr. Eric Marchand, Dr. Federica Bogo, for their valuable insights, constructive feedback, and scholarly input. Your collective expertise has enriched the quality of this work and broadened my understanding of the subject matter.

Besides, I would like to thank the help and guidance of other professors and researchers I worked with: Dr. Renaud Marlet, Dr. Xavier Alameda-Pineda, Dr. Francesc Moreno-Noguer, Dr. Philippe Weinzaepfel, Dr. Romain Brégier, Dr. Artsiom Sanakoyeu, Dr. Albert Pumarola, Dr. Ali Thabet, Dr. Sebastian Starke, Dr. Robin Kips. And I would also like to thank my other collaborators: Dr. Wen Guo, Dr. Yang Xiao, Dr. Xi Shen, Dr. Xu Hu, Dr. Xiaoyu Bie, Yangtao Wang, Van Nguyen Nguyen.

I am grateful to my parents for their unwavering support throughout this journey. A heartfelt thanks to all my friends who stood by me during these challenging pandemic years. I want thank Yang, who shared the experience of exploring the magnificent Alps

with me during one of the most difficult time of my Ph.D career. A special acknowledgment goes to my wife, who constantly supports and encourages me, and most importantly, gives me the opportunity to become her husband.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

In recent times, the increasing integration of robots into various aspects of modern life has made it necessary for them to be able to interact with humans in complex environments. Achieving this goal requires that robots possess the ability to comprehend their surroundings, anticipate human actions and subsequently adapt their movements. Hence, there is a need for research in improving a robot's capabilities in areas such as scene understanding, human motion forecasting and synthesis, as well as the application of human motion knowledge to robot movements. When robots possess a deep understanding of their environment and humans, they can work harmoniously with humans and guarantee safety, efficiency, and effectiveness. Such research is essential for fostering effective human-robot collaboration across several industries and applications, including manufacturing, healthcare, autonomous driving, entertainment etc.

Manufacturing. The utilization of collaborative robots in modern manufacturing facilities has become more prevalent, with these machines working alongside human workers. Enhancing their scene understanding capabilities is crucial in enabling collaborative robots to accurately recognize unseen objects, predict human movements, and prevent unsafe actions. This ultimately promotes a safer work environment while simultaneously enhancing production line efficiency. Furthermore, the application of human motion knowledge enables collaborative robots to execute tasks in a more intuitive and coordinated manner, consequently fostering seamless human-robot collaboration.

Healthcare. In healthcare settings, accurate identification of medical instruments, patients, and other pertinent objects is fundamental to the success of subsequent treatments. To this end, it is imperative that robots possess strong scene understanding capabilities and the capacity to navigate complex home environments. Enhanced human motion forecasting abilities enable robots to anticipate medical professionals' actions, thereby offering timely and precise assistance. Additionally, robots can serve as essential aides in physical therapy and rehabilitation, as they can replicate human movements to guide patients

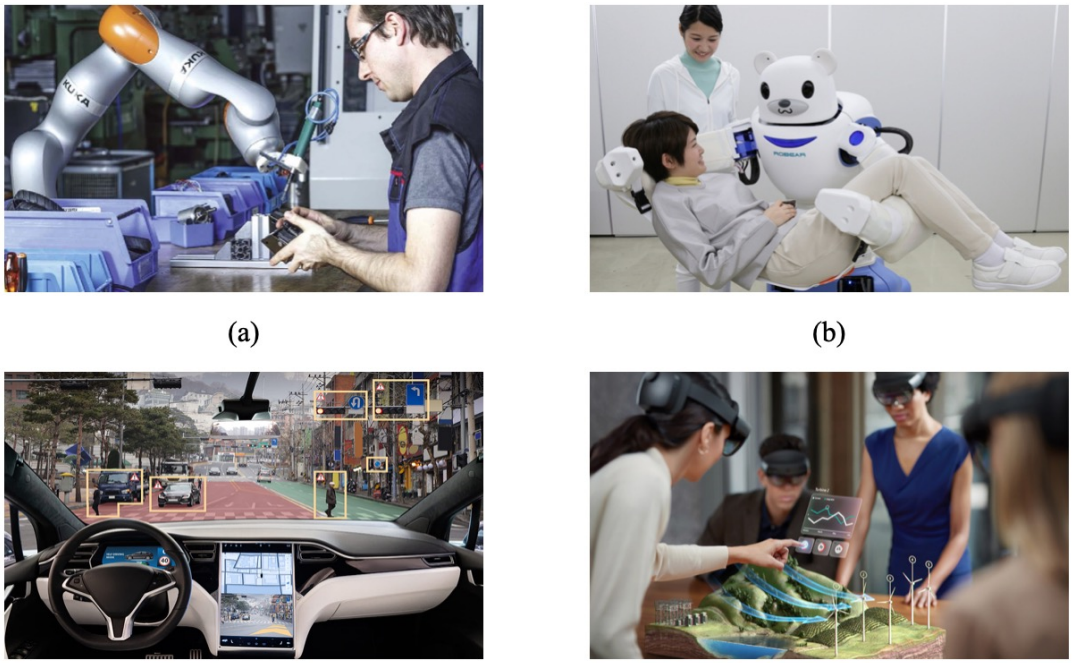


Figure 1.1: Human-robot interaction in complex environment. (a) shows a case for robot working with human in industries, image from [256]. (b) shows a case for healthcare robot helping people, image from [190]. (c) shows a case for autonomous driving where the self-driving cars need to discover and identify the emerging objects, image from [179]. (d) shows a case for augmented reality, image from [186].

through exercises and monitor their progress.

Autonomous driving. Autonomous driving technology heavily relies on the integration of scene understanding and human motion forecasting capabilities. Advanced scene understanding is a critical component that facilitates accurate environmental perception by vehicles, enabling them to detect and identify previously unseen obstacles, pedestrians, and other vehicles on the road. Human motion forecasting, on the other hand, enables vehicles to predict pedestrian behavior, adjust their speed, and take evasive measures when required. The significance of these capabilities cannot be overstated, as they are crucial for guaranteeing the safety of both autonomous vehicle passengers and other road users.

Entertainment and leisure. In recent times, the growing prevalence of entertainment applications in Augmented Reality (AR) has heightened the importance of the ability to discover and reconstruct unseen objects. Current AR devices have the capacity to capture only a small portion of human joints, the ability to synthesize full-body movements can significantly enhance the realism of the virtual environment, promoting greater immersion and more realistic interactions.

1.2 CHALLENGES AND GOALS

There are multiple challenges faced by robots when attempting to interact with humans in complex environments. These challenges must be addressed to enable seamless human-robot collaboration across various domains.

Object localization in unfamiliar environments. A primary challenge for robots interacting with humans in complex settings is the ability to locate previously unseen objects in new environments. To navigate and operate efficiently, robots must be capable of accurately localizing objects, even if they have not encountered them before. This requires the development of advanced object recognition algorithms and machine learning techniques that can generalize from prior experiences and adapt to novel situations.

In-depth understanding of human motions for forecasting and synthesis. To facilitate effective interaction with humans, robots must possess a profound understanding of human motions. This knowledge is crucial for accurately forecasting human actions and synthesizing human motion during the interaction. Developing algorithms and models that can capture the intricacies of human movement presents a significant challenge. Furthermore, robots must be able to adapt their predictions and responses in real-time as human behaviors evolve during the interaction.

Transferability of human demonstration to diverse robot types and sizes. Robots come in various shapes, sizes, and configurations, presenting another challenge for human-

robot interaction. Current methods predominantly focus on transferring human demonstrations to a specific type of robot, limiting their applicability across diverse robotic platforms. It is essential to develop methods and techniques that can transfer human demonstrations to any robots, irrespective of its type or size. This requires the creation of robust and adaptable frameworks that can generalize across different robot architectures, taking into account factors such as kinematic constraints, control methods, and sensor modalities.

Therefore, in this thesis, our objective is to develop sophisticated machine learning techniques capable of addressing the aforementioned challenges, thereby facilitating seamless human-robot collaboration. By focusing on the development of these advanced methods, we aspire to enhance the efficacy, intuitiveness, and adaptability of robots, enabling them to effectively interact with humans in a multitude of complex environments.

1.3 CONTRIBUTIONS

Considering the aforementioned challenges, we begin with tackling the problem of monocular depth estimation, as depth information is an essential component of perception and scene understanding. We propose a novel method that enhances the reconstruction accuracy and occlusion boundary localization of existing monocular depth estimation techniques while maintaining low computational costs, thus providing a generalizable solution. Due to the absence of a suitable dataset for evaluating occlusion boundary reconstruction, we have undertaken the task of manually annotating a dataset to facilitate the evaluation of our proposed method. Additionally, we have investigated the problem of detecting and segmenting previously unseen objects in an open-world setting without the need for supplementary labeled data. In such scenarios, new object classes may emerge, and traditional detection or instance segmentation approaches necessitate the use of additional labeled data for training. To overcome this challenge, we have introduced a framework that employs a pre-trained instance segmentation model to automatically generate high-quality pseudo-labels, which can be utilized to enhance the model's perfor-

mance on unseen object classes. Through the meticulous design of benchmarks, we have demonstrated the effectiveness of our proposed method in addressing the central issues of monocular depth estimation and object detection in complex environments.

Subsequently, our research concentrates on enhancing the understanding of human motion. Specifically, we have devised a novel approach to address the challenge of human motion prediction. Our proposed method, consisting of merely fully connected layers and layer normalization, outperforms all previous state-of-the-art approaches and is characterized by its lightweight design and ease of integration into mobile devices for rapid inference. Expanding on this work, we have further confronted the issue of motion synthesis based on sparse input signals. In particular, our approach seeks to synthesize full-body movements, conditioned on a subset of available joint positions and rotations. To accomplish this, we have devised a novel framework inspired by the aforementioned network, capable of synthesizing plausible human motions with high speed and minimal memory consumption. By augmenting our comprehension of human motion prediction and synthesis, we can bolster the abilities of robots to collaborate more effectively with humans across various industries and applications. These advancements pave the way for innovative methods of streamlining operations, minimizing errors, and boosting productivity.

Finally, we have directed our research efforts towards addressing the challenge of transferring human demonstrations to robot actions. In particular, we have proposed a multi-step, optimization-based approach for transferring grasping techniques from human demonstrations to arbitrary multifingered robotic grippers. This approach enables robots to grasp objects with human-like dexterity, broadening their capabilities. Moreover, we have developed novel metrics to quantify the similarity between human and robot grasps, thereby providing a more accurate assessment of the transferability of grasping techniques. To ensure the real-world applicability of our proposed method, we have utilized an Allegro gripper mounted on a Panda arm for experimental validation.

Through these advancements, we aim to facilitate the seamless transfer of human demonstrations to diverse robot types and sizes, further enhancing human-robot collabo-

ration across a wide array of industries and applications. To summarize, the contributions of this thesis are as follows:

- We first tackle the scene understanding problem, proposing a novel method that enhances reconstruction accuracy. This includes manually annotating a dataset and developing a framework that generates high-quality pseudo-labels for object detection and segmentation in open-world settings.
- The study focuses on human motion prediction and synthesis, devising a lightweight method that outperforms previous approaches and can be integrated into mobile devices. This approach synthesizes full-body movements based on sparse input signals, facilitating better human-robot collaboration across industries and applications.
- Lastly, the research tackles the challenge of transferring human demonstrations to robot actions, proposing a multi-step, optimization-based approach for transferring grasping techniques to multifingered robotic grippers. This includes developing novel metrics to quantify similarity between human and robot grasps and testing the approach on a real-world robot setup.

1.4 Manuscript Structure

This manuscript is organized as follows. In Part I, we introduce our approaches for enhancing monocular depth estimation accuracy in Chapter 3 and generating pseudo-labels for unseen classes from unlabeled videos in Chapter 4; Part II introduces the methods we propose for human motion prediction in Chapter 5 and synthesis 6; Part III presents a multi-step optimization based method for transferring human grasp to any multifingered robotic grippers grasp in Chapter 7. Chapter 8 concludes the thesis.

1.5 Publications and Submissions

This section lists the papers I published during my Ph.D. In the first part of this thesis, we mainly discuss two papers related to scene understanding:

- [221] Michael Ramamonjisoa, Yuming Du, Vincent Lepetit. Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2020.*
- [58] Yuming Du, Yang Xiao, Vincent Lepetit. Learning to Better Segment Objects from Unseen Classes with Unlabeled Videos. In *IEEE/CVF International Conference on Computer Vision (ICCV) 2021.*

Then, in the second part, we discuss two papers related to human motion understanding:

- [85] Wen Guo, Yuming Du, Xi Shen, Vincent Lepetit, Xavier Alameda-Pineda, Francesc Moreno-Noguer. Back to MLP: A Simple Baseline for Human Motion Prediction. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2023.*
- [56] Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali Thabet, Artsiom Sanakoyeu. Avatars Grow Legs: Generating Smooth Human Motion from Sparse Tracking Inputs with Diffusion Model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2023.*

In the last part of this thesis, we discuss our paper for transferring human knowledge to robot:

- [57] Yuming Du, Philippe Weinzaepfel, Vincent Lepetit, Romain Brégier. Multi-Finger Grasping Like Humans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2022.*

Besides, we have won two championships on the Unidentified Video Objects (UVO) workshop for open-world object segmentation on image-track and video-track [1] and created a dataset for evaluating occlusion boundaries based on the NYUv2 dataset [233].

There are also some other works which are not discussed in this thesis:

- [195] Van Nguyen Nguyen, Yuming Du, Yang Xiao, Michael Ramamonjisoa, Vincent Lepetit. PIZZA: A Powerful Image-only Zero-Shot Zero-CAD Approach to 6DoF Tracking. In *International Conference on 3D Vision (3DV) 2022.*
- [215] Georgy Ponimatkin, Nermin Samet, Yang Xiao, Yuming Du, Renaud Marlet, Vincent Lepetit. A Simple and Powerful Global Optimization for Unsupervised Video Object Segmentation. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2023.*
- [275] Yang Xiao, Yuming Du, Renaud Marlet. PoseContrast: Class-Agnostic Object Viewpoint Estimation in the Wild with Pose-Aware Contrastive Learning. In *International Conference on 3D Vision (3DV) 2021.*

[1] https://sites.google.com/view/unidentified-video-object/2021-iccv-workshop-program/challenge-intro

CHAPTER 2

RELATED WORKS

In this section we review the existing works related to this thesis. Section 2.1 and Section 2.2 include the related works for the Part I. In Section 2.1 we review monocular depth methods and the methods that are used to refine the estimated depth. In Section 2.2 we review the instance segmentation methods for images and videos, which can be used to generate masks on unlabeled videos. Section 2.3 includes the related works for the Part II, where we review the methods for human motion prediction and human motion synthesis. Section 2.4 includes the related works for Part III, where we review the methods for robot grasping simulation and transferring human grasp demonstration to robot gripper grasp.

2.1 Monocular Depth Estimation

Monocular depth estimation is a specialized task in the field of computer vision that revolves around deducing the depth or distance information of various elements within a scene, based solely on one single image. This effectively involves gauging the distance of different objects in the scene from the perspective of a single camera lens. The significance of monocular depth estimation spans a wide range of applications. It plays a crucial role in 3D reconstruction, a process vital for creating detailed and realistic digital representations of objects or environments. In the realm of augmented reality, it serves to seamlessly integrate virtual elements into the real world. Moreover, it holds a key position in autonomous driving systems where understanding the depth of objects is paramount for safe navigation. Furthermore, in robotics, depth estimation assists robots in efficiently navigating their environment. Despite its broad applications, monocular depth estimation is inherently challenging. It demands the model to comprehend and decode intricate relationships between the various objects in a scene and their corresponding depth values. Additionally, the task is further complicated by various influencing factors such as the prevalent lighting conditions, occlusions where one object might partially hide another, and the texture of the objects, each of which can alter the perceived depth information.

2.1.1 Monocular Depth Estimation (MDE)

Monocular depth estimation (MDE) has been a topic of extensive research due to its practical and fundamental importance. With the emergence of deep learning methods and large-scale datasets, MDE has experienced significant progress, with both supervised and self-supervised approaches being explored. To address the issue of scale ambiguity discussed in [60], multi-scale deep learning methods have been widely employed to learn depth priors, as they are capable of extracting global context from images [59]. This approach has been continually refined and improved, with the development of more sophisticated and deeper architectures [228, 95, 137]. Ordinal regression was proposed as an alternative approach to direct depth regression in MDE [70]. This method achieved state-of-the-art performance on popular MDE benchmarks such as NYUv2-Depth [233] and KITTI [72], and has been further extended in [141].

2.1.2 Occlusion Boundaries in Depth Maps

Occlusion boundaries in an image are areas where the depth changes abruptly and markedly. They essentially delineate the regions where one object overlaps or obscures another, resulting in a significant variation in depth. An example of occlusion boundaries is shown in Figure 2.1. Occlusion boundaries present a significant and persistent challenge in dense reconstruction from images for several reasons. For instance, in stereo reconstruction, pixels within the vicinity of occluding boundaries may be concealed in one image but visible in the other, thereby complicating pixel matching. Over the years, various solutions have been proposed to address this issue, including approaches such as those described in [71, 107, 121, 73]. In this thesis, we focus on recent techniques employed in monocular depth estimation to enhance the reconstruction of occlusion boundaries. Despite the numerous advancements made in MDE, improving occlusion boundary reconstruction remains a crucial and challenging problem. Our work aims to address this challenge by introducing a novel approach that enhances the sharpness and localization accuracy of occlusion boundaries in depth maps.

(a) (b) (c)

Figure 2.1: The figure shows an example of the desired results for monocular depth estimation and occlusion boundary estimation. (a) displays the input RGB image. (b) shows the ground truth depth image. (c) shows the ground truth occlusion boundaries of the depth image. Images from [217].

2.1.3 Occlusion Boundaries Refinement Methods

A variety of techniques exist that strive to enhance the accuracy of occlusion boundaries in predicted depth images. Broadly, these can be classified into two main categories, based on their reliance on post-processing procedures. One category employs post-processing, tweaking the generated depth images after initial prediction. In contrast, the other category incorporates strategies that do not depend on post-processing, focusing instead on improving the accuracy during the training.

Many methods focus on improving the occlusion boundaries during the training, ordinal regression [70, 141] has been shown to enhance occlusion boundary sharpness, However, our experiments indicate that these boundaries are often inaccurately located in the image. Another approach to improving the sharpness of object and occlusion boundaries in MDE is through loss function design. L1-loss and its variants, such as the Huber and BerHu estimators [202, 303], have become popular alternatives to L2 since they tend to penalize discontinuities less. However, this solution is not perfect and inaccuracies or smoothness in occlusion boundaries may persist [119]. To further enhance the reconstruction quality of occlusion boundaries, previous work has explored depth gradient matching constraints [59] and depth-to-image gradient constraints [96, 79]. However, for the latter, occlusion boundaries do not always correspond to strong image gradients, especially in ar-

eas with texture gradients. Constraints explicitly based on occlusion boundaries have also been proposed [286, 285, 222], leading to performance improvements in occlusion boundary reconstruction quality. Our work complements all of the aforementioned approaches, as we demonstrate that our method can enhance the localization and reconstruction of occlusion boundaries in all state-of-the-art deep learning-based MDE methods.

For the post-processing methods, several previous works have utilized the post-processing potential of Conditional Random Fields (CRFs) to refine depth map predictions. These methods generally define pixel-wise and pair-wise loss terms between pixels and their neighbors using an intermediate predicted guidance signal, such as geometric features [262] or reliability maps [97]. An initially predicted depth map is then refined by performing CRF inference, sometimes iteratively or using cascades of CRFs [280, 281]. Although these methods help improve initial depth predictions and yield qualitatively more appealing results, they typically under-perform state-of-the-art non-CRF MDE methods and are computationally more expensive. An alternative option for depth refinement is to use image enhancement methods, which may not necessarily explicitly target occlusion boundaries, but can potentially serve as alternative solutions to our proposed approach. Bilateral filtering is a popular method for image enhancement, particularly as a denoising method that preserves image contours. Although it was historically limited to post-processing due to its computational complexity, recent work has successfully made bilateral filters reasonably efficient and fully differentiable [153, 272]. These recent methods have been successful when applied in downsampling-upsampling schemes but have not yet been used in the context of MDE. Guided filters [94] have been proposed as a simpler alternative version of the bilateral filter. Our experiments show that both guided and bilateral filters can sharpen occlusion boundaries thanks to their use of the image for guidance. However, they sometimes produce false depth gradient artifacts. The bilateral solver [13] formulates the bilateral filtering problem as a regularized least-squares optimization problem, allowing fully differentiable and much faster computation. However, we demonstrate in our experiments that our end-to-end trainable method compares favorably against this method, both in terms of speed and accuracy.

2.1.4 Datasets with Image Contours

Several datasets of image contours or occlusion boundaries already exist. Popular datasets for edge detection training and evaluation have focused on perceptual boundaries [180, 181, 7] or object instance boundaries [61]. However, these datasets often lack annotation of the occlusion relationship between two regions separated by the boundaries. Other datasets [225, 101, 102, 263] have annotated the occlusion relationship between objects, but they do not contain ground truth depth information. The NYUv2-Depth dataset [233] is a popular MDE benchmark that provides ground truth depth information. Several methods for instance boundary detection have benefited from this depth information [86, 53, 276, 87, 47] to improve their performance on object instance boundary detection. The above cited datasets all lack annotations for object self-occlusion boundaries and are sometimes inaccurately annotated. Our NYUv2-OC++ dataset provides manual annotations for occlusion boundaries on top of NYUv2-Depth for all 654 test images. As discussed in [222], even though it is a tedious task, manual annotation is much more reliable than automated annotation that could be obtained from depth maps. This dataset enables the simultaneous evaluation of depth estimation methods and occlusion boundary reconstruction, like the 100-image iBims dataset [131], but is larger and has been widely used for MDE evaluation.

2.2 Instance Segmentation for Images and Videos

Instance segmentation, a task in the realm of computer vision, revolves around the detection, distinction, and separation of individual objects. This includes the precise delineation of each object's boundaries and assigning a unique identifier or label to each distinct entity. In the case of an image, the ultimate objective of instance segmentation is to create a comprehensive pixel-level map of the image. In this detailed segmentation map, each pixel is meticulously allocated to a particular object instance, resulting in a visual representation where every object is distinctly identified and segmented. Analogously, in the case of a video, the instance segmentation aims to create a pixel-level map that is

consistent across time. In this section, we first provide an overview of recent works on instance segmentation from color images, then we survey several works on video object segmentation. An example of these two tasks are shown in Figure 2.2

2.2.1 Image-level Instance Segmentation

The use of Deep Learning has recently significantly improved the performance of instance segmentation, with top-down solutions occupying top ranks in various benchmarks [93, 33, 130]. These methods typically follow a detect-then-segment pipeline to generate instance mask based on the generated object proposals. The way to generate object proposals is critical. Some works [93, 106, 212, 35] are anchor-based, which generate object proposals with the help of pre-defined anchors of different scales and shapes. Others are anchor-free [32, 295, 278], which remove the need of anchor and generate object proposals by regressing object bounding boxes directly from the feature maps for the whole image. Alternatively, bottom-up approaches assign pixels to objects by clustering the feature embedding predicted for each pixel [10, 129, 45, 198, 142, 37, 158]. Unlike top-down approaches which need non-maximum-suppression (NMS) to remove redundant detections, bottom-up approaches eliminate the NMS as there is no overlap among the predictions of bottom-up approach. Additionally, some recent approaches [265, 266] propose to directly predict the instance masks, without dependency on proposals or pixel-embedding. However, the performances of both bottom-up and direct methods are lagging behind top-down methods on most benchmarks.

2.2.2 Video Object Segmentation

There are several settings for video object segmentation (VOS). This thesis monstly related to the One-Shot, Zero-Shot video object segmentation. Zero-Shot VOS can be also considered as saliency-based video object segmentation. One-Shot VOS methods aim to segment objects in a video when ground truth segmentation is available for a frame. These methods typically warp the provided segmentation to other frames [24, 151, 264]. However, they require manual annotations and cannot generate new predictions if new objects

Figure 2.2: The figure shows an example of the instance segmentation for images and videos. The first row displays the instance segmentation task for single RGB images, images from [93]. The second row shows the desired output for video instance segmentation task, images from [284].

appear. Some Zero-Shot methods [39, 199, 255, 257] are trained with video labels on seen classes and can generalize to unseen classes, but video labeling is labor-intensive. Some methods search for salient regions in videos [36, 42, 44, 63, 113, 132, 168, 204], as salient regions tend to correspond to objects. However, saliency prediction has two major limitations for our purpose: (a) it can be fooled by non-salient camouflaged objects, and (b) it merges adjacent objects into a single salient region, whereas we aim to identify them individually.

2.2.3 Self-Learning on Unlabeled Data for Segmentation Tasks

Recently, several methods have been proposed to explore self-supervision using unlabeled static images. These include data distillation [218], unlabeled images from the web [152], consistency across image flipping [116], or an estimate of the uncertainty of prediction [194]. While these approaches are compelling, unlabeled videos are readily available and have the potential to yield more reliable results. In our experiments, we compare our method against the most representative methods and demonstrate that our approach achieves significantly better performance. Another approach involves grouping pixels with similar colors or image features in a bottom-up manner to generate masks [21, 82, 160, 211, 270]. However, this approach can be easily influenced by local textures or

colors, and some of these methods have only been tested on synthetic images. In contrast, our approach is much more robust as it starts from a pre-trained instance segmentation model. Similar to our approach, some methods leverage unlabeled videos for urban scene segmentation [34] and face and human detection [120]. However, these works focus only on enhancing model performance on existing classes and do not consider novel classes. In contrast, our approach addresses the problem of detecting and segmenting objects from new classes without any human intervention. Another related work is [200], which uses stereo video data with depth information to reconstruct a static background, and object proposals are generated from the foreground regions by subtraction. While this is an interesting approach, it requires depth data and static backgrounds, which are not always available in real-world scenarios.

2.3 Human Motion Prediction and Synthesis

Human motion understanding involves the interpretation and recognition of human actions from observed motion data. With the advancement of computer vision and machine learning technologies, it has become a critical factor in the creation of intelligent systems. In this thesis, we mainly focus on two subdomains of human motion understanding, which is human motion prediction and human motion synthesis. An example of human prediction and human synthesis is shown in Figure 2.3.

2.3.1 Human Motion Prediction

Human motion prediction is framed as a sequence-to-sequence task, where the past observed motion serves as input to predict the future motion sequence. Traditional methods explore human motion prediction using nonlinear Markov models [143], Gaussian Process dynamical models [260], and Restricted Boltzmann Machine [242]. These approaches have demonstrated effectiveness in predicting simple motions but struggle to predict complex and long-term motions [69]. In the era of deep learning, human motion prediction has achieved tremendous success with the use of deep networks, including Re-

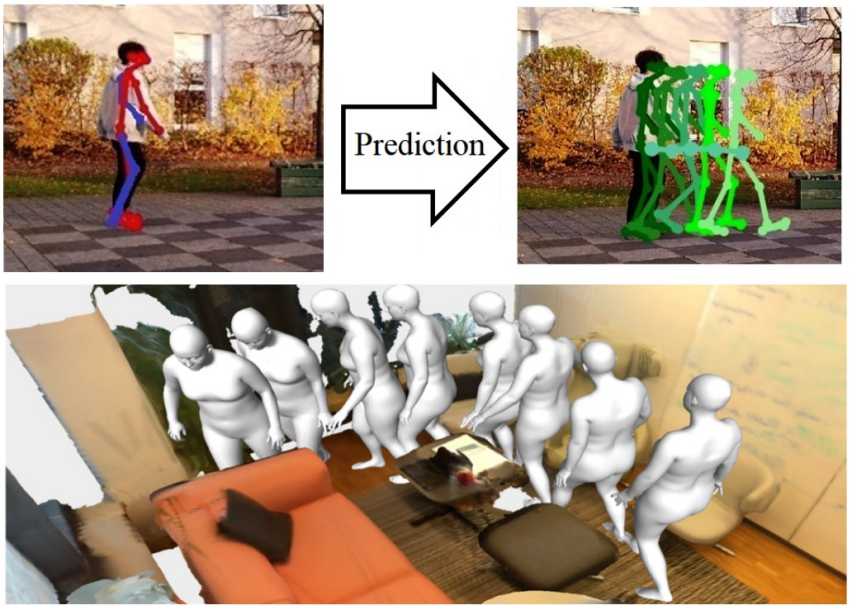


Figure 2.3: The figure shows an example of the human motion prediction and human motion synthesis. The first row displays pipeline of human motion prediction, where historical observed motion sequences are used to predict future motion, image from [183]. The second row shows human motion synthesis task, image from [261].

current Neural Networks (RNNs) [69, 111, 182, 159, 40], Graph Convolutional Networks (GCNs) [178, 177, 84, 172, 43, 149, 148], and Transformers [177, 5, 26].

RNN-based human motion prediction. Due to the inherent sequential structure of human motion, some works address 3D human motion prediction using recurrent models. [69] proposed an encoder-decoder framework to embed human poses, and an LSTM to update the latent space and predict future motion. [111] manually encoded the semantic similarity between different parts of the body and forwarded them via structural RNNs. However, these two methods suffer from discontinuity, and they are only trained on action-specific models, i.e., a single model is trained for a specific action. [182] studied multi-actions instead of action-specific models, i.e., training a single model for multiple actions, which allows the network to exploit regularities across different actions in large-scale datasets. This approach has been widely adopted by most of the subsequent works. They also introduced a residual connection to model velocities instead of abso-

lute values to have smoother predictions. Nevertheless, the above-mentioned methods suffer from multiple inherent limitations of RNNs. First, as a sequential model, RNNs are difficult to parallelize during training and inference. Second, memory constraints prevent RNNs from exploring information from farther frames. Some works alleviate this problem by using RNN variants [159, 40], sliding windows [22, 23], convolutional models [98, 145], or adversarial training [83], as described in the following sections. However, their networks are still complex and have a large number of parameters.

GCN-based human motion prediction. Recent works on human motion prediction have adopted Graph Convolutional Networks (GCNs) to better encode the spatial connectivity of human joints by building the human pose as a graph. Mao et al. [178] used a stack of blocks consisting of GCNs, non-linear activation, and batch normalization to encode spatial dependencies and DCT to encode temporal information. Lebailly et al. [139] used a multi-scale temporal input embedding, applying various sized convolutional layers for different input sizes to have different receptive fields in the temporal domain. Mao et al. [177] improved temporal encoding by cutting past observations into sub-sequences and using an attention mechanism to find similar previous motion sub-sequences in the past with the current observations. Ma et al. [172] proposed two GCN variants to extract spatial and temporal features and built a multi-stage structure with an encoder and decoder in each stage. During training, the model is trained with intermediate supervision to progressively refine the prediction. Other works [43, 149, 148] extended the graph of human pose to multi-scale versions across the abstraction levels of human pose.

Attention-based human motion prediction. Recently, some works have explored the use of attention mechanisms and transformers for human motion prediction. Mao et al. [177] introduced an attention mechanism to capture temporal relations, while Aksan et al. [5] used a combination of spatial and temporal attention to model pairwise relations between joints. Cai et al. [26] proposed a transformer-based architecture that progressively predicts the DCT coefficients of target joints based on the kinematic tree, using a memory-based dictionary to preserve global motion patterns in the training data. These

approaches offer promising results and show the potential of attention-based models in human motion prediction.

2.3.2 Human Motion Synthesis

Early works in human motion synthesis rose under the task of future motion prediction. Works around this task saw various modeling approaches ranging from sequence to sequence models [69] to graph modeling of each body part [112]. These supervised models were later replaced by generative methods [83, 145] based on Generative Adversarial Networks (GANs) [81]. These approaches tend to diverge from realistic motion and require access to all body joint positions, making them impractical for avatar animation in VR [88]. A second family of motion synthesis methods revolves around character control. In this setting, character motion must be generated according to user inputs and environmental constraints, such as the virtual environment properties. This research direction has practical applications in the field of computer gaming, where controller input is used to guide character motion. Taking inspiration from these constraints, [267] formulated motion synthesis as a control problem by using a GAN architecture that takes direction and speed input into account. Similar efforts are found in [237], where the method learns fast and dynamic character interactions that involve contacts between the body and other objects, given user input from a controller. These methods are impractical in a VR setting, where users want to drive motion using their real body pose instead of a controller.

Diffusion models [235, 100, 197] are a class of likelihood-based generative models based on learning progressive noising and denoising of data. Diffusion models have recently have garnered significant attention in the field of image generation [49] due to their ability to significantly outperform popular GAN architectures [122, 20] and is better suited for handling a large amount of data. Furthermore, diffusion models can support conditional generation, as evidenced by the classifier guidance approach presented in [49] and the CLIP-based text conditional synthesis for diffusion models proposed in [196]. More recently, concurrent works have also extended diffusion models to motion synthesis, with particular focus on the text-to-motion task [294, 126, 244]. However, these

models are both complex in architecture and require multiple iterations at inference time. This hinders them unsuitable for real-time applications like VR body tracking. We circumvent this problem by designing a custom and efficient diffusion model. To the best of our knowledge, we present the first diffusion model solely purposed for solving motion reconstruction from sparse inputs. Our model leverages a simple MLP architecture, runs in real-time, and provides accurate pose predictions, particularly for lower bodies.

2.3.3 Motion Tracking from Sparse Tracking Inputs

The generation of full-body poses from sparse tracking signals of body joints has become an area of considerable interest within the research community. For instance, recent works such as [105] have demonstrated the ability to track full bodies using only 6 IMU inputs and employing a bi-directional LSTM to predict SMPL body joints. Additionally, in [283], a similar approach is used to track with 4 IMU inputs, specifically the head, wrists, and pelvis. However, in the practical HMD setting, only 3 tracking signals are typically available: the head and 2 wrists. In this context, AvatarPoser [117] provides a solution to the 3-point problem through the use of a transformer-based architecture. Other methods attempt to solve sparse input body tracking as a synthesis problem. To that extent, Aliakbarian *et al.* [6] proposed a flow-based architecture derived from [51], while Dittadi *et al.* [52] opted for a Variational Autoencoder (VAE) method. While more complex methods have been developed that involve Reinforcement Learning, as seen in [269, 287], these approaches may struggle to simultaneously maintain accurate upper-body tracking while generating physically realistic motions. In summary, all methods presented in this section either require more than three joints input or face difficulties in accurately predicting full body pose, particularly in the lower body region. Our proposed method, on the other hand, utilizes a custom diffusion model and employs a straightforward MLP-based architecture to predict full body pose with a high degree of accuracy, while utilizing only three IMU inputs.

2.4 Robot Grasping Simulation

The ever-advancing domain of robotics persistently aspires to foster systems that harmoniously blend into human-oriented environments and proficiently accomplish tasks traditionally carried out by humans. An essential element of this assimilation hinges on the robot's capacity to emulate human manipulation skills, especially the grasping and handling of objects. Robot grasp simulation is an area of research focused on the computational modeling and prediction of the success rates of diverse grasp strategies within a simulated, virtual context. This capability to faithfully replicate robot grasping is critical to the efficient implementation of robotic systems across an extensive range of sectors. Successfully simulating robot grasping paves the way for safer, more reliable, and more effective robots, propelling our ability to automate and enhance an array of human-centric tasks. An example of robot grasping simulation is shown in Figure 2.4

2.4.1 Grasp Prediction

Predicting potential grasps for a given object is a classical research topic, as demonstrated by the seminal *GraspIt!* simulator [189]. Recent approaches [170, 271, 253, 167, 157] have predominantly focused on learning-based techniques, with some approaches [239, 271] incorporating reachability constraints in the scene.

2.4.2 Grasping from Demonstration

Learning from demonstration is also an important paradigm in robotics [214, 16, 8, 219]. It aims at teaching a particular task to a robot from a few examples of a human performing a similar task. Most current approaches for learning from demonstration in the context of object manipulation focus on complex manipulation tasks with simple parallel-jaw grippers [296, 230]. On the contrary, we focus in this study on simpler manipulation tasks (static grasping) but with more complex multi-fingered grippers – that could allow more advanced grasps and manipulations.

Figure 2.4: The figure shows an example of the grasp simulation in NVIDIA's Isaac Gym environment, image from [175].

2.4.3 Pose Retargeting

A solution to transfer a human grasp demonstration to a robotic gripper it to define some fixed correspondences between the human hand and the robotic gripper. DexPilot [90] and DexMV [216] use some handcrafted motion retargeting techniques to do so. Similarly, ContactTransfer [138] relies on fixed correspondences between the surface of the human hand and the robotic gripper. The applicability of such approaches is arguably limited however because the human hand and the gripper may have significantly different characteristics in practice.

There are some approaches trying to predict robotic grasps sharing similar contact areas with the object as in the human demonstration, without requiring explicit correspondences between fingers of the human and the robot. [302] proposes to annotate functional parts of the objects – *i.e.*, where humans would grasp the object or not – to generate potential grasps for these objects. Recently, ContactGrasp [18] was proposed and uses GraspIt! [189] to generate a set of grasps that are iteratively refined and reranked such that the contact areas of the gripper on the object become closer to the ones of the human grasp. This approach has several drawbacks however. First, it is about 40 times slower than our method as it has to generate and refine hundreds of grasp candidates each

time. Second, *GraspIt!* mainly generates power grasps, and thus the refined grasps have similar properties. Third, by focusing only on contact areas, ContactGrasp can produce grasps in which the gripper is occluding some important parts for the affordance of the grasped object. In comparison, our proposed optimization approach is faster and leads to grasps more similar to human ones, thanks to a simple yet effective initialization and thanks to additionally taking into account the grasp orientation.

PART I:

SCENE UNDERSTANDING

In this part of the thesis, we tackle the scene understanding problem, with the specific aim of enhancing object localization capabilities in unfamiliar environments - a scenario frequently encountered by robots. Our approach to this challenge is twofold. Firstly, we aim to increase the precision of occlusion boundaries in the anticipated depth, a step that could potentially assist robots in more effectively distinguishing regions of interest from the undesired background. Secondly, we aspire to boost the quality of detection in scenarios involving previously unseen classes. This would equip robots with the capacity to identify objects from images or videos, even when such objects were not part of the original training dataset.

This part includes two chapters, in the Chapter 3, we introduce a generalist method to improve monocular depth estimation method, instead of relying on various forms of filtering or predict an additive residual depth map, we learn to predict, given a depth map predicted by some reconstruction method, a 2D displacement field able to re-sample pixels around the occlusion boundaries into sharper reconstructions. In addition, for evaluation, we manually annotated a high-quality occlusion boundaries dataset.

In the Chapter 4, we propose a Bayesian framework to improve the instance segmentation methods using only unlabeled videos. Specifically, our framework is designed to automatically create such a training set: Our method starts from a set of object proposals and relies on (non-realistic) analysis-by-synthesis to select the correct ones by performing an efficient optimization over all the frames simultaneously. Our method can generate a high-quality training set which significantly boosts the performance of segmenting objects of unseen classes.

CHAPTER 3

DISPLACEMENT FIELDS FOR DEPTH REFINEMENT

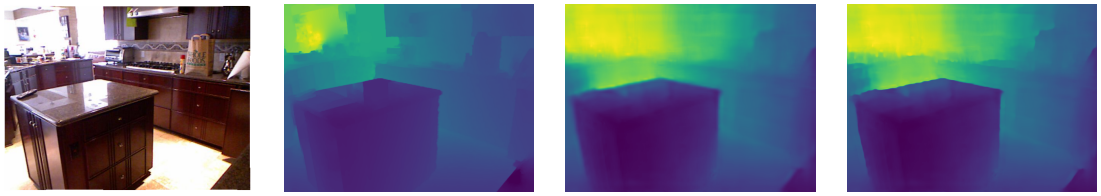
In this chapter, we introduce our innovative method for predicting a corrective two-dimensional displacement field for depth refinement. This approach aims to re-sample pixels in the vicinity of occlusion boundaries, resulting in sharper, more precise reconstructions. Occlusion boundaries serve as crucial indicators for object recognition, providing significant insights into the structure and position of the object. As we will demonstrate, harnessing these insights could potentially pave the way for the discovery of new objects within the reconstructed scene.

3.1 INTRODUCTION

Monocular depth estimation (MDE) is a task that involves predicting a depth map from a single input image. MDE has attracted significant interest due to its potential applications in various computer vision tasks, such as scene understanding, robotic grasping, and augmented reality. In recent years, Deep Learning approaches have been extensively used to tackle this problem, with many supervised learning-based methods proposed, such as [60, 137, 70, 59] using deep convolutional neural networks. Self-learning-based methods, including [79, 279, 213], have also been developed. These approaches have shown impressive results, making MDE a highly active research field.

Despite recent advances in Monocular Depth Estimation, as shown in Figure 3.1, the reconstruction of occlusion boundaries in predicted depth maps remains a challenging task. These boundaries correspond to depth discontinuities that occur along object silhouettes [123, 240]. Accurate reconstruction of these contours is crucial for applications such as handling partial occlusions between real and virtual objects in Augmented Reality (AR), and for object understanding, as illustrated in Figure 3.2. Therefore, we consider this direction of research particularly important. MDE has shown good generalization performance to unseen objects and categories, and improving occlusion boundary reconstruction could lead to promising research in unsupervised object discovery.

In our work, we present a novel approach for enhancing smooth occlusion boundaries in images. Our method not only enhances the sharpness of these boundaries but also im-



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

Figure 3.1: The figure shows the results of our occlusion boundary (OB) refinement approach. (a) displays the input image. (b) shows the ground truth depth from NYUv2-Depth. (c) shows the predicted depth using the approach described in [222]. Finally, (d) presents the refined depth using our pixel displacement method.

proves their localization. Our approach leverages a differentiable module that takes an initial depth map produced by an existing depth prediction method and re-samples it to obtain more precise occlusion boundaries. Additionally, our method can incorporate a color image as input to provide further guidance information and achieve even better contour localization. To achieve this, we train a deep network to predict a 2D displacement field, which is applied to the initial depth map. This is in contrast to previous approaches, such as those described in [115, 297], which predict residual offsets for depth values. Our experiments demonstrate that predicting displacements instead of residuals produces sharper occluding boundaries. Our approach is complementary to existing methods, and our experiments show that it consistently improves the localization and reconstruction of occlusion boundaries.

To evaluate the performance of existing MDE methods and our proposed method in reconstructing occlusion boundaries, we manually annotated the occlusion boundaries in all images of the NYUv2 test set. Selected annotations can be found in Fig. 3.3. We used the metrics introduced by [131] to assess the accuracy of occlusion boundary reconstruction and localization in predicted depth maps. Our experimental results demonstrate that our proposed method quantitatively improves the localization accuracy of all state-of-the-art MDE methods, while maintaining or even improving their overall depth reconstruction performance on two benchmark datasets.

In the rest of this chapter, we begin by presenting our proposed approach for enhanc-

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

Figure 3.2: Application of our depth map refinement to 3D object extraction. (a-b) and (c-d) are two point cloud views of our extracted object. The left column shows point clouds extracted from the initially predicted depth image while the right one shows the result after using our depth refinement method. Our method suppresses long tails around object boundaries, as we achieve sharper occlusion boundaries.

ing occlusion boundaries in depth maps. Next, we provide detailed descriptions of our experiments and results, demonstrating the effectiveness of our method in improving the performance of state-of-the-art MDE methods. Please refer to our project page for video and poster.

3.2 Displacement Fields

In this section, we present our occlusion boundary refinement method. Firstly, we hypothesize that the structure of predicted depth maps around occlusion boundaries should have a certain pattern, and we derive a model that can transform this structure into the expected one. We then verify our hypothesis using a hand-crafted method. Based on this model, we propose an end-to-end trainable module that can resample the pixels of an input depth map to restore its sharp occlusion boundaries.

Occlusion boundaries are characterized by regions in the image where the depth exhibits sharp and large variations, while other regions tend to vary much more smoothly. Due to the relatively small proportion of such sharp regions, neural networks often predict over-smoothed depths in the vicinity of occlusion boundaries.

We propose a method to recover sharp and accurately located occlusion boundaries by resampling pixels in the predicted depth map. This resampling can be formalized as:

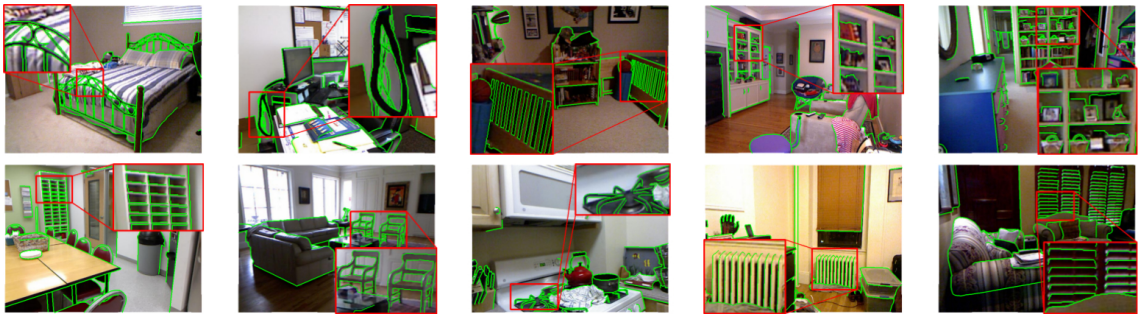


Figure 3.3: Samples of our NYUv2-OC++ dataset, which extends NYUv2-OC from [222]. The selected highlighted regions in red rectangles emphasize the high-quality and fine-grained annotations.

$$\forall \boldsymbol{p} \in \Omega, \quad D(\boldsymbol{p}) \leftarrow D(\boldsymbol{p} + \boldsymbol{\delta p}(\boldsymbol{p})) \,, \tag{3.1}$$

where D is a depth map, $\boldsymbol{p}$ denotes an image location in domain Ω, and $\boldsymbol{\delta p}(\boldsymbol{p})$ a 2D displacement that depends on $\boldsymbol{p}$. This formulation enables the stitching together of depth values on both sides of occlusion boundaries, replacing the over-smoothed depth values.

Another option to improve depth values would be to predict an additive residual depth, which can be formalized as, for comparison:

$$\forall \boldsymbol{p} \in \Omega, \quad D(\boldsymbol{p}) \leftarrow D(\boldsymbol{p}) + \Delta D(\boldsymbol{p}) \,. \tag{3.2}$$

We argue that updating the predicted depth $\hat{D}$ using predicted pixel shifts to recover sharp occlusion boundaries in depth images works better than predicting the residual depth. We validate this assertion through experiments presented below on toy problems and on real predicted depth maps in Section 3.5.

To first validate our assumption that a displacement field $\boldsymbol{\delta p}(\boldsymbol{p})$ can improve the reconstructions of occlusion boundaries, we estimate the optimal displacements using ground truth depth for several predicted depth maps as:

$$\forall \boldsymbol{p} \in \Omega, \boldsymbol{\delta p}^* = \underset{\substack{\boldsymbol{\delta p}: \\ \boldsymbol{p}+\boldsymbol{\delta p} \in \mathcal{N}(\boldsymbol{p})}}{\operatorname{argmin}} (D(\boldsymbol{p}) - \widehat{D}(\boldsymbol{p} + \boldsymbol{\delta p}))^2 \,. \tag{3.3}$$

In simpler terms, the problem can be solved by finding the optimal displacement $\boldsymbol{\delta p}^*$ for

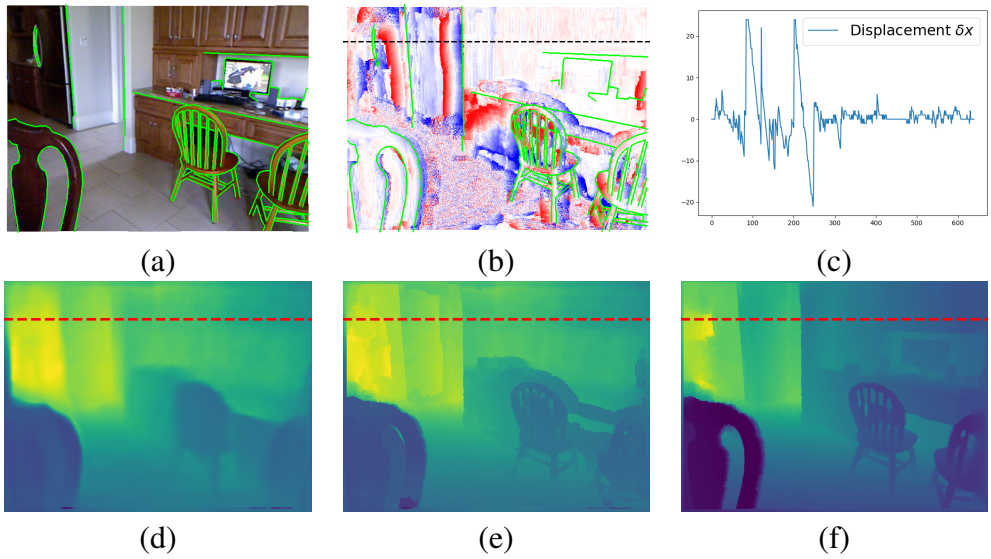


Figure 3.4: Refinement results using the gold standard method described in Section 3.2 to recover the optimal displacement field (best seen in color). (a) is the input RGB image with superimposed NYUv2-OC++ annotation in green and (d) its associated Ground Truth depth. (e) is the prediction using [137] with pixel displacements $\boldsymbol{\delta p}$ from Eq. (3.3) and (f) the refined prediction. (b) is the horizontal component of the displacement field $\boldsymbol{\delta p}^*$ obtained by Eq. (3.3). Red and blue color indicate positive and negative values respectively. (c) is the horizontal component δx of displacement field $\boldsymbol{\delta p}^*$ along the dashed red line drawn in (b,c,d,e).

each pixel that reconstructs the ground truth depth map D from a predicted depth map $\widehat{D}$. To achieve this, we solve Eq. (3.3) by performing an exhaustive search of $\boldsymbol{\delta p}$ for all pixels $\boldsymbol{p}$ within a neighborhood $\mathcal{N}(\boldsymbol{p})$ of size 50×50. Qualitative results are shown in Fig. 3.4. The depth map obtained by applying this optimal displacement field is clearly much better.

Drawing from our model, we suggest learning the pixel displacements in predicted depth images using CNNs. The pipeline of our approach is demonstrated in Fig. 3.7: Given a predicted depth image $\widehat{D}$, our network predicts a displacement field $\boldsymbol{\delta p}$ to resample the image locations in depth map $\widehat{D}$ according to Eq. (3.1). This approach can be implemented with the help of the Spatial Transformer Network [110]. Using image guidance can potentially enhance the precision of occlusion boundaries in refined depth maps and also enable the detection of edges that were not visible in the initial predicted depth

map $\widehat{D}$. Nonetheless, it is worth mentioning that our network can still operate without image guidance.

3.3 Simplified 1D Problem

To confirm that the displacement fields $\boldsymbol{\delta p}$ outlined in Section 3.2 can be learned, we begin by defining a toy problem in 1D. In this 1D toy problem, as depicted in Fig. 3.5, we model the signals D to be reconstructed as piecewise continuous functions, generated as sequences of basic functions such as step, affine, and quadratic functions. These samples demonstrate significant discontinuities at junctions and smooth variations elsewhere, which is a feature similar to real depth maps. We subsequently convolve the D signals with randomly-sized Gaussian kernels (blurring) to obtain their smooth versions $\widehat{D}$. This gives us a training set $\mathcal{T}$ of $(\widehat{D}, D)$ pairs.

We use $\mathcal{T}$ to train a network $f(.;\Theta_f)$ of parameters to predict a displacement field:

$$\min_{\Theta_f} \sum_{(\widehat{D},D)\in\mathcal{T}} \sum_{\boldsymbol{p}} L\left(D(\boldsymbol{p}) - \widehat{D}\left(\boldsymbol{p} + f(\widehat{D};\Theta_f)(\boldsymbol{p})\right)\right) . \tag{3.4}$$

and a network $g(.;\Theta_g)$ of parameters to predict a residual depth map:

$$\min_{\Theta_g} \sum_{(\widehat{D},D)\in\mathcal{T}} \sum_{\boldsymbol{p}} L\left(D(\boldsymbol{p}) - \widehat{D}(\boldsymbol{p}) + g(\widehat{D};\Theta_g)(\boldsymbol{p})\right) , \tag{3.5}$$

where $L(.)$ is some loss. In our experiments, we evaluate the l_1, l_2, and Huber losses.

As depicted in Fig. 3.5, we observed that predicting a residual update causes severe artifacts around edges, such as overshooting effects. We contend that these problems arise because the residual CNN g is trained on regions where the values of $\hat{D}$ and D differ, even away from edges, thus promoting network g to also correct these regions. Conversely, our method does not produce overshooting effects as it does not modify the local range of values around edges.

It is worth noting that even when we permit $\widehat{D}$ and D to have slightly different values

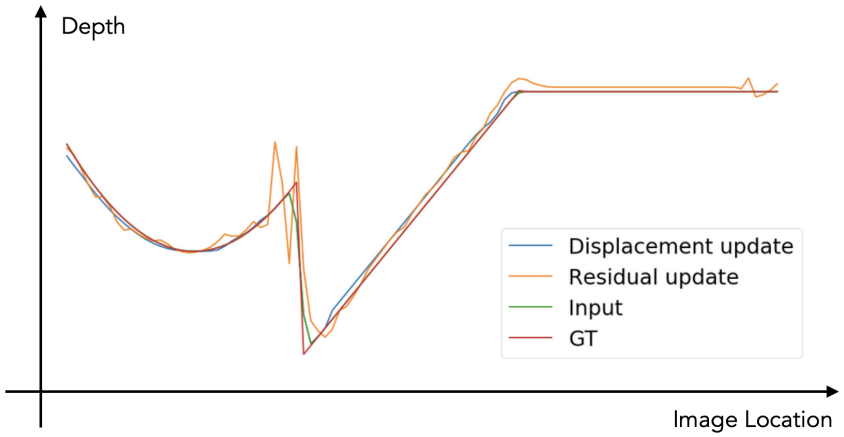


Figure 3.5: Comparison between displacement and residual update learning. Both residual and displacement learning can predict sharper edges, however residual updates often produce artifacts along the edges while displacements update does not.

in non-edge regions - which simulates residual error between predicted and ground truth depth, our method still converges to a satisfactory solution when contrasted to the residual CNN.

We extend our approach validation from 1D to 2D, where the 1D signal is substituted with 2D images composed of various polygons of different values. We perform the same operation of smoothing the images and then employ our network to retrieve the original sharp images from the smoothed ones. We observed analogous results in 2D: The residual CNN consistently produces artifacts.

To learn how to generate sharper depth predictions using displacement fields, we first trained our method in a similar fashion to the toy problem described in Section 3.3. While this already improves the quality of occlusion boundaries of all depth map predictions, we demonstrate that we can achieve further improvement in quantitative results by training our method using predictions from an MDE algorithm on the NYUv2-Depth dataset as input and the corresponding ground truth depth as target output. We contend that this way, the network learns to correct more complex distortions of the ground truth than Gaussian blurring. We used the predictions of [222] to demonstrate this. We demonstrate that this not only enhances the quality of depth maps predicted by [222], but also improves the

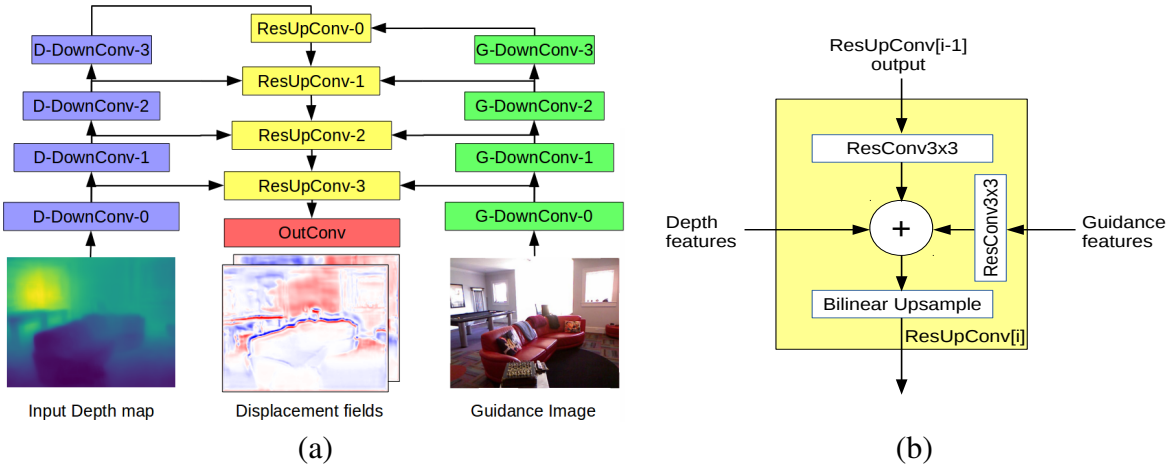


Figure 3.6: (a) Detailed architecture of our displacement field prediction network. Details on the Depth and Guidance encoders are provided in Sections 3.4.1 and 3.4.2 respectively. (b) Details of the ResUpConv block used in our displacement field decoder.

performance of all other available MDE algorithms. Although training our network using the predictions of other algorithms on the NYUv2 could further enhance their results, not all of them provide their code or their predictions on the official *training set*. Lastly, our method is fully differentiable.

3.4 Network Architecture

In Fig. 3.6, we elaborate on the architecture of our network, which comprises two encoders, one for Depth and an optional one for Guidance. We adopt a single decoder that combines the corresponding outputs of the Depth and Guidance encoders using residual blocks and skip connections. We provide complete details of each block in the subsequent sections.

3.4.1 Depth Encoder

Our Depth encoder follows a standard architecture with a sequence of four down-convolutions, labeled *D-DownConv*. The D-DownConv blocks consist of a convolution layer with a 3x3 kernel, followed by a 2x2 MaxPooling layer, and a LeakyReLU [173] activation function.

The convolution layers of the D-DownConv blocks have 32, 64, 128, and 256 channels, respectively. All blocks employ batch normalization, are initialized using Xavier [77] initialization, and use a LeakyReLU [173] activation function.

3.4.2 Guidance Encoder

Our Guidance encoder is comprised of a series of four down-convolutions, as in [95], labeled G-DownConv. It is similar to the D-DownConv block described in 3.4.1, except that it uses ReLU [78] activations and batch normalization for the convolution layers. The convolution layers have 32, 64, 128, and 256 channels, respectively.

3.4.3 Displacement Field Decoder

The displacement field decoder is composed of a sequence of four ResUpConv blocks described in Fig 3.4.4, followed by a convolution block OutConv.

3.4.4 ResUpConv

The ResUpConv block is the primary component of our decoder. It combines depth and guidance features at multiple scales using skip connections. The architecture of this block is described in Fig 3.6. Guidance features are refined using a 3x3 residual convolution layer [95], denoted ResConv3x3. All blocks use batch normalization, LeakyReLU [173] activation, and filter weights are initialized using Xavier initialization.

3.4.5 Output Layers

The final output block, OutConv, comprises two Conv3x3 layers with batch normalization and ReLU [78] activation, followed by a simple 3x3 convolution layer without batch normalization or activation. The number of channels for these layers are 32, 16, and 2, respectively. The weights are initialized using Xavier initialization.

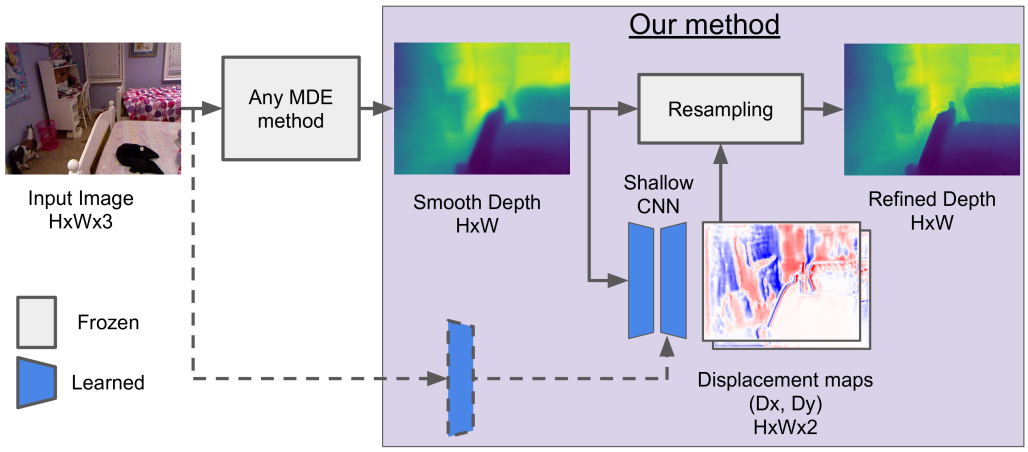


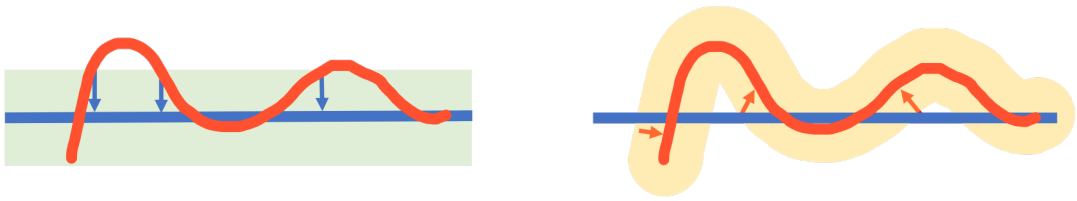
Figure 3.7: Our proposed pipeline for depth edge sharpening. The dashed lines define the optional guidance with RGB features for our shallow network.

3.5 Experiments

In this section, we begin by detailing our implementation and outlining the metrics used to evaluate the reconstruction of occlusion boundaries and the accuracy of depth prediction. We then present the results of the evaluation of our method on the outputs of various MDE methods.

3.5.1 Implementation Details

We implemented our network using the Pytorch framework [205]. We trained our network on the output of a state-of-the-art MDE method [222], using Adam optimization with an initial learning rate of 5e-4 and weight decay of 1e-6, and the *poly* learning rate policy [298], for 32k iterations on the NYUv2 dataset [233]. This dataset consists of 1449 pairs of RGB and depth images, split into 795 samples for training and 654 for testing, with a batch size of 1. The input images were resized with scales $[0.75, 1, 1.5, 2]$ and then cropped and padded to 320×320. We attempted to learn on the *raw* depth maps from the NYUv2 dataset, but it was unsuccessful. This is likely due to the fact that missing data occurs mostly around occlusion boundaries. Therefore, dense depth maps are required, and we use [144] to inpaint the missing data in the raw depth maps.



(a) (b)

Figure 3.8: Occlusion boundary evaluation metrics based on the Chamfer distance, as introduced in [131]. The blue lines represent the ground truth boundaries, the red curve the predicted boundary. Only the boundaries in the green area are taken into account during the evaluation of accuracy (a), and only the yellow area are taken into account during the evaluation of completeness (b). (a) The accuracy is evaluated from the distances of points on the predicted boundaries to the ground truth boundaries. (b) The completeness is evaluated from the distances of points on the ground truth boundaries to the predicted boundaries.

3.5.2 Evaluation of monocular depth prediction

As in previous work [60, 59, 137], we assess the quality of monocular depth predictions using the following metrics: Root Mean Squared Linear Error (R_{lin}), mean absolute relative error (rel), mean log10 error (log10), Root Mean Squared Log Error (R_{log}), and the accuracy under threshold ($\sigma_i < 1.25^i$, $i = 1, 2, 3$).

3.5.3 Evaluation of occlusion boundary accuracy

To evaluate the accuracy of occlusion boundaries (OB), we adopt the depth boundary errors proposed by Koch *et al.* [131]. The boundaries are first extracted using a Canny edge detector [28] with predefined thresholds on a normalized predicted depth image. Following the method of Koch *et al.*, we evaluate the accuracy ϵ_a and completion ϵ_c of predicted occlusion boundaries. Specifically, ϵ_a is calculated as the average Chamfer distance in pixels [64] from the predicted boundaries to the ground truth boundaries, while ϵ_c is calculated as the average Chamfer distance from the ground truth boundaries to the predicted boundaries. These computations are illustrated in Fig. 3.8.

Method	Refine	Depth error (↓)				Depth accuracy (↑)			OBs (↓)	
		rel	$\log_{10}$	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Eigen *et al.* [60]	-	0.234	0.095	0.760	0.265	0.612	0.886	0.971	9.936	9.997
	✓	0.232	0.094	0.758	0.263	0.615	0.889	0.971	2.168	8.173
Laina *et al.* [137]	-	0.142	0.059	0.510	0.181	0.818	0.955	0.988	4.702	8.982
	✓	0.140	0.059	0.509	0.180	0.819	0.956	0.989	2.372	7.041
Fu *et al.* [70]	-	0.131	0.053	0.493	0.174	0.848	0.956	0.984	3.872	8.117
	✓	0.136	0.054	0.502	0.178	0.844	0.954	0.983	3.001	7.242
Ramamonjisoa and Lepetit [222]	-	0.116	0.053	0.448	0.163	0.853	0.970	0.993	3.041	8.692
	✓	0.117	0.054	0.457	0.165	0.848	0.970	0.993	1.838	6.730
Jiao *et al.* [119]	-	0.093	0.043	0.356	0.134	0.908	**0.981**	**0.995**	8.730	9.864
	✓	**0.092**	**0.042**	**0.352**	**0.132**	**0.910**	**0.981**	**0.995**	2.410	8.230
Yin *et al.* [290]	-	0.112	0.047	0.417	0.144	0.880	0.975	0.994	1.854	7.188
	✓	0.112	0.047	0.419	0.144	0.879	0.975	0.994	**1.762**	**6.307**

Table 3.1: Evaluation of our method on the output of several state-of-the-art methods on NYUv2. Our method significantly improves the occlusion boundaries metrics ϵ_a and ϵ_c without degrading the other metrics related to the overall depth accuracy. These results were computed using available depth maps predictions (apart from Jiao *et al.* [119] who sent us their predictions) within the image region proposed in [60]. (↓: Lower is better; ↑: Higher is better).

NYUv2 dataset We evaluate the performance of our method by refining the predictions of several state-of-the-art methods [60, 137, 70, 222, 119, 290]. Our network is trained on the NYUv2 depth dataset, which contains 795 labeled images for training, along with their corresponding RGB images used as guidance. To enable a fair comparison, we evaluate only the pixels inside the crop defined in [60] for all methods. Table 3.1 presents the evaluation of refined predictions of various methods using our network. Our proposed network significantly improves the occlusion boundary accuracy of all methods without compromising the global depth estimation accuracy. We also include qualitative results of our refinement method in Fig. 3.9.

iBims dataset We applied our method, which was trained on the NYUv2 dataset, to refine the predictions of various methods [60, 59, 137, 156, 147, 155, 222] on the iBims dataset [131]. Our results, presented in Table 3.2, show that our network significantly improves the accuracy and completeness metrics for the occlusion boundaries of all predictions on this dataset as well.

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

Figure 3.9: Refinement results using our method (best seen in color). From left to right: (a) input RGB image with NYUv2-OC++ annotation in green, (b) SharpNet [222] depth prediction, (c) Refined prediction, (d) Ground truth depth, (e) Horizontal and (f) Vertical components of the displacement field. Displacement fields are clipped between ± 15 pixels. Although SharpNet is used as an example here because it is currently state-of-the-art on occlusion boundary accuracy, similar results can be observed when refining predictions from other methods.

3.5.4 Comparison with Other Methods

To demonstrate the efficiency of our proposed method, we compare it with existing filtering methods [247, 94, 13, 272]. We use the predictions of [60] as input and compare the accuracy of depth estimation and occlusion boundaries for each method. For filters with hyperparameters, we test each filter with a range of hyperparameters and select the best-refined results. For the Fast Bilateral Solver (FBS) [13] and the Deep Guided Filter (GF) [272], we use their default settings from the official implementation. We keep the same network architecture for both the cases with and without Deep GF and train them with the same learning rate and data augmentation.

As shown in Table 3.3, our method achieves the best accuracy for occlusion boundaries. Finally, we compare our method against the additive residual prediction method discussed in Section 3.2. We keep the same U-Net architecture, but replace the displacement operation with an addition as described in Eq. (3.2), and show that we obtain better results. We argue that the performance of the deep guided filter and additive residual are lower due to generated artifacts which are discussed in Section 3.3. In Table 3.4, we also

Method	Refine	Depth error (↓)			Depth accuracy (↑)			OB (↓)	
		rel	$\log_{10}$	R_{lin}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Eigen *et al.* [60]	-	0.32	0.17	1.55	0.36	0.65	0.84	9.97	9.99
	✓	0.32	0.17	1.54	0.37	0.66	0.85	4.83	8.78
Eigen and Fergus (AlexNet) [59]	-	0.30	0.15	1.38	0.40	0.73	0.88	4.66	8.68
	✓	0.30	0.15	1.37	0.41	0.73	0.88	4.10	7.91
Eigen and Fergus (VGG) [59]	-	0.25	0.13	1.26	0.47	0.78	0.93	4.05	8.01
	✓	0.25	0.13	1.25	0.48	0.78	0.93	3.95	7.57
Laina *et al.* [137]	-	0.26	0.13	1.20	0.50	0.78	0.91	6.19	9.17
	✓	0.25	0.13	1.18	0.51	0.79	0.91	3.32	7.15
Liu *et al.* [156]	-	0.30	0.13	1.26	0.48	0.78	0.91	2.42	7.11
	✓	0.30	0.13	1.26	0.48	0.77	0.91	2.36	7.00
Li *et al.* [147]	-	**0.22**	**0.11**	1.09	0.58	**0.85**	**0.94**	3.90	8.17
	✓	**0.22**	**0.11**	1.10	0.58	0.84	**0.94**	3.43	7.19
Liu *et al.* [155]	-	0.29	0.17	1.45	0.41	0.70	0.86	4.84	8.86
	✓	0.29	0.17	1.47	0.40	0.69	0.86	2.78	7.65
Ramamonjisoa and Lepetit [222]	-	0.27	**0.11**	**1.08**	**0.59**	0.83	0.93	3.69	7.82
	✓	0.27	**0.11**	**1.08**	**0.59**	0.83	0.93	**2.13**	**6.33**

Table 3.2: Evaluation of our method on the output of several state-of-the-art methods on iBims dataset.

compare the computational efficiency of the network against reference depth estimation and refinement methods.

3.5.5 Loss Functions for Depth Prediction

In Table 3.5, we show the influence of different loss functions. We apply the Pytorch official implementation of l_1, l_2, and the Huber loss. The Disparity loss supervises the network with the reciprocal of depth, the target depth y_{target} is defined as $y_{target} = M/y_{original}$, where M here represents the maximum of depth in the scene. As shown in Table 3.5, our network trained with l_1 loss achieves the best accuracy for the occlusion boundaries.

	Depth error (↓)				Depth accuracy (↑)			OB (↓)	
Method	rel	log10	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Baseline [60]	0.234	0.095	0.766	0.265	0.610	0.886	*0.971*	9.926	9.993
Bilateral Filter [247]	0.236	0.095	0.765	0.265	0.611	0.887	*0.971*	9.313	9.940
GF [94]	0.237	0.095	0.767	0.265	0.610	0.885	*0.971*	6.106	9.617
FBS [13]	0.236	0.095	0.765	0.264	0.611	0.887	*0.971*	5.428	9.454
Deep GF [272]	0.306	0.116	0.917	0.362	0.508	0.823	0.948	4.318	9.597
Residual	0.286	0.132	0.928	0.752	0.508	0.807	0.931	5.757	9.785
Our Method	**0.232**	**0.094**	**0.757**	**0.263**	**0.615**	**0.889**	*0.971*	**2.302**	**8.347**

Table 3.3: Comparison with existing methods for image enhancement, adapted to the depth map prediction problems. Our method performs the best for this problem over all the different metrics.

Method	SharpNet [222]	VNL [290]	Deep GF[272]	Ours
FPS - GPU	83.2 $\pm$ 6.0	32.2$\pm$ 2.1	70.5 $\pm$ 7.5	**100.0** $\pm$ 7.3
FPS - CPU	2.6 $\pm$ 0.0	*	4.0 $\pm$ 0.1	**5.3** $\pm$ 0.15

Table 3.4: Speed comparison with other reference methods implemented using Pytorch. Those numbers were computed using a single GTX Titan X and Intel Core i7-5820K CPU. using 320x320 inputs. Runtime statistics are computed over 200 runs.

3.5.6 Guidance image

We investigate the impact of different types of guidance images. To extract the features of the guidance images, we employ an encoder with the same architecture as the depth encoder, but with all Leaky ReLU activations replaced by standard ReLU activations [78]. We fuse the guidance and depth features using skip connections from the guidance and depth decoder, respectively, at comparable scales.

Table 3.6 presents the impact of different choices for the guidance image. The edge images are generated by accumulating the detected edges using a series of Canny detectors with varying thresholds. As depicted in Table 3.6, using the original RGB image as guidance yields the highest accuracy. Conversely, using the grayscale image as guidance produces the lowest accuracy since information is lost during the conversion process. However, using the Canny edge detector can help mitigate this issue, as the network

	Depth error (↓)				Depth accuracy (↑)			OB (↓)	
Method	rel	log10	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Baseline [60]	0.234	0.095	0.766	0.265	0.610	0.886	0.971	9.926	9.993
l_1	*0.232*	*0.094*	0.758	*0.263*	*0.615*	*0.889*	0.971	**2.168**	**8.173**
l_2	*0.232*	*0.094*	**0.757**	*0.263*	*0.615*	*0.889*	0.971	2.302	8.347
Huber	*0.232*	0.095	0.758	*0.263*	*0.615*	*0.889*	**0.972**	2.225	8.282
Disparity	0.234	0.095	0.761	0.264	0.613	0.888	0.971	2.312	8.353

Table 3.5: Evaluation of different loss functions for learning the displacement field. The l_1 norm yields the best results.

	Depth error (↓)				Depth accuracy (↑)			OB (↓)	
Method	rel	log10	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Baseline [60]	0.234	0.095	0.760	0.265	0.612	0.886	0.971	9.936	9.997
No guidance	0.236	0.096	0.771	0.268	0.608	0.883	0.969	6.039	9.832
Gray	*0.232*	*0.094*	*0.757*	*0.263*	*0.615*	*0.889*	*0.972*	2.659	8.681
Binary Edges	*0.232*	*0.094*	*0.757*	*0.263*	*0.615*	*0.889*	*0.972*	2.466	8.483
RGB	*0.232*	*0.094*	0.758	*0.263*	*0.615*	*0.889*	0.971	**2.168**	**8.173**

Table 3.6: Evaluation of different ways of using the input image for guidance. Simply using the original color image works best.

obtains superior results when switching from grayscale images to binary edge maps.

3.6 Conclusion

We have demonstrated that by predicting a displacement field to resample depth maps, we can substantially enhance the reconstruction accuracy and the localization of occlusion boundaries for any existing method of monocular depth prediction. Additionally, we have created a new dataset that provides precisely labeled occlusion boundaries to evaluate our approach. Beyond evaluating occlusion boundary reconstruction, this dataset should be an extremely valuable resource for future methods to learn more accurate occlusion boundary detection.

CHAPTER 4

GLOBAL OPTIMISATION FOR UNSEEN CLASS INSTANCE SEGMENTATION

As shown in Chapter 3, Occlusion boundaries offer valuable cues for object discovery. However, their utility can be limited in situations where multiple objects are in close proximity in the foreground, necessitating methods capable of distinct instance separation. In this chapter, we put forward a novel Bayesian approach aimed at enhancing instance segmentation methods for unseen classes using unlabeled videos. Our method starts with a set of object proposals and employs an analysis-by-synthesis approach to select the correct ones. This process involves an efficient optimization carried out across all frames simultaneously. By employing this strategy, we demonstrate that our method can generate a high-quality training set, which considerably enhances the performance of segmenting objects from previously unseen classes. Thus, we believe that our method could open the door for open-world instance segmentation using abundant Internet videos.

4.1 Introduction

Instance segmentation models are capable of predicting the masks of objects of known classes in query images [93, 245, 265], providing valuable information for many downstream applications, such as scene understanding [76, 229] and robot grasping [259, 277, 288]. However, existing instance segmentation methods have limited performance on new classes [48]. This is a significant obstacle to the development of autonomous systems operating in open worlds, where there will always be objects that do not belong to known classes. The ability to detect and segment these objects would be a crucial starting point for learning to grasp and manipulate them.

As depicted in Figure 4.1, our objective is to enhance the performance of instance segmentation models on static images containing objects from new classes automatically, without human intervention. This is distinct from previous works that aim to reduce the manual labeling burden for object segmentation by solely employing bounding boxes [104, 134, 300] or developing few-shot techniques [65, 282], which still necessitate human intervention for new classes.

More precisely, we do *not* aim to predict the categories of new objects, but instead

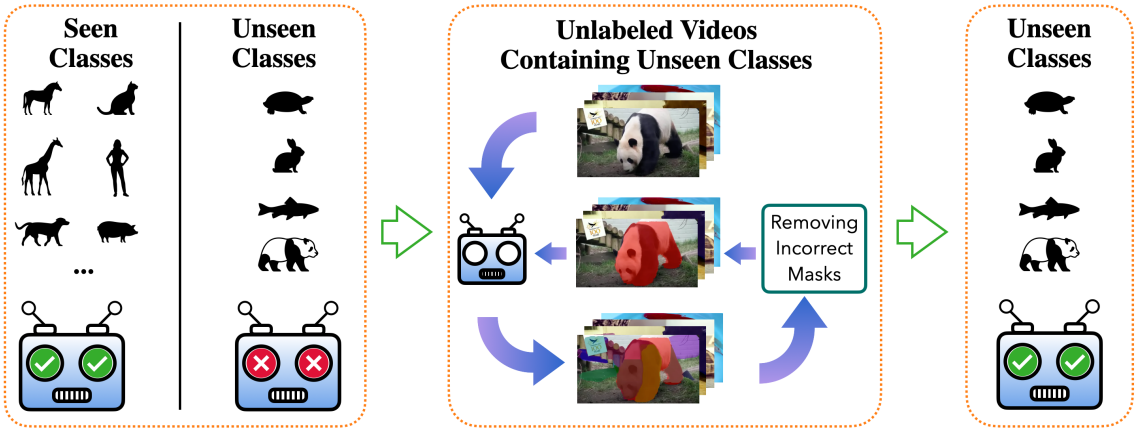


Figure 4.1: Our objective is to train an instance segmentation model to localize and segment objects from new classes without human labeling, beginning with a model trained on a specific set of classes. To achieve this, we employ unlabeled videos, which provide an abundant source of data. Our method automatically detects and selects object masks in the videos. We subsequently use the selected masks to retrain the initial model, which enables it to localize and segment objects from the new classes in still frames without losing performance on the old ones.

concentrate on robustly localizing and accurately segmenting them. In this sense, our work is more closely related to recent object discovery methods that strive to segment objects without manual segmentation labels by grouping pixels based on certain criteria [21, 82, 160, 211, 270]. However, these approaches are still vulnerable to low-level perturbations in color, lighting, or texture, making them rather fragile.

Our approach relies on unlabeled video sequences since such videos can be obtained effortlessly while providing rich information. The concept of using videos for self-supervision is not novel [44, 114, 151, 168, 264, 268]. In this work, we utilize video sequences to automatically generate masks for the objects present in their frames. Employing these masks to train an instance segmentation model should enhance its performance on new objects visible in the videos, even in the absence of human intervention. Although we provide the videos in our experiments, it is possible to envision a system that captures videos autonomously.

Unfortunately, our preliminary experiments revealed that state-of-the-art video segmentation methods [63, 169, 204, 151, 255] were inadequate for our purposes. Con-

sequently, we developed our own method for automatically creating object masks from videos. It should be noted that our task is slightly different from video segmentation methods, which aim to keep track of objects across consecutive frames. Our goal, on the other hand, is to precisely localize and segment the objects in each frame. Similar to some video segmentation methods [169], we commence with mask hypotheses for the objects visible in the videos, obtained via a pre-trained class-agnostic instance segmentation. This model generalizes to some extent to objects from unseen classes, but at the cost of introducing numerous false positives [201]. Figure 4.2 demonstrates that when the confidence threshold is lowered, both the detections of objects from unseen classes and incorrect detections are accepted.

However, it is possible to filter the incorrect masks using information provided by the videos. Some methods [38, 146, 169, 187, 209] rely on tracklets to track and filter the masks. We argue that such an approach is suboptimal, particularly for our objective, for two reasons: firstly, image background is underutilized despite being valuable for indicating the *absence* of objects; secondly, optical flow is not employed to its full potential, despite providing strong cues about moving objects in videos. To fully leverage the background and motion information across unlabeled video frames, we developed a (non-realistic) analysis-by-synthesis approach. We derive an objective function within a Bayesian framework, incorporating a non-overlapping constraint. The objective function comprises three loss terms and is designed to explore the background and motion information to remove incorrect masks and select masks that are temporally consistent throughout the video. The non-overlapping constraint acknowledges that one pixel can belong to, at most, one object in the image and helps to eliminate some false positives. Furthermore, we present a two-stage optimization algorithm to optimize this objective function efficiently.

To assess the efficacy of our approach, we created a novel dataset named Unseen-VIS based on the YouTube Video Instance Segmentation (YouTube-VIS) dataset [284], which contains objects that are *not* part of COCO classes. We began with the raw masks generated on the training portion of Unseen-VIS using a class-agnostic Mask R-CNN

Score > 1.0 Score > 0.5 Score > 0.1 Score > 0.01

Figure 4.2: The class 'tortoise' does not belong to COCO classes. By decreasing the confidence score threshold of Mask R-CNN [93] trained on COCO, we can eventually localize and segment all the tortoises at the price of introducing many false positives. We filter these false positives using unlabeled video data.

pretrained on the COCO dataset and applied our method to automatically select the appropriate masks. We demonstrate that utilizing these masks enhances the performance of Mask R-CNN on the test set of Unseen-VIS without compromising performance on COCO classes. Please refer to our project page for more videos.

To summarize our contributions:

- We propose a Bayesian method to generate high-quality masks on unlabeled videos containing unseen classes;
- We create a benchmark to evaluate the quality of generated masks on unlabeled videos;
- We demonstrate on our benchmark that our proposed method can be used to improve the performance of an instance segmentation model on unseen classes.

4.2 Global Optimization

As discussed in Section 4.1, our goal is to improve the performance of a pre-trained class-agnostic instance segmentation on unseen classes. Our pipeline consists of three steps:

- Mask Generation: We use our baseline instance segmentation network on unlabeled videos containing unseen classes for mask generation;
- Mask Selection: We apply our method to automatically select the correct masks on unlabeled videos;
- Model Refinement: We use our generated masks to fine-tune or retrain our baseline network to boost its performance on unseen classes.

In this section, we present our baseline instance segmentation network and our approach for automatically selecting high-quality masks by exploring video information. As we will demonstrate in the following sections, our approach is highly efficient compared to exhaustive search and requires few easy-to-tune hyper-parameters.

4.2.1 Baseline Network for Mask Generation

To generate masks from unlabeled videos, we utilize a class-agnostic Mask R-CNN [93] with a ResNet-50-FPN [154] backbone as our baseline network. As in previous work [201], we refer to this class-agnostic Mask R-CNN as "MP R-CNN" for Mask Proposal R-CNN, as it generates mask proposals regardless of object classes. Note that in practice, Mask R-CNN can be replaced by any other trainable instance segmentation method. As mentioned in Section 4.1, the instance segmentation network may assign low confidence scores for some correct detections of unseen classes. Therefore, during the mask generation stage, we set the confidence score threshold to 0 to obtain as many detections as possible.

4.2.2 Mask Selection

Given a video of T frames, we start from a set of mask candidates $\mathcal{M}_t = \{M_{t,1}..M_{t,N}\}$ for each frame I_t obtained using our baseline network, with N the number of masks candidates in I_t. To select the mask candidates that actually correspond to objects, we exploit the following cues and constraint:

- The **"Background cue"**: Segmenting typical backgrounds such as sky or grass gives us a cue about where the objects are;
- The **"Flow cue"**: The optical flow between consecutive frames gives us a cue about the moving objects;
- The **"Consistency cue"**: The selected masks should be consistent not only between consecutive frames, but also over long sequences;
- The **"Non-overlapping constraint"**: An additional constraint that is usually overlooked is that the masks should not overlap: Ideally, one pixel in the image can belong to at most one mask.

As we will show in the following sections, each cue corresponds to a loss term in the final objective function. Each of them and the non-overlapping constraint contribute to removing the false positives, as will be demonstrated by our ablation study in Section 4.5.5.

To combine these cues to select the correct masks in a given video sequence, we rely on a Bayesian framework. This selection problem can be formalized as maximizing the probability of the detected masks given the frames of the video:

$$P\big(\mathbf{C}_1, .., \mathbf{C}_T | I_1, .., I_T\big) \ , \tag{4.1}$$

where $\mathbf{C}_t$ is a set of binary random variables, with $\mathbf{C}_{t,i} = 1$ corresponding to the event that the mask $M_{t,i}$ is selected and 0 that it is not. By applying Bayes' theorem, the problem becomes the maximization of:

$$P\big(I_1, .., I_T | \mathbf{C}_1, .., \mathbf{C}_T\big) P\big(\mathbf{C}_1, .., \mathbf{C}_T\big) \ . \tag{4.2}$$

To keep both the image and optical flow cues, we use a Product-of-Experts [99] with cliques made of two consecutive images to model the first term as:

$$P\big(I_1, .., I_T | \mathbf{C}_1, .., \mathbf{C}_T\big) \propto \prod_t P\big(I_t, I_{t+1} | \mathbf{C}_1, .., \mathbf{C}_T\big) \ . \tag{4.3}$$

We make the standard assumptions that the successive states (the $\mathbf{C}_t$) follow a Markov process, and that the measurements at time t (the I_t) depend only upon the current state ($\mathbf{C}_t$). We denote $\delta_{t,i}$ the realization of the random binary variable $\mathbf{C}_{t,i}$ (thus $\delta_{t,i} \in \{0, 1\}$) and $\Delta_t = \{\delta_{t,1}, .., \delta_{t,N}\}$ the realization of $\mathbf{C}_t$. After standard derivations, the optimization problem becomes:

$$arg\,max_{\{\Delta_1, .., \Delta_T\}} P(\mathbf{C}_0) \prod_t P\big(I_t, I_{t+1} | \mathbf{C}_t, \mathbf{C}_{t+1}\big) P(\mathbf{C}_{t+1} | \mathbf{C}_t) \ . \tag{4.4}$$

$P\big(I_t, I_{t+1} | \mathbf{C}_t, \mathbf{C}_{t+1}\big)$ is high when the image likelihoods $P\big(I_t | \mathbf{C}_t\big)$ and $P\big(I_{t+1} | \mathbf{C}_{t+1}\big)$ are high and when the object motions between I_t and I_{t+1} are consistent with states $\mathbf{C}_t$ and

$\mathbf{C}_{t+1}$. $P\big(I_t, I_{t+1}|\mathbf{C}_t, \mathbf{C}_{t+1}\big)$ is thus directly proportional to:

$$P\big(I_t|\mathbf{C}_t\big)P\big(I_{t+1}|\mathbf{C}_{t+1}\big)P\big(F_t|I_t, I_{t+1}, \mathbf{C}_t, \mathbf{C}_{t+1}\big)\ , \tag{4.5}$$

where F_t is the optical flow for the pair of frames (I_t, I_{t+1}). After taking the $\log$ of Eq. (4.4), the optimization problem can be written as:

$$\begin{aligned} arg\,min_{\{\Delta_1,..,\Delta_T\}} \quad & \textstyle\sum_t \big(\lambda_I \mathcal{L}_I\big(I_t, \Delta_t\big) + \\ & \lambda_F \mathcal{L}_F\big(F_t, I_t, I_{t+1}, \Delta_t, \Delta_{t+1}\big) + \\ & \lambda_p \mathcal{L}_p(\Delta_t, \Delta_{t+1})\big)\ , \end{aligned} \tag{4.6}$$

under the non-overlapping constraint that will be detailed below. λ_I, λ_F, and λ_p are weights. We take $\lambda_I = \lambda_F = 1$ and $\lambda_p = 0.5$ in all our experiments. F_t represents the optical flow for the pair of frames (I_t, I_{t+1}). $\Delta_t = \{\delta_{t,1}, .., \delta_{t,N}\}$ denotes the realization of $\mathbf{C}_t$ where $\delta_{t,i}$ is the realization of the random binary variable $\mathbf{C}_{t,i}$. $\delta_{t,i} = 1$ when $M_{t,i}$ is selected, otherwise $\delta_{t,i} = 0$.

We call $\mathcal{L}_I$ the Background loss and $\mathcal{L}_F$ the Flow loss, as they exploit the Background cue and the Flow cue respectively. $\mathcal{L}_p$ enforces consistent selections between consecutive frames. We detail these three losses below.

4.2.3 Background Loss $\mathcal{L}_I$

We use $\mathcal{L}_I$ to exploit the Background cue that hints at where the objects are. As shown in Figure 4.3, to evaluate it, we compare a binary image generated for the selected masks and the foreground/background probability map predicted by a binary segmentation network f by calculating their cross entropy, as the image background should match the background of the selected masks. By doing this comparison over all the image locations, we can exploit information from the whole image to guide the mask selection—we will rely on the same strategy for the other terms. Formally, we take

$$\mathcal{L}_I\big(I_t, \Delta_t\big) = \mathrm{CE}\big(\mathrm{Bg}(I_t), 1 - \overline{\mathrm{Fg}}(\Delta_t)\big)\ , \tag{4.7}$$

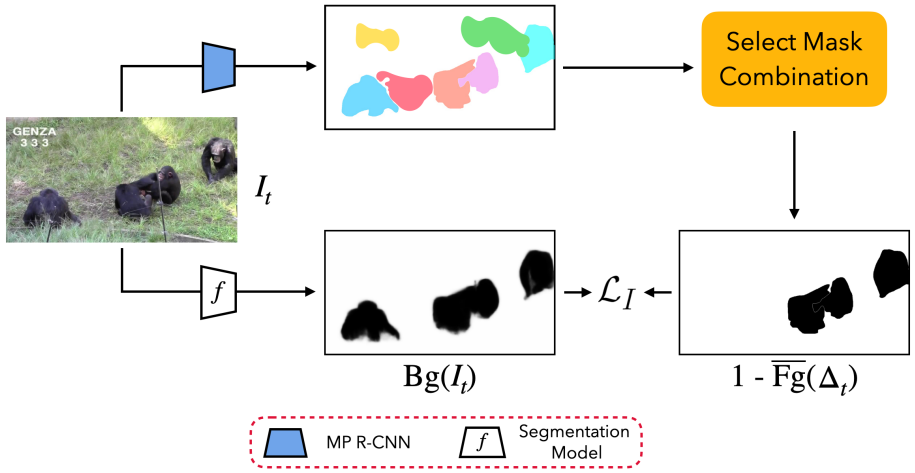


Figure 4.3: To evaluate the background loss $\mathcal{L}_I$ of Eq. (4.7), we compare the background predicted for the image and the background of the selected masks.

where CE denotes the cross-entropy, $\text{Bg}(I_t)$ is a probability map for each pixel to belong to the background as predicted by the network f, and $\overline{\text{Fg}}(\Delta_t)$ is the binary image of masks in $\mathcal{M}_t$ such that $\delta_{t,i} = 1$. For f, we use the network architecture proposed in [128] trained on the same training data as our baseline network for mask generation.

4.2.4 Flow Loss $\mathcal{L}_F$

We use $\mathcal{L}_F$ to exploit the Flow cue: The optical flow is the result of the object motions and the camera motion, and thus also hints at where the objects are. Even if an object is static but the camera is in motion, when the distance between the camera and the background is large enough, relative motion will make the optical flow of the object regions stand out from the background optical flow.

Figure 4.4 shows how we evaluate this term. We compare the flow predicted by an optical flow estimator g and a "synthetic optical flow" generated using the masks selected in $\mathcal{M}_t$ and $\mathcal{M}_{t+1}$. Similar to the $\mathcal{L}_I$ term, this comparison allows us to exploit information from all the image locations. In practice, we use the method of [243] for g. To generate the synthetic optical flow, we use the colors of the pixels in the selected masks to compute their optical flow. We average the flow in F_t on the pixels that do not belong to any mask

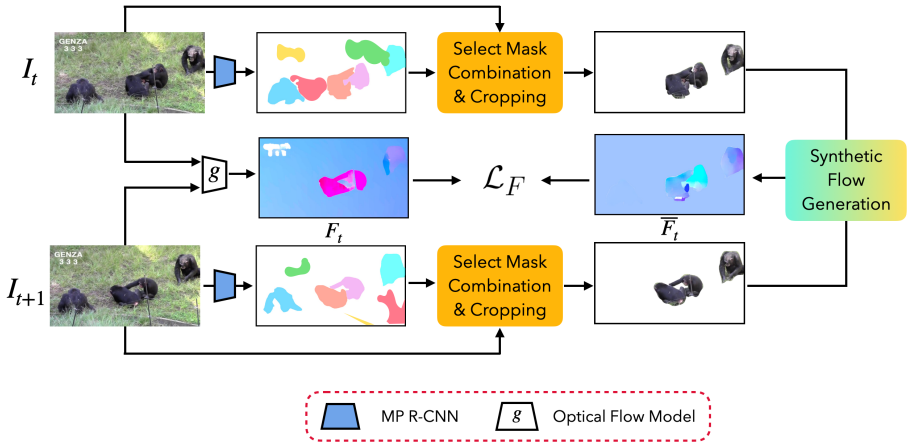


Figure 4.4: To evaluate the flow loss $\mathcal{L}_F$ of Eq. (4.8), we compare the optical flow estimated between two consecutive images and the optical flow computed for the masks selected in the two images.

to assign these pixels the resulting value.

Using this procedure, the measured flow $F_t = g(I_t, I_{t+1})$ and the synthetic flow are similar when all moving objects are correctly selected in both frames, even when the camera is in motion. More formally, we take:

$$\mathcal{L}_F\big(F_t, I_t, I_{t+1}, \Delta_t, \Delta_{t+1}\big) = \big\|F_t - \overline{F}_t\big\|_1 , \tag{4.8}$$

where $\overline{F}_t = \overline{F}_t(I_t, I_{t+1}, \Delta_t, \Delta_{t+1})$ is the synthetic flow generated for the selected masks in $\mathcal{M}_t$ and $\mathcal{M}_{t+1}$. We use the L1-norm to compare the two flows to be robust to outlier values that are very common in the predicted flow. Figure 4.4 shows examples for F_t and $\overline{F}_t$.

4.2.5 Regularization Loss $\mathcal{L}_p$ and Constraint

As discussed above, the optimization in Eq. (4.6) should be done under the constraint that no masks selected for the same frame overlap each other.

$\mathcal{L}_p\big(\Delta_t, \Delta_{t+1}\big)$ is usually interpreted in tracking problems as a motion model. We use it to enforce a consistent selection of masks between consecutive frames. Figure 4.5 shows

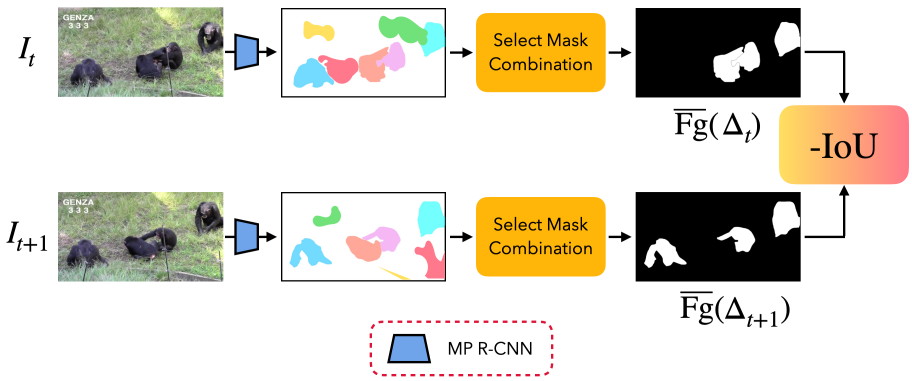


Figure 4.5: To evaluate the regularization loss $\mathcal{L}_p$ of Eq. (4.9), we compare the binary images of the selected masks in two consecutive images.

how we compute it: We use a very simple motion model and assume that the objects move slowly, in other words, the areas segmented as objects do not change abruptly between two consecutive frames. Formally, we take :

$$\mathcal{L}_p\big(\Delta_t, \Delta_{t+1}\big) = -\text{IoU}\big(\overline{\text{Fg}}(\Delta_t), \overline{\text{Fg}}(\Delta_{t+1})\big)\ , \tag{4.9}$$

i.e. the negative Intersection-over-Union between the binary images of the masks selected for frames I_t and I_{t+1}. It is set to 0 when no masks are selected for none of the two images.

4.3 Two-Stage Optimization

In this section, we introduce an efficient way to minimize Eq. (4.6). Note that minimizing this function requires to optimize on all the frames simultaneously. A naive approach is to apply exhaustive search for the solution of the problem, where the number of evaluations of the objective function would be $O(2^{NT})$, with N the number of mask candidates per frame and T the number of frames (typical values $N = 15$ and $T = 180$ would require $\sim 10^{810}$ evaluations). This is clearly computationally prohibitive. Recall that, in the first stage, we select the most promising combinations of masks for each frame independently using the background loss $\mathcal{L}_I$ and under the constraint that the selected masks should not

overlap, we formulate this problem as a K-shortest path search problem and detail our method below in Section 4.3.1. Then, we detail in Section 4.3.2 the second stage where we optimise the full objective function on the whole video while considering only the top combinations selected during the first stage.

We provide here an efficient two-stage algorithm, depicted in Figure 4.9. In the first stage, based on the background loss $\mathcal{L}_I$ and the non-overlapping constraint, we select the top-K most promising combinations of masks for each frame independently. Note that the combinations violating the non-overlapping constraint are simply discarded. Then, in the second stage, we optimize the complete objective function over all frames simultaneously to find the best combinations for each frame. For both stages, we can use Dijkstra's algorithm [50] to significantly decrease the complexity of the computations. In the worst case, the number of evaluations of the objective function becomes $O(KTN^3 + K^2T^2)$. We use $K = 10$ in practice, which reduces the required evaluations from $\sim 10^{810}$ to $\sim 10^7$ for the numerical example above.

This optimization problem is related to many previous works on multiple object tracking [14, 171, 209, 274, 291, 292], which typically use graph-based methods to solve related problems efficiently. One of the main differences with these works is that, in our video-level optimization, each node corresponds to a combination of masks instead of a single bounding box or mask. Besides, we do not have access to the object classes, and our optimization is under the constraint that the masks do not overlap, while these works typically rely on bounding boxes that can overlap when the objects are close to each other.

4.3.1 Image-Level Optimization

At this stage, for each frame I_t, we look for the top-K combinations of masks in the power set $\mathcal{P}(\mathcal{M}_t)$ that minimize the Eq.(4.7) under the non-overlapping constraint. An exhaustive search would take 2^N evaluations of the objective function.

However, we note that this problem can be formulated as a K-shortest path search problem in a binary tree, where each pair of branches of a node corresponds to the selection or not of a mask, and each branch has an associated weight: This weight is set to

infinity if the branch corresponds to the selection of a mask that overlaps with one of its ancestors, otherwise it depends on the value of $\mathcal{L}_I$ computed only on the mask. By iteratively applying Dijkstra's algorithm to find the top-K combinations of masks, we reduce the number of evaluations to $O(KN^3)$. Note that our proposed algorithm is agnostic to the order of the mask candidates in $\mathcal{M}_t$.

In details, given a frame I_t, we obtain a set of mask candidates $\mathcal{M}_t$ using MP R-CNN. To identify the top-K combinations of masks according to $\mathcal{L}_I\big(I_t, \Delta_t\big)$ without having to perform an exhaustive evaluation, we iteratively apply Dijkstra's algorithm [50]. To do this, we first construct a binary tree $\mathcal{B}(\mathcal{V}, \mathcal{E})$. As shown in Figure 4.6 for the easy case where the masks in $\mathcal{M}_t$ do not overlap, nodes $\mathcal{V}_i = \{\mathcal{V}_{i,j}\}_j$ at the same level in the tree correspond to the state of the i-th mask $M_{i,t} \in \mathcal{M}_t$ *i.e.* if it is selected or not, and edges $\mathcal{E}_i = \{\mathcal{E}_{i,j}\}_j$ at the same level are weighted with the contribution to $\mathcal{L}_I\big(I_t, \Delta_t\big)$ when mask $M_{i,t}$ is selected or not. j is the index of the submasks in the overlapping case, as shown in Figure 4.7. In this way, the problem of finding the most promising combinations of masks for a frame thus becomes to find the K-shortest paths in the binary tree $\mathcal{B}(\mathcal{V}, \mathcal{E})$. We iteratively apply Dijkstra's algorithm [50] to efficiently find the top-K shortest paths. We get the shortest path $\mathcal{H}_1$ by directly applying Dijkstra's algorithm on the binary tree $\mathcal{B}(\mathcal{V}, \mathcal{E})$. To find the next shortest path, we consider a set U of paths, initialized with $\mathcal{H}_1$ as its only element. The whole procedure consists of two nested loops. The first loop is over each path $\mathcal{H}$ in U. Given $\mathcal{H}$, the second loop is over each edge $\mathcal{E}_{i,j}$ of $\mathcal{H}$. We set the weight of $\mathcal{E}_{i,j}$ to infinity and apply Dijkstra's algorithm on the resulting tree. Each time we run Dijkstra's algorithm, we obtain an "intermediate shortest path" different from $\mathcal{H}$. After exiting the two loops, we add all the intermediate shortest paths to U and remove $\mathcal{H}_1$ from it. The shortest path in U is now the second shortest path $\mathcal{H}_2$ in the tree. We can then repeat the above procedure to get the next shortest paths. This method decreases the time complexity from 2^N to $O(KN^3)$, where N is the number of masks per image and K is the number of optimal combinations required.

Below, we start with the easier case, where there is no overlap among the mask candidates in $\mathcal{M}_t$. In this case, each pixel in the image belongs to at most one mask candidate.

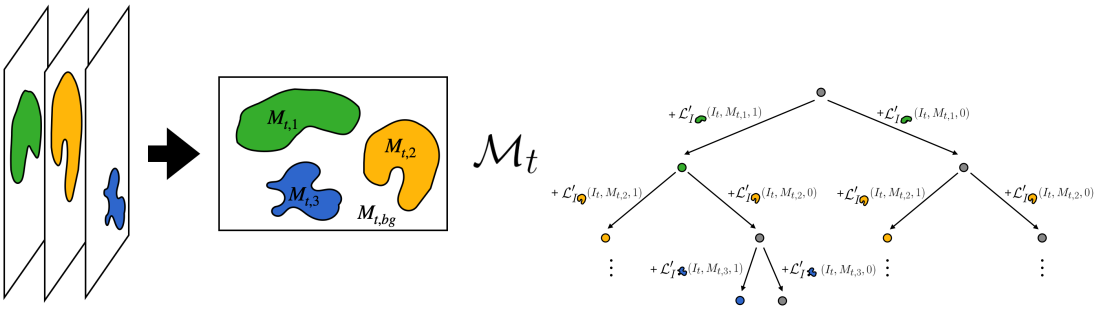


Figure 4.6: Image-Level Optimization. Example of a tree $\mathcal{B}(\mathcal{V}, \mathcal{E})$ in the non-overlapping case. Nodes at the i-th level correspond to the selection ($\delta_{t,i} = 1$) or non-selection ($\delta_{t,i} = 0$) of the i-th mask. Edges at the i-th level are weighted according to the image term $\mathcal{L}_I$ and depending whether or not the i-th mask is selected. As there is no overlaps among the three masks, the weights of edges can be calculated independently. A colored node corresponds to the selection of the mask with the same color; a gray node corresponds to the case where the mask is not selected.

The non-overlapping constraint is naturally satisfied, and the contributions of the masks to $\mathcal{L}_I\big(I_t, \Delta_t\big)$ are independent of each other. Then, we dive into the overlapping case and show that we can transform the overlapping case to the non-overlapping case with a decomposing-then-hashing operation, and calculate $\mathcal{L}_I\big(I_t, \Delta_t\big)$ exactly.

Non-overlapping case. We first consider the easier case where there is no overlap between the mask candidates, illustrated by Figure 4.6. In such case, image term $\mathcal{L}_I\big(I_t, \Delta_t\big)$ can be written as a sum over the masks in $\mathcal{M}_t$ plus a special additional mask made of the pixels that do not belong to any mask (recall that $\Delta_t = \{\delta_{t,1}, .., \delta_{t,N}\}$):

$$\begin{aligned}\mathcal{L}_I\big(I_t, \Delta_t\big) &= \mathrm{CE}\big(\mathrm{Bg}(I_t), 1 - \overline{\mathrm{Fg}}(\Delta_t)\big) \\ &= \sum_{M_{t,i} \in \mathcal{M}_t} \mathcal{L}'_I(I_t, M_{t,i}, \delta_{t,i}) + \bar{\mathcal{L}}_I(M_{t,bg})\end{aligned} \tag{4.10}$$

with

$$\begin{aligned}\mathcal{L}'_I(I_t, M_{t,i}, \delta_{t,i}) = &- \sum_{x \in M_{t,i}} (1 - \delta_{t,i}) \log \mathrm{Bg}(I_t)(x) \\ &+ \delta_{t,i} \log\big(1 - \mathrm{Bg}(I_t)(x)\big)\,,\end{aligned} \tag{4.11}$$

where the sum is over image locations x in mask $M_{t,i}$, and

$$\bar{\mathcal{L}}_I(M_{t,bg}) = -\sum_{x \in M_{t,bg}} \log \mathrm{Bg}(I_t)(x) \,. \tag{4.12}$$

Term $\bar{\mathcal{L}}_I(M_{t,bg})$ in Eq. (4.10) is constant. $M_{t,bg}$ represents the background image generated for the selected masks. We can ignore it here for the selection of the top-K mask combinations. Note that it still needs to be included in $\mathcal{L}_I$ for the Video-Level Optimization. We can use the value of $\mathcal{L}'_I(I_t, M_{t,i}, 1)$ as weight for an edge corresponding to the selection of mask $M_{i,t}$, $\mathcal{L}'_I(I_t, M_{t,i}, 0)$ as weight for an edge when mask $M_{i,t}$ is not selected.

Overlapping case. In general, the mask candidates predicted by MP R-CNN overlap, and the contributions of masks $\{M_{i,t}\}_i$ to $\mathcal{L}_I\big(I_t, \Delta_t\big)$ are no longer independent. We propose a decomposing-then-hashing method which first decomposes each mask into a group of non-overlapping sub-masks, then identify the contribution of each mask using the sum of the sub-masks. When adding a novel node to the path, we fit the non-overlapping constraint by inspecting if there are some sub-masks selected in both the current node and the nodes in this path.

As shown in Figure 4.7, we first decompose each mask $M_{t,i}$ into a combination of sub-masks $\{M'_{t,i,j}\}_j$ depending on whether the pixels in the mask belong to other masks or not. The whole image can thus be decomposed into the combination of all sub-masks $\{M'_{t,i,j}\}_{i,j}$ and $M_{t,bg}$, with no overlaps among the sub-masks in $\{M'_{t,i,j}\}_{i,j}$. Then, we can construct a hash table where the key is the index of each mask $M_{t,i}$ and the value is the index of corresponding sub-masks $\{M'_{t,i,j}\}_j$ for each mask $M_{t,i}$.

Similar to the non-overlapping case, with the help of non-overlapping sub-masks, we have:

$$\begin{aligned}\mathcal{L}'_I(I_t, M_{t,i}, \delta_{t,i}) = -\sum_{S \in \{M'_{t,i,j}\}_{i,j}} \sum_{x \in S} \Big(&(1-\delta_{t,i}) \log \mathrm{Bg}(I_t)(x) \\ &+ \delta_{t,i} \log\big(1 - \mathrm{Bg}(I_t)(x)\big)\Big)\end{aligned} \tag{4.13}$$

which means that the contribution of each mask to $\mathcal{L}_I\big(I_t, \Delta_t\big)$ can be decomposed

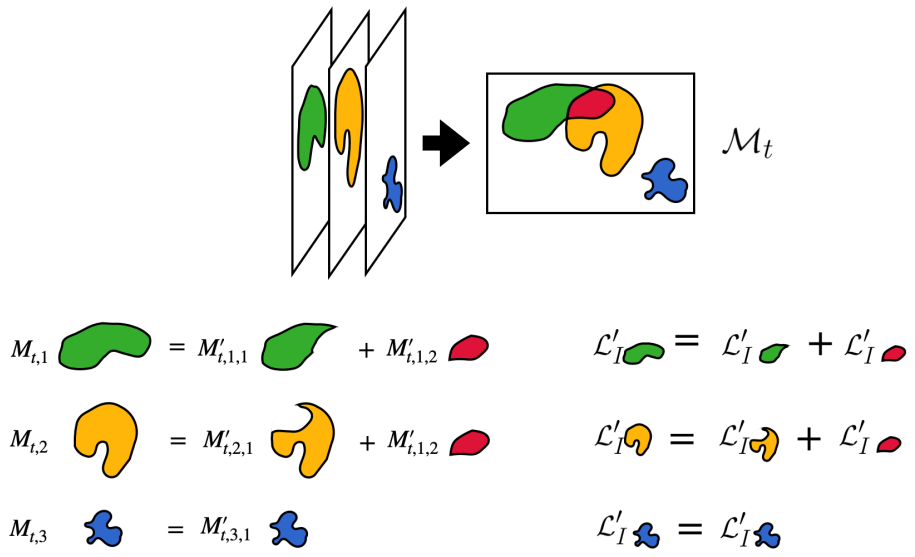


Figure 4.7: Image-Level Optimization. Decomposing-then-hashing for the overlapping case. The red mask represents the overlapping region between $M_{t,1}$ and $M_{t,2}$. The contribution of $M_{1,t}$ can thus be decomposed into the sum of contributions of the two non-overlapping green and red sub-masks.

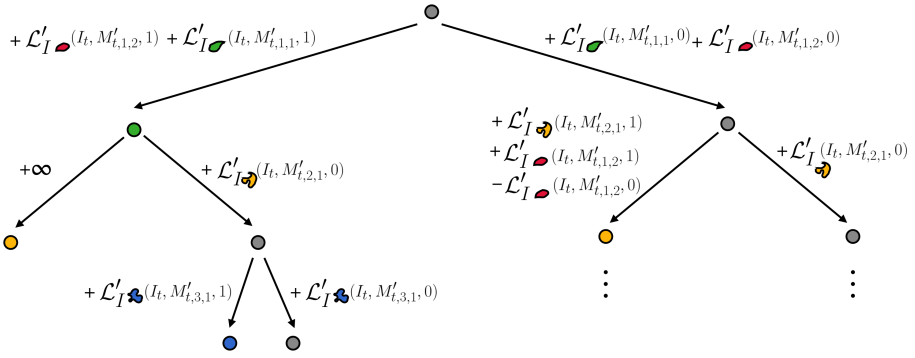


Figure 4.8: Image-Level Optimization. Example of $\mathcal{B}(\mathcal{V}, \mathcal{E})$ in the overlapping case. Each time a node is added in the path, the weight of the edge is calculated by considering all the previous nodes in the path.

into the sum of the contributions of its corresponding sub-masks. Eq. (4.10) can thus be written as:

$$\mathcal{L}_I\big(I_t, \Delta_t\big) = \sum_{S \in \{M'_{t,i,j}\}_{i,j}} \mathcal{L}'_I(I_t, S) + \bar{\mathcal{L}}_I(M_{t,bg}) \,. \tag{4.14}$$

A tree for an overlapping case is shown in Figure 4.8. When searching for the shortest path in the graph, given a path, each time a node is added to the path, with the help of the hash table constructed aforementioned, we first check whether there are any sub-masks $M'_{t,i,j}$ already being assigned a certain "state" in the previous nodes (by state, we mean if it has been selected as foreground or selected as background). If not, like the non-overlapping case, we consider the $\mathcal{L}'_I(I_t, M_{t,i}, \delta_{t,i})$ for the weight of the edge $B, \forall i > 0$. If a sub-mask has already been assigned some state, there are two possible cases. The first case is that the sub-masks $M'_{t,i,j}$ selected both in the current node and a previous node have the same state. When the state is "background", as we have already counted the contribution of sub-masks in previous edges, the weight of the current edge is assigned to the sum of the contribution of the rest of masks. When the state is "foreground", since the masks should not overlap, the weight of the current edge is set to infinity to prohibit this path from becoming one of the top-K shortest paths. The second case is that some sub-masks $M'_{t,i,j}$ selected both in the current node and a previous node possess different states. In order to calculate the weight of the current edge, along with the contribution of each sub-mask by considering the state assigned by the current node, we subtract the contribution of the shared sub-masks according to their assigned state in the previous node.

4.3.2 VIDEO-LEVEL OPTIMIZATION

As shown in Figure 4.9, we generate a graph with the remaining top-K combinations of masks for each frame: Each node corresponds to one combination and each edge is labeled with the loss given in Eq.(4.6) for the two combinations it links. Finding the best combination for each frame becomes the problem of finding the shortest path in this graph. Instead of doing 2^{KT} evaluations of the objective function to find the shortest path,

we can use Dijkstra's algorithm [50] here as well to significantly accelerate the speed of our algorithm. The number of evaluations is reduced to $O(K^2T^2)$.

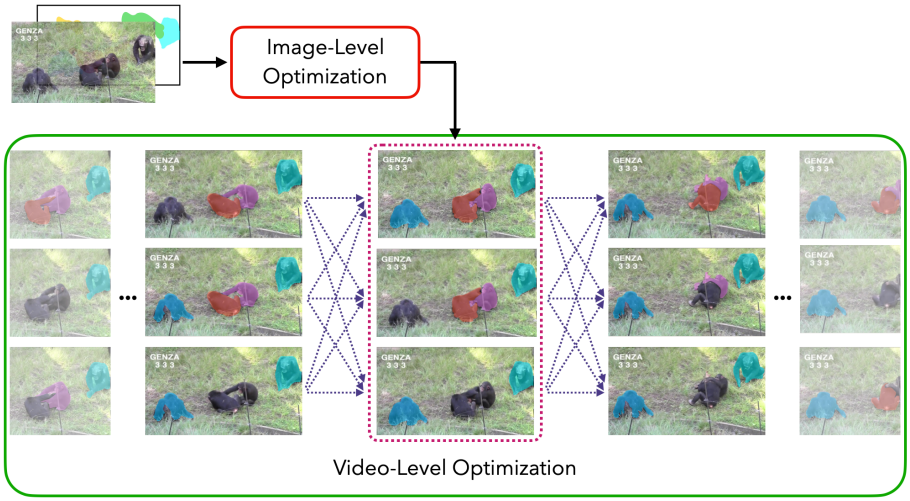


Figure 4.9: Given a video, we first run the image-level optimisation on each frame and get the top-K combinations of masks for each frame. The video-level optimisation selects the best combination for each frame efficiently by solving a shortest path problem. For this figure, K is set to 3. More details are given in Section 4.3.

In details, as shown in Figure 4.10, after the top-K combinations of masks for each frame have been found by the first stage, we construct a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ to find the best combinations of masks for each frame by optimizing the objective function over the whole video. Node $\mathcal{V}_{t,k}$ represents the k-th combination of masks of frame I_t. $\mathcal{E}_{t,t+1,k,k'}$ represents the edge that connects node $\mathcal{V}_{t,k}$ and node $\mathcal{V}_{t+1,k'}$. The weight of edge is a sum over the image term, the flow term and the motion model term:

$$\begin{aligned} \text{Weight}_{\mathcal{E}_{t,t+1,k,k'}} =& \lambda_I \mathcal{L}_I\big(I_t, \Delta_t\big) \\ &+ \lambda_F \mathcal{L}_F\big(F_t, I_t, I_{t+1}, \Delta_t, \Delta_{t+1}\big) \\ &+ \lambda_p \mathcal{L}_p(\Delta_t, \Delta_{t+1})\,. \end{aligned} \tag{4.15}$$

In our case, as the weight of every edge is non-negative, we can use Dijkstra's algorithm [50] to select the combination of masks among the top-K combinations for each frame.

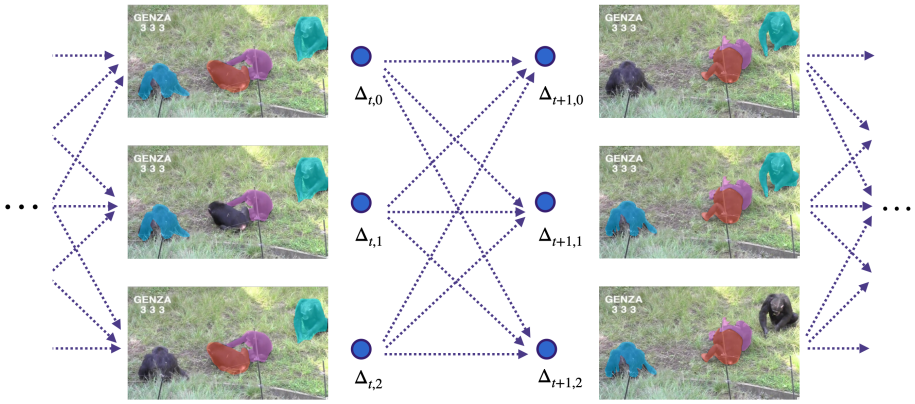


Figure 4.10: Video-Level Optimization. Example of graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, represented only between two consecutive frames I_t and I_{t+1} for a video sequence of chimpanzees. Each image in the column represents a combination of masks Δ_t, different masks are shown in different colors (Best seen in color).

4.4 Implementation Details

In this section, we first present the implementation details of all the methods including ours, then we present our benchmark for evaluating our approach. We compare our method with several previous methods for the purpose of generating masks on unlabeled videos and analyze the results. We also conduct a thorough ablation study to show the influence of different components of our method. All our experiments were carried out on 4 Nvidia RTX 2080Ti GPUs. During the training, mixed precision training is used to reduce memory consumption and accelerate training.

4.4.1 Network Pre-training

For the pre-training on the COCO dataset, we use the open source repo of Mask-RCNN [184] and follow the training setting of [93], except that we adopt the class-agnostic setting, where all 80 classes are merged into a single "object" category. We use ResNet-50-FPN [154] as our backbone. As in [201], we call the Mask R-CNN model trained using this class-agnostic setting "MP R-CNN" for Mask Proposal R-CNN. Our backbone network is initialized with weights pre-trained on ImageNet [46]. During training, the shorter edge of images are resized to 800 pixels. Each GPU has 4 images and each image has

512 sampled RoIs, with a ratio of 1:3 of positives to negatives. We train our MP R-CNN for 90k iterations. The learning rate is set to 0.02 at the beginning and is decreased by 10 at the 60k and 80k iteration. We use a weight decay of 0.0001 and momentum of 0.9.

4.4.2 Network Fine-tuning

We finetune MP R-CNN on the masks of Unseen-VIS-train generated by our method for 1k iterations. The initial learning rate is set to 0.002. The rest of the parameters and data augmentation strategies are set to be the same with the parameters used for pre-training. For the *DAVIS Unsupervised* benchmark, we finetune MP R-CNN on the masks generated on the 60 training videos of DAVIS2017 dataset, using the same hyper-parameters as for Unseen-VIS-train.

4.4.3 Network Re-training

For retraining, we use a naive strategy to train MP R-CNN from scratch on the mixture of COCO dataset and labels from Unseen-VIS-train. We add the images and annotations of Unseen-VIS-train into the COCO dataset, the annotations can be either ground truth or masks generated with our approach. Then we follow the training strategy given in Section 4.4.1. MP R-CNN is trained on 120k training images of COCO for 90k iterations with batch size 16. The mixture of COCO and Unseen-VIS-train has 20k additional training images from Unseen-VIS-train, we therefore train the our MP R-CNN on the mixture dataset for 110k iterations.

4.4.4 Foreground/Background Segmentation

In order to estimate the background regions present in the image, similar to [128], we adopt a FPN based network for segmentation. We use ResNest-200 [293] as our backbone. The regions that contain instances are considered as foreground and background otherwise to generate binary mask for training. We set the initial learning rate to 0.02,

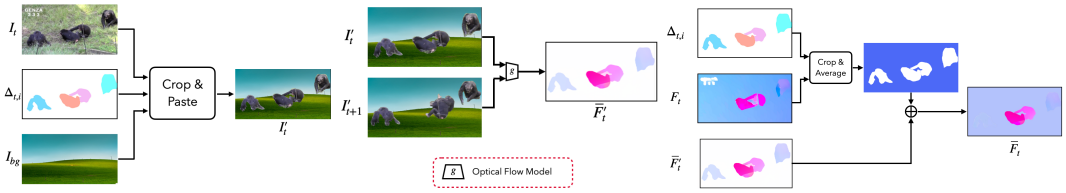


Figure 4.11: Generation pipeline for "synthetic optical flow" $\overline{F}_t$. **(a)** We first generate two synthetic images I'_t and I'_{t+1} by cropping and pasting the content of selected masks in frame I_t and I_{t+1} to a background image I_{bg} randomly selected from the Internet. **(b)** We then feed the pair of synthetic images to a optical flow model g to generate a synthetic $\overline{F}'_t$. **(c)** The pixels outside the selected masks are assigned the average flow in F_t computed over the background.

then decrease it by 10 at the 240k and 255k iteration. The batch size is set to 16. Scale augmentation is applied during training to further improve the performance.

4.4.5 Optical Flow Estimation

We use the model released by [243] trained on FlyingThings [185]. FlyingThings is a large-scale synthetic dataset for optical flow estimation. The dataset is generated by randomizing the movement of the camera and synthetic objects collected from the ShapeNet dataset [29]. The model for optical flow estimation is pre-trained on FlyingThings for 100k iterations with a batch size of 12, then for 100k iterations on FlyingThings3D with a batch size of 6.

4.4.6 Generation of Synthetic Flow $\overline{F}_t$

The motion of the selected objects should be consistent with the optical flow. To deal with camera motion, we consider that the optical flow of the background can be different from 0 (but uniform). Given two consecutive frames I_t and I_{t+1}, together with two combinations of masks Δ_t and Δ_{t+1} for these frames, we propose a method to generate a "synthetic optical flow" which is then compared against the optical flow predicted by some optical flow model.

An overview of the generation pipeline is shown in Figure 4.11. We first generate two synthetic images I'_t and I'_{t+1} by cropping and pasting the content of selected masks in frame I_t and I_{t+1} to a background image I_{bg} randomly selected from the Internet. Then we pass the pair of synthetic images into a optical flow model g to generate a synthetic $\overline{F}'_t$. The pixels outside the selected masks are assigned the average flow in F_t computed over the background.

The motivation for using a background image rather than a uniform background color which seems more natural is to make sure the objects are visible before computing the flow. For example, in the case of Figure 4.10, using a uniform black background would make optical flow prediction fail for the dark chimpanzees. Using a real image is a simple way to prevent this problem.

4.4.7 Implementation Details of Other Methods

NLC [63]. We use the re-implementation code provided by [207]. In the original implementation of the NLC algorithm, an edge detector pre-trained on labeled edge images [54] is used, while the re-implementation of [207] replaces the trained edge detector with unsupervised superpixels. In order to obtain a motion segmentation for each frame, a per-frame saliency map based on motion is computed then averaged over superpixels calculated using method from [3]. Meanwhile, a nearest neighbor graph is computed over the superpixels in the video using location and appearance as features. Finally, the saliency is propagated across frames using a nearest neighbor voting scheme. For more details, please refer to [63, 207].

FST [204]. We use the official code released by the authors to generate masks on the videos from Unseen-VIS-train.

IOA [42]. We use the official code released by the authors. We first use the official code of the VideoPCA algorithm [238] to find the saliency regions which stand out the background in terms of motion and appearance. Then we follow the same approach as

in [42] to keep only the top 20% saliency maps based on the mean score of the non-zero pixels. The remaining saliency maps are used for training of the PSPNet [**?**] with ResNet-50 [95] as backbone, and initialized with ImageNet pre-trained weights [46].

UnOVOST [169]. We use the official code released by the authors, and use our MP R-CNN for mask proposals generation. All the hyper-parameters are kept the same as in the original work [169].

TWB [15]. We use the official code released by the authors, and replace the human detector with our MP R-CNN. Since the confidence score for objects of unseen classes are sometimes low, we lower the thresholds for filtering detection proposals and regression results to 0.1. Note that [15] applies a Re-ID network to re-identify the persons that are missing in previous frames, in order to re-identify the animals in Unseen-VIS-train, we replace the Re-ID network trained on the pedestrian dataset with a ResNet-50 pretrained on ImageNet [46].

Data distillation [218]. We use the official code released in Detectron2 [273] and follow the same pipeline as stated in [218] for filtering and retraining.

UVC [151]. We use the model released by the authors for mask wrapping. The model is trained in a self-supervised manner on the Kinetics [125] dataset. During inference, we follow the same setting of UVC to resize the shorter edge of image to 480 pixels. For Zero-Shot UVC, the masks for the first frame are obtained by selecting the masks predicted by MP-RCNN on the first frame whose confidence score is larger than 0.1.

RVOS [255]. We use the official code released by the authors, and replace the backbone of RVOS by ResNet-50. Input images are resized to 256×448. Each batch is composed with 4 clips of 5 consecutive frames. The model is trained on the 1089 videos of seen classes on YouTube VIS [284] for 50 epochs. The Adam optimizer is used to train our network and the initial learning rate is set to $1e{-}6$.

4.5 EXPERIMENTS

4.5.1 EVALUATION

To benchmark our method, we created a dataset we call "Unseen-VIS". The training part of "Unseen-VIS" consists of videos collected from the YouTube Video Instance Segmentation (YouTube VIS) [284] and is used for mask generation. The test part of "Unseen-VIS" contains static images extracted from YouTube VIS for evaluation.

The original YouTube VIS dataset contains 2,883 videos with 131k object instances spanning 40 classes, among which 24 coincide with COCO. Thus, we consider the remaining 16 classes, containing panda, lizard, seal, shark, mouse, frog, tiger, leopard, fox, deer, ape, snake, monkey, rabbit, fish, turtle, as *unseen* classes which results in 795 videos in total. We randomly selected 595 videos as the training set, which we refer as Unseen-VIS-train. The labeled static images in the remaining 200 videos are used for evaluation, which we refer as Unseen-VIS-test. All the videos of Unseen-VIS-train are used as unlabeled videos, and their ground truth masks are ignored.

For quantitative evaluation, we rely on the standard COCO metrics: AP, AP_{50}, AP_{75}, and AR_1, AR_3 and AR_5 as the maximum number of objects per image in our testing set is 4. We do not use AP_S, AP_M, and AP_L as the object scales in COCO differ largely from YouTube VIS.

4.5.2 VIDEO-ANNOTATION-FREE MASK GENERATION

We first compare our method to other approaches that can also generate masks given video sequences without using any video annotations. Each method is first applied on the Unseen-VIS-train dataset for mask generation, then we compare the performance of MP R-CNN on the Unseen-VIS-test set after fine-tuning on these masks. As only one over five frames is annotated in Unseen-VIS-train (19352 annotated frames in total), we therefore use only the masks of these frames for training for fair comparison among different methods.

Saliency/Flow-based methods. FST [204] and NLC [63] can generate masks from videos by estimating the saliency and motion of the objects in videos. IOA [42] trains a deep neural network on the output of an unsupervised soft foreground segmentation algorithm [238] to segment objects in videos. These methods can identify moving regions in the videos but can not separate adjacent objects in the images, thus a proposal generated by these methods may actually correspond to several objects.

Tracking-based methods. TWB [15] and UnOVOST [169] rely on a frame-by-frame tracking pipeline applied to mask proposals. These methods are the closest methods to ours as we all rely on an instance segmentation model for proposal generation. However, as we mentioned before, for our final goal (training a better object detector), we do not need to keep track of the detected objects.

Similarity propagation methods. Given masks for a frame, UVC [151] warps these masks to consecutive frames based on the estimated correspondences between consecutive frames. For this experiment, we use it in a zero-shot setting ("ZS-UVC"), where the masks of the first frame are instead generated by thresholding the confidence score on the first frame prediction of MP R-CNN (we use a threshold of 0.1 in practice).

Self-training methods. We also compare our method with the self-learning Data Distillation (DD) method [218]. We follow their proposed test time augmentation to generate masks on each frame of the Unseen-VIS-train videos independently, as this method performs on single images.

We report the results in Table 4.1. Compared with the Saliency/Flow-based methods and Tracking-based methods, the masks generated by our method are of high quality and can largely improve the performance of the baseline network in all metrics. Our method also outperforms the state-of-the-art self-training method (DD) [218] for two-stage object detection by a large margin.

Method used for mask generation	Unseen-VIS-test					
	AP	AP_{50}	AP_{75}	AR_1	AR_3	AR_5
(bef. fine-tuning)	35.8	61.2	38.1	33.3	47.3	50.3
NLC [63]	1.2	3.8	1.0	2.4	5.3	6.9
IOA [42]	2.4	8.5	0.9	6.9	8.7	9.5
FST [204]	17.0	41.8	11.3	22.0	30.6	33.1
UnOVOST [169]	31.1	55.6	32.2	29.9	44.5	48.2
TWB [15]	31.2	53.4	32.8	31.5	46.7	50.0
DD [218]	36.6	63.8	38.5	32.5	46.2	49.3
ZS-UVC [151]	21.2	42.6	19.9	26.3	40.0	43.2
Ours	**39.0**	**67.9**	**41.3**	**35.2**	**48.9**	**51.4**

Table 4.1: Mask Generation without Video Annotation. Performance of MP R-CNN on Unseen-VIS-test after fine-tuning on masks generated by various methods applied to Unseen-VIS-train without using any video annotation.

4.5.3 Video-Annotation-Dependent Mask Generation

In addition to the aforementioned methods, we further compare with two methods that adopt different settings and explore the upper bound of our method.

RVOS. RVOS [255] is an end-to-end video object segmentation framework that directly runs on videos, which requires labeled videos for training. We adopt the Zero-Shot setting for RVOS, where it is trained with ResNet50 [95] as backbone on 1089 videos (25869 annotated frames in total) of seen classes of YouTube VIS and directly applied to the Unseen-VIS-train videos for mask generation.

OS-UVC. Here, we consider the One-Shot setting for UVC, where the ground truth masks of the first frame are given for all Unseen-VIS-train videos.

Selected using ground truth. We use the ground truth mask labels of Unseen-VIS-train to select the masks predicted by MP R-CNN. The similarity among masks is evaluated based on their Intersection-over-Union, and the Hungarian algorithm is used to select the masks that best match with the ground truth. This can be regarded as an upper bound that

Method used for mask generation	Video Annotations Seen	Unseen	Unseen-VIS-test AP	AP_{50}	AP_{75}	AR_1	AR_3	AR_5
(bef. fine-tuning)			35.8	61.2	38.1	33.3	47.3	50.3
RVOS [255]	✓		38.5	68.9	38.0	35.4	49.5	52.8
Ours	-	-	39.0	67.9	41.3	35.2	48.9	51.4
OS-UVC [151]		✓	41.5	73.7	42.9	39.1	52.7	54.9
Selected using GT		✓	42.7	75.1	45.3	37.3	53.4	53.6
Trained with GT		✓	50.8	80.9	54.6	43.6	58.6	60.6

Table 4.2: Mask Generation with Video Annotations. Performance of MP R-CNN on Unseen-VIS-test after fine-tuning on the masks generated by various methods that require manual video annotations except ours. RVOS uses labeled videos of Seen classes for training. OS-UVC uses the ground truth masks for the first frame for mask generation. "Selected using GT" represents the masks generated by MP R-CNN selected using ground truth masks and can be regarded as an upper-bound.

we can achieve given the predictions of MP R-CNN.

Trained with ground truth. We report the performance of MP R-CNN fine-tuned with ground truth mask labels of Unseen-VIS-train. This can also be regarded as an upper bound, where all the classes have already been Seen.

We report the results in Table 4.2. Compared to RVOS [255] trained with labeled videos, our method can still achieve comparable results on recall while surpassing their results by a large margin on AP_{75}, which means that the masks selected by our method are of better quality. Importantly for practical applications, our method was able to deal with the large domain gap between the images in COCO on which we pre-train MP R-CNN and the frames in YouTube VIS on which we apply and evaluate our method. RVOS was trained and applied on videos from YouTube VIS, and therefore was not confronted to a domain gap.

While OS-UVC [151] achieves higher results than our approach, it relies mainly on a high-quality first frame mask: We observe a large performance drop when we replace the ground truth masks ("OS-UVC" in Table 4.2) by the predicted masks ("ZS-UVC" in Table 4.1). Besides, it can only track objects visible in the first frame as it does not handle

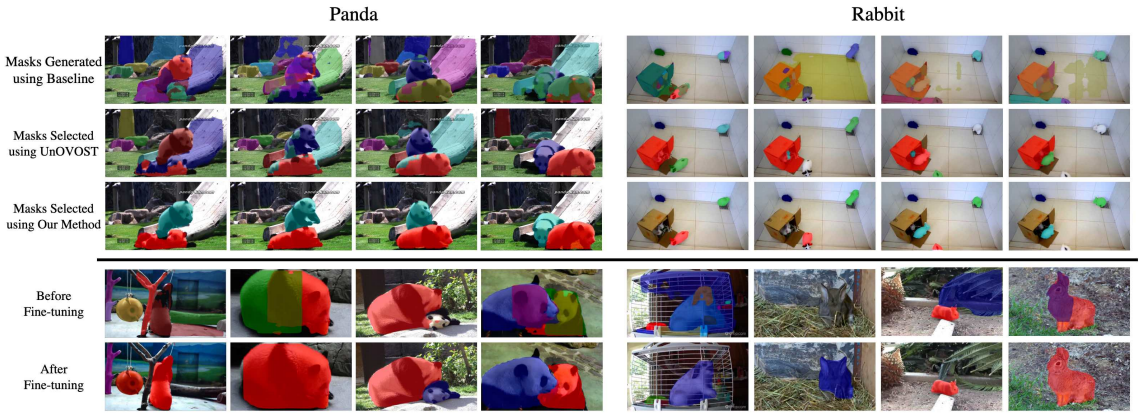


Figure 4.12: Qualitative results of selected masks on Unseen-VIS-train and detections of new classes on Unseen-VIS-test after fine-tuning on these selected masks. **Top:** First row: Masks detected by our baseline network MP R-CNN on two sequences from Unseen-VIS-train; Second row: Masks selected by UnOVOST [169]; Third row: Masks selected by our approach. Note that we keep the masks for the pandas and rabbits, and reject the masks that do not correspond to real objects. **Bottom:** Masks detected in still images from Unseen-VIS-test. Fourth row: Masks detected by MP R-CNN before we fine-tuned it on the masks selected by our approach on Unseen-VIS-train; Fifth row: Masks detected by MP R-CNN *after* fine-tuning. The masks generated by our method results in a significantly better model for the new classes: We can now correctly segment pandas and rabbits in new videos, even if no manual segmentations for pandas and rabbits were provided.

the emergence of new objects.

After simply retraining from scratch both on the 80 seen classes of COCO and the masks generated by our approach on Unseen-VIS-train, MP R-CNN achieves 35.2 mask AP on COCO minival and 38.9 mask AP on Unseen-VIS-test. Compared with the MP R-CNN pre-trained only with COCO dataset, which achieves 35.3 mask AP on COCO minival, we achieve better performance on Unseen-VIS-test while maintaining the performance on COCO.

4.5.4 Application to Zero-Shot Video Object Segmentation

As one of the state-of-the-art zero-shot video object segmentation methods, UnOVOST [169] segments the objects in the videos by linking the masks predicted by an instance segmentation network on each frame. As show in Table 4.3, by fine-tuning the original mask generation model on the masks generated on the DAVIS training dataset [208] using our

Method	*Use Unlabeled Data*	$\mathcal{J}$ and $\mathcal{F}$ **Mean**	$\mathcal{J}$ **Mean**	$\mathcal{J}$ **Recall**	$\mathcal{J}$ **Decay**	$\mathcal{F}$ **Mean**	$\mathcal{F}$ **Recall**	$\mathcal{F}$ **Decay**
UnOVOST [169]		56.2	54.4	63.7	-0.01	57.9	65.0	0.00
UnOVOST+	✓	**59.9**	**59.1**	**70.0**	**-0.06**	**60.8**	**70.8**	**-0.03**

Table 4.3: Zero-Shot Video Object Segmentation evaluation on the DAVIS dataset [208]. UnOVOST [169] relies on an instance segmentation network for mask generation. UnOVOST+ row: After fine-tuning its mask generation network on the DAVIS training dataset using the masks generated by our approach, it achieves higher results on all metrics.

method, we achieve much better results. This demonstrates that downstream tasks can benefit from the performance boost brought by our method.

4.5.5 Ablation Study

Table 4.4 shows the positive impact made by each loss term and constraint in Eq. (4.6). The masks obtained by applying only the background loss $\mathcal{L}_I$ can already improve the performance of baseline on unseen classes. Similarly, adding the constraint that the masks should not overlap, the flow loss $\mathcal{L}_F$, or the regularization loss $\mathcal{L}_p$ has a positive impact. In particular, this shows that both the flow loss $\mathcal{L}_F$ and the regularization loss $\mathcal{L}_p$ help reranking the combinations of masks given by the background loss.

4.5.6 Retraining vs Fine-tuning

Detailed results are shown in Table 4.5. The performance of MP R-CNN on COCO-2017-val degrades dramatically after finetuning on either the ground truth of Unseen-VIS-train or the masks generated by our approach. Our guess is that this degradation results from the significant domain gap between images in YouTube-VIS and COCO. Interestingly, compared with the ground truth of Unseen-VIS-train, the performance of MP R-CNN is much less affected by finetuning on the masks generated by our approach, which demonstrates that our approach helps to preserve to some extent the information of the dataset used for pre-training.

$\mathcal{L}_I$	Overl.Constr.	$\mathcal{L}_F$	$\mathcal{L}_p$	AP	AP_{50}	AP_{75}	AR_1	AR_3	AR_5
				35.8	61.2	38.1	33.3	47.3	50.3
✓				36.6	65.2	37.8	33.3	47.7	50.8
✓	✓			38.1	64.6	40.2	34.2	48.5	51.2
✓	✓	✓		38.7	67.0	40.5	34.7	48.7	51.3
✓	✓	✓	✓	39.0	67.9	41.3	35.2	48.9	51.4

Table 4.4: Ablation study on the different components of our method.

	Unseen-VIS-train		Training	Unseen-VIS-test						COCO val					
Pretrained on COCO	GT	Generated Masks	Strategy	AP	AP_{50}	AP_{75}	AR_1	AR_3	AR_5	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
✓			–	35.8	61.2	38.1	33.3	47.3	50.3	35.3	62.6	35.9	20.8	40.6	52.4
✓	✓		retrain	51.2	80.2	56.8	45.0	56.8	59.5	34.9	61.8	35.7	20.6	40.2	52.3
✓		✓	retrain	38.9	67.8	41.2	35.2	48.9	51.4	35.0	62.0	35.8	20.5	40.2	52.3
✓	✓		finetune	50.8	80.9	54.6	43.6	58.6	60.6	24.0	44.9	23.4	15.3	28.7	33.6
✓		✓	finetune	39.0	67.9	41.3	35.2	48.9	51.4	32.1	57.0	32.8	17.5	39.3	51.1

Table 4.5: Results on the COCO2017 validation dataset and Unseen-VIS-test adopting different training strategies. The performance of MP R-CNN fine-tuned on our generated masks on Unseen-VIS-train is much less affected compared to MP R-CNN fine-tuned on the ground truth masks. By training on the dataset created by naively mixing the training dataset of COCO with our generated masks on Unseen-VIS-train, we can achieve the same results on Unseen classes as fine-tuning while preserving the performance on Seen classes.

4.6 Conclusion

In this chapter, we tackled the challenge of localizing and segmenting objects from unseen classes without any manual mask labels. We demonstrated that our method, based on an instance segmentation model pretrained on some seen classes, generates high-quality masks for unseen classes by analyzing unlabeled videos without the need for hard-to-tune hyper-parameters. Furthermore, we presented an efficient implementation by breaking down the computationally expensive optimization into a two-stage process.

However, it should be noted that in the unsupervised case, the concept of objects is often ill-defined. The boundary between ”things” and ”stuff” [25, 68] can be ambiguous at times. For instance, should we consider a stone on the ground as an object or part of the background? What if a person pushes this stone? Furthermore, the granularity of the problem is also not clear. Should we consider a person as one object, or each item of

clothing they wear as individual objects? This ambiguity does not arise in the supervised case, as the annotators determine what constitutes an object, but it makes the evaluation of unsupervised object detection difficult. It may be essential to reconsider the definition of object based on either its shape or function to make the evaluation of unsupervised object segmentation more meaningful.

PART II:

HUMAN MOTION UNDERSTANDING

In this part of the thesis, we tackle the human motion understanding problem, where we aim to comprehend and forecast human motion from historical observations or partial observations. Indeed, it is crucial for the robots to understand the human that they are interacting with and prevent risky movements.

This part consists of two chapters, in the Chapter 5, we introduce a simple yet effective method for forecasting future body poses from historically observed sequences. In particular, we show that a light-weight network based on multi-layer perceptrons(MLPs) with only 0.14 million parameters can achieve superior performance on various benchmarks.

In the Chapter 6, we tackle the conditional motion synthesis problem, where only a sparse tracking signal is available from standalone devices. We present a novel conditional diffusion model specifically designed to track full bodies given sparse upper-body tracking signals. It can predict accurate and smooth full-body motion, particularly the challenging lower body movement. Our compact architecture can run in real-time, making it suitable for online body-tracking applications. We demonstrate that our approach outperforms state-of-the-art methods in generated motion accuracy and smoothness.

CHAPTER 5

A SIMPLE BASELINE FOR HUMAN MOTION PREDICTION

In this chapter, we introduce our simple yet effective method for forecasting short time human motion based on historical observations. The ability to anticipate short-term human motion holds tremendous significance across a multitude of domains, offering key contributions to their success and progress. Our method is designed to be not only accurate but also simple and adaptable, thus making it applicable to a wide array of applications. In the following sections, we delve into the technical details of our approach.

5.1 INTRODUCTION

The objective of human motion prediction is to forecast the subsequent 3D body poses in a sequence. Precise prediction of future human motion is crucial for several applications, including accident prevention in autonomous driving [203], people tracking [80], and human-robot interaction [133].

Due to the spatio-temporal nature of human motion, the prevailing trend in the literature is to design models that can integrate spatial and temporal information. Traditional approaches primarily relied on hidden Markov models [19] or Gaussian process latent variable models [260]. However, while these approaches performed well on simple and periodic motion patterns, they were severely inadequate for complex motions [178]. In recent years, with the success of deep learning, various methods based on different types of neural networks have been developed to handle sequential data. For instance, some works use Recurrent Neural Networks (RNN)[182] to model human motion[69, 111, 182, 159, 40], and more recent works [178, 177, 84, 172, 43, 149, 148] propose networks based on Graph Convolutional Networks (GCN)[178], or try with Transformers[5]-based methods [177, 5, 26] to fuse spatial and temporal information of the motion sequence across human joints and time. However, the architectures of these recent methods are often complex, and some of them require additional priors, which makes their network difficult to analyze and modify. Therefore, a natural question arises: *"Can we address human motion prediction with a simple network?"*

To address this question, we initially attempted a naive solution by simply repeating

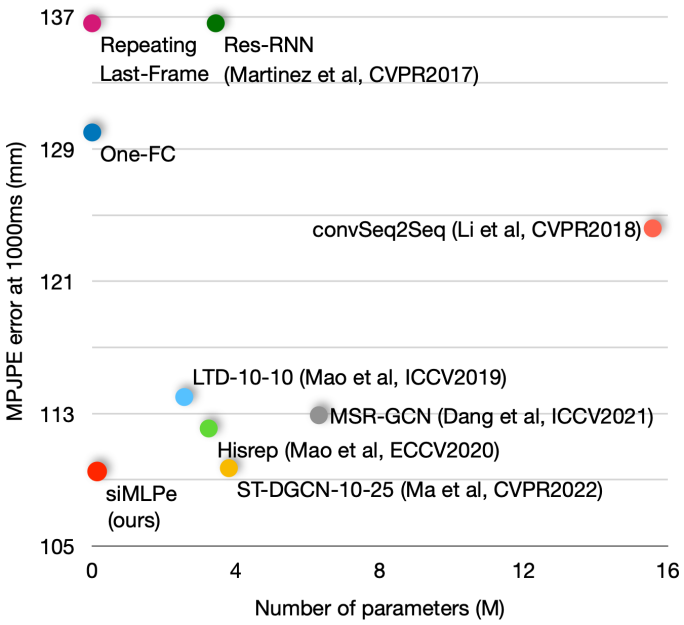


Figure 5.1: Comparison of parameter size and performance on the Human3.6M dataset [108]. We report the MPJPE metric in *mm* at 1000 ms as performance on the vertical axis. Our method (SIMLPE, in red) achieves the lowest error with significantly fewer parameters. We also show the performance of two simple methods: 'Repeating Last-Frame' systematically repeats the last input frame as output prediction, and 'One-FC' uses only one single fully connected layer to predict the future motion. *The closer to the bottom-left, the better.*

the last input pose and utilizing it as the output prediction. As depicted in Figure 5.1, this naive solution already achieved reasonable results, indicating that the last input pose is "close" to the future poses. Motivated by this, we trained only one fully connected layer to predict the residual between the future poses and the last input pose, achieving better performance, demonstrating the potential of a simple network for human motion prediction built on fundamental layers such as the fully connected layer.

Based on the above observations, we return to multi-layer perceptrons (MLPs) and develop a simple yet effective network named SIMLPE, which comprises only three learnable components: fully connected layers, layer normalization [11], and 1D convolutional layer with kernel size 1 [140]. The network architecture is illustrated in Figure 5.2. Notably, we discovered that even commonly used activation layers such as ReLU [193] are unnecessary, rendering our network a completely linear model except for layer normal-

ization. Despite its simplicity, siMLPe achieves strong performance when appropriately combined with three straightforward practices: applying the Discrete Cosine Transform (DCT), predicting the residual displacement of joints, and optimizing velocity as an auxiliary loss.

siMLPe attains state-of-the-art performance on several standard benchmarks, including Human3.6M [108], AMASS [174], and 3DPW [258]. Moreover, siMLPe is lightweight and necessitates $20\times$ to $60\times$ fewer parameters than previous state-of-the-art approaches. Figure 5.1 depicts a comparison between siMLPe and previous methods, demonstrating the Mean Per Joint Position Error (MPJPE) at $1,000$ms on Human3.6M of different networks versus their complexity. siMLPe achieves the best performance with high efficiency.

In summary, our contributions are as follows:

- We show that human motion prediction can be modeled in a simple way without explicitly fusing spatial and temporal information. As an extreme example, a single fully connected layer can already achieve reasonable performance.
- We propose siMLPe, a simple yet effective network for human motion prediction with only three learnable components: fully connected layers, layer normalization, and 1D convolutional layer with kernel size 1, achieving state-of-the-art performance with far fewer parameters than existing methods on multiple benchmarks such as Human3.6M, AMASS and 3DPW datasets.

5.2 Our Approach: siMLPe

In this section, we formulate the problem and present the formulation of the DCT transformation in Section 5.2.1, details of the network architecture in Section 5.2.2, and the losses we use for training in Section 5.2.3.

Given a sequence of 3D human poses in the past, our goal is to predict the future sequence of poses. We denote the observed 3D human poses as $\mathbf{P}_{1:T} = [P_1^\top, .., P_T^\top]^\top \in$

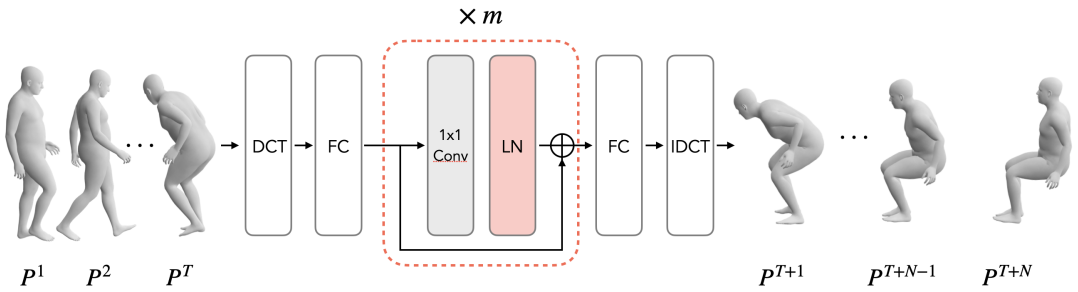


Figure 5.2: Overview of our approach SIMLPE for human motion prediction. *FC* denotes a fully connected layer, *LN* denotes layer normalization [11] and *1x1 Conv* denotes a 1D convolutional layer with kernel size 1. *DCT* and *IDCT* represent the discrete cosine transformation and inverse discrete cosine transformations respectively. The MLP blocks (in gray), composing FC and LN, are repeated m times.

$\mathbb{R}^{T\times C}$, consisting of T consecutive human poses, where the pose at the t-th frame P_t is represented by a C-dimensional vector, i.e. $P_t \in \mathbb{R}^C$. In this work, similar to previous works [182, 178, 177, 172], P_t is the 3D coordinates of joints at t-th frame and $C = 3 \times \mathbf{J}$, where $\mathbf{J}$ is the number of joints. Our task is to predict the future N motion frames $\mathbf{G}_{T+1:T+N} = [P_{T+1}^\top, .., P_{T+N}^\top]^\top \in \mathbb{R}^{N\times C}$.

5.2.1 Discrete Cosine Transform (DCT)

We adopt the DCT transformation to encode temporal information, which is proven to be beneficial for human motion prediction [178, 177, 172]. More precisely, given an input motion sequence of T frames, the DCT matrix $\mathbf{D} \in \mathbb{R}^{T\times T}$ can be calculated as:

$$\mathbf{D}_{i,j} = \sqrt{\frac{2}{T}} \frac{1}{\sqrt{1+\delta_{i,0}}} \cos\left(\frac{\pi}{2T}(2j+1)i\right) , \tag{5.1}$$

where $\delta_{i,j}$ denotes the *Kronecker* delta:

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \, . \end{cases} \tag{5.2}$$

The transformed input is $\mathcal{D}(\mathbf{P}_{1:T}) = \mathbf{D}\mathbf{P}_{1:T}$. We apply the Inverse Discrete Cosine Transform (IDCT) to transform the output of the network back to the original pose representation, denoted as $\mathcal{D}^{-1}$ and the inverse of $\mathbf{D}$. Note that the DCT and IDCT operations can be also merged to the first and last fully connected layers by matrix muliplication once the network is trained.

5.2.2 Network Architecture

Figure 5.2 depicts the architecture of our proposed network, which consists of only three components: fully connected layers, 1D convolutional layer with kernel size 1, and layer normalization [11]. The input dimension of each fully connected layer is the same as its output dimension.

Formally, given an input sequence of 3D human poses $\mathbf{P}_{1:T} = [P_1^\top, .., P_T^\top]^\top \in \mathbb{R}^{T\times C}$, our network predicts a sequence of future poses $\mathbf{P}_{T+1:T+N} = [P'^\top_{T+1}, .., P'^\top_{T+N}]^\top \in \mathbb{R}^{N\times C}$:

$$\mathbf{P}_{T+1:T+N} = \mathcal{D}^{-1}(\mathcal{F}(\mathcal{D}(\mathbf{P}_{1:T}))) \,, \tag{5.3}$$

where $\mathcal{F}$ denotes our network.

After the DCT transformation, we apply one fully connected layer to operate only on the spatial dimension of the transformed motion sequence $\mathcal{D}(\mathbf{P}_{1:T}) \in \mathbb{R}^{T\times C}$:

$$\mathbf{z}^0 = \mathcal{D}(\mathbf{P}_{1:T})\mathbf{W}_0 + \mathbf{b}_0 \,, \tag{5.4}$$

where $\mathbf{z}^0 \in \mathbb{R}^{T\times C}$ is the output of the fully connected layer, $\mathbf{W}_0 \in \mathbb{R}^{C\times C}$ and $\mathbf{b}_0 \in \mathbb{R}^C$ represent the learnable parameters of the fully connected layer.

Then, a series of m blocks are introduced to only operate on the temporal dimension, i.e., only to merge information across frames. Each block consists of a 1D convolutional layer with kernel size 1 followed by layer normalization, formally:

$$\mathbf{z}^i = \mathbf{z}^{i-1} + \mathrm{LN}(\mathbf{W}_i\mathbf{z}^{i-1} + \mathbf{b}_i) \,, \tag{5.5}$$

where $\mathbf{z}^i \in \mathbb{R}^{T \times C}, i \in [1, .., m]$ denotes the output of the i-th MLP block, LN denotes the layer normalization operation, and $\mathbf{W}_i \in \mathbb{R}^{T \times T}$ and $\mathbf{b}_i \in \mathbb{R}^T$ are the learnable parameters of the 1D convolutional layer with kernel size 1 in the i-th MLP block.

Finally, after the MLP blocks, we add another fully connected layer to operate only on the spatial dimension of the feature. The output of this layer is then passed through an Inverse Discrete Cosine Transform (IDCT) to obtain the final prediction:

$$\mathbf{P}_{T+1:T+N} = \mathcal{D}^{-1}(\mathbf{z}'\mathbf{W}_{m+1} + \mathbf{b}_{m+1}) \,, \tag{5.6}$$

where $\mathbf{W}_{m+1}$ and $\mathbf{b}_{m+1}$ are the learnable parameters of the last fully connected layer.

Note that the length of the input sequence T and the number of joints N do not necessarily need to be equal. When $T > N$, we only take the first N frames of the prediction. In the case of $T < N$, we can pad our input sequence to N by repeating the last frame, as done in [178, 177]. This padding approach helps to make the input sequence size consistent with the network input size.

5.2.3 LOSSES

As mentioned in Section 5.1 and shown in Figure 5.1, the last input pose is "close" to the future poses. Inspired by this observation, instead of predicting the absolute 3D poses from scratch, we let our network predict the residual between the future pose P_{T+t} and the last input pose x_T. As we will show in Section 5.3.6, this eases learning and improves performance.

Our objective function $\mathcal{L}$ includes two terms $\mathcal{L}_{re}$ and $\mathcal{L}_v$:

$$\mathcal{L} = \mathcal{L}_{re} + \mathcal{L}_v \,. \tag{5.7}$$

$\mathcal{L}_{re}$ aims to minimize the $\mathcal{L}_2$-norm between the predicted poses $\mathbf{P}_{T+1:T+N}$ and ground-truth one $\mathbf{G}_{T+1:T+N}$:

$$\mathcal{L}_{re} = \mathcal{L}_2(\mathbf{P}_{T+1:T+N}, \mathbf{G}_{T+1:T+N}) \,. \tag{5.8}$$

$\mathcal{L}_v$ aims to minimize the $\mathcal{L}_2$-norm between the velocity of the predicted motion $\mathbf{v}^{\mathbf{GT}}{}_{T+1:T+N}$ and the ground truth one $\mathbf{v}_{T+1:T+N}$:

$$\mathcal{L}_v = \mathcal{L}_2(\mathbf{v}^{\mathbf{GT}}{}_{T+1:T+N}, \mathbf{v}_{T+1:T+N}) \,, \tag{5.9}$$

where $\mathbf{v}_{T+1:T+N} = [v_{T+1}^\top, .., v_{T+N}^\top]^\top \in \mathbb{R}^{N \times C}$, v_t represents the velocity at frame t and is computed as the time difference: $v_t = P_{t+1} - P_t$. We provide a full analysis of the loss terms in Section 5.3.6.

5.3 Experiments

This section covers our experimental setup and results. We begin by discussing the datasets and evaluation metrics in Section 5.3.1. In Section 5.3.3, we provide details of our implementation. In Section 5.3.4, we present the quantitative and qualitative results of our proposed method. Finally, we conduct an ablation analysis to evaluate the contribution of different components of our approach in Section 5.3.6.

5.3.1 Datasets

Human3.6M dataset [108]. The Human3.6M dataset consists of 7 actors performing 15 actions, and 32 joints are labeled for each pose. We adopt the same testing protocols as [177] and use *S5* as the test set, *S11* as the validation set, and the remaining subjects as the train set. Different test sampling strategies have been used in previous works, including 8 samples per action [182, 178], 256 samples per action [177] or all samples in the test set [43]. Since 8 samples are too few and taking all testing samples may lead to an imbalance between different actions with varying sequence lengths, we follow the strategy of using 256 samples per action for testing and evaluate the results on 22 joints, as in [182, 178, 177, 172].

AMASS dataset [174] AMASS is a collection of multiple Mocap datasets [67, 174, 9, 135, 248, 251, 17, 166, 74, 31, 232, 176, 161, 192, 4, 252, 103, 249] unified by SMPL

parameterization [163]. We follow [177] to use AMASS-BMLrub [248] as the test set and split the rest of the AMASS dataset into training and validation sets. The model is evaluated on 18 joints as in [177].

3DPW dataset [258]. 3DPW is a dataset that consists of indoor and outdoor scenes, and provides 3D body pose annotations for 26 joints. However, to evaluate the generalization ability of our model, we follow the same evaluation protocol as [177] and only use 18 joints for evaluation, which are the same as the joints in the AMASS dataset.

5.3.2 Evaluation Metrics

We report the Mean Per Joint Position Error (MPJPE) on 3D joint coordinates as our primary evaluation metric, which is the most widely used metric for evaluating 3D pose errors. Specifically, we calculate the average L2-norm across different joints between the predicted and ground-truth poses. Following the common practice in previous works [178, 177, 43, 172], we ignore the global rotation and translation of the poses and sample the poses at a fixed rate of 25 frames per second (FPS) for all datasets. Note that for the Human3.6M dataset, we evaluate our method on 22 joints as in [182, 178, 177, 172], while for the 3DPW dataset, we evaluate our method on 18 joints as in [177] to evaluate the generalization ability of our model.

5.3.3 Implementation Details

In practice, we set the length of the input sequence to $T = 50$ and the length of the output sequence to $N = 10$ for the Human3.6M dataset, and $N = 25$ for the AMASS and 3DPW datasets. During testing, we apply our model in an auto-regressive manner to generate motion for longer periods. The feature dimension is set to $C = 3 \times \mathbf{J}$, where $\mathbf{J}$ is the number of joints in the dataset. Specifically, we set $\mathbf{J} = 22$ for the Human3.6M dataset and $\mathbf{J} = 18$ for the AMASS and 3DPW datasets.

To train our network, we set the batch size to 256 and use the Adam optimizer [127]. Our network consumes approximately $1.5GB$ of memory during training and all experi-

ments are conducted using the Pytorch [206] framework on a single NVIDIA RTX 2080Ti graphics card. We train our network on the Human3.6M dataset for 35k iterations, where the learning rate starts from 0.0003 at the beginning and drops to 0.00001 after 30k steps. The training takes around 30 minutes. On the AMASS dataset, we train our network for 115k iterations, where the learning rate starts from 0.0003 at the beginning and drops to 0.00001 after 100k steps. The training takes approximately 2 hours. During training, we only apply the front-back flip as data augmentation, which randomly inverts the motion sequence during the training.

5.3.4 Quantitative Results

In this section, we compare our approach to existing state-of-the-art methods on different datasets. We report MPJPE in *mm* at different prediction time steps up to 1000ms.

Human3.6M dataset. In Table 5.1, we compare our method with other state-of-the-art methods on the Human3.6M dataset. Our method outperforms all previous methods on every frame with much fewer parameters. As explained in Section 5.3.1, some different methods have taken different test sampling strategies. Following [177], we choose to test with 256 samples on 22 joints. To make a fair comparison, we evaluate all the methods using the same testing protocol. Our method outperforms all previous methods on every frame with a much less number of parameters. Besides, previous works usually report short-term ($0 \sim 500ms$) and long-term ($500 \sim 1000ms$) predictions separately, and [172] reports short-/long- term results using two different models. In our tables, all the results from $0 \sim 1000ms$ are predicted by a single model, and for [172], we report the results of their model which achieves the best performance on long-term prediction. In addition, we also evaluate the two simple approaches mentioned in Section 5.1 on the Human3.6M dataset in Table 5.1: 'Repeating Last-frame' takes the last input pose and repeats it N times to serve as output, and 'One FC' uses only one single fully connected layer trained on Human3.6m. These results show that the task of human motion prediction could be potentially modeled in a completely different and simple way without explicitly fusing

spatial and temporal information. Furthermore, similar to all the previous works, we also detail the action-wise results in Table 5.2.

AMASS and 3DPW datasets. In Table 5.3, we report the performance of the model trained on AMASS and tested on the AMASS-BMLrub and 3DPW datasets, following the evaluation protocol of [177]. Different from the Human3.6M dataset where the training and testing data are from the same types of actions performed by different actors, the difference between training and testing data under this protocol is much larger, which makes the task more challenging in terms of generalization. As shown in the table, our approach performs consistently better on long-term prediction. Moreover, our model is much lighter. For example, the parameter size of our model is $\sim 4\%$ of Hisrep [177].

While it is common to evaluate the predicted error at different time steps, some recent works, such as [234] and [299], report the average error from the first time step to a certain time step. In our experiments, we report the predicted error at different time steps in all tables except Table 5.4, where we report the average error for comparison with these two methods. It is worth noting that our approach outperforms both [234] and [299] in terms of average error.

5.3.5 Qualitative Results

In addition to presenting quantitative results, we also provide qualitative results of our method in Figure 5.3, which shows some testing examples on the Human3.6M dataset. These examples illustrate that our predictions perfectly match the ground-truth on short-term predictions and generally fit the ground-truth on long-term predictions. However, the error increases when predicting longer-term motions, which is a common problem for all motion prediction methods, as shown in Table 5.1 and Table 5.3. This is because most current methods use auto-regression for predicting a longer future, which causes the error to accumulate. Additionally, uncertainty increases rapidly with time when predicting human motions.

Time (ms)	MPJPE (mm) ↓ 80	160	320	400	560	720	880	1000	# Param.(M) ↓
Repeating Last-Frame	23.8	44.4	76.1	88.2	107.4	121.6	131.6	136.6	0
One FC	14.0	33.2	68.0	81.5	101.7	115.1	124.8	130.0	0.003
Res-RNN † [182]	25.0	46.2	77.0	88.3	106.3	119.4	130.0	136.6	3.44
convSeq2Seq † [145]	16.6	33.3	61.4	72.7	90.7	104.7	116.7	124.2	15.58
LTD-50-25 † [178]	12.2	25.4	50.7	61.5	79.6	93.6	105.2	112.4	2.56
LTD-10-10 † [178]	11.2	23.4	47.9	58.9	78.3	93.3	106.0	114.0	2.55
Hisrep † [177]	10.4	22.6	47.1	58.3	77.3	91.8	104.1	112.1	3.24
MSR-GCN ⋆ [43]	11.3	24.3	50.8	61.9	80.0	-	-	112.9	6.30
ST-DGCN-10-25 ⋆ [172]	10.6	23.1	47.1	57.9	76.3	90.7	102.4	109.7	3.80
siMLPe (Ours)	**9.6**	**21.7**	**46.3**	**57.3**	**75.7**	**90.1**	**101.8**	**109.4**	**0.14**

Table 5.1: Results on Human3.6M for different prediction time steps (ms). We report the MPJPE error in *mm* and number of parameters (M) for each method. Lower is better. 256 samples are tested for each action. † indicates that the results are taken from the paper [177], ⋆ indicated that the results are taken from the paper [172]. Note that ST-DGCN [172] use two different models to evaluate their short-/long- term performance, here we report their results of a single model which performs better on long-term for fair comparison. We also show results of two simple baselines: 'Repeating Last-Frame' repeats the last input frame 25 times as output, 'One FC' uses only one single fully connected layer for the prediction.

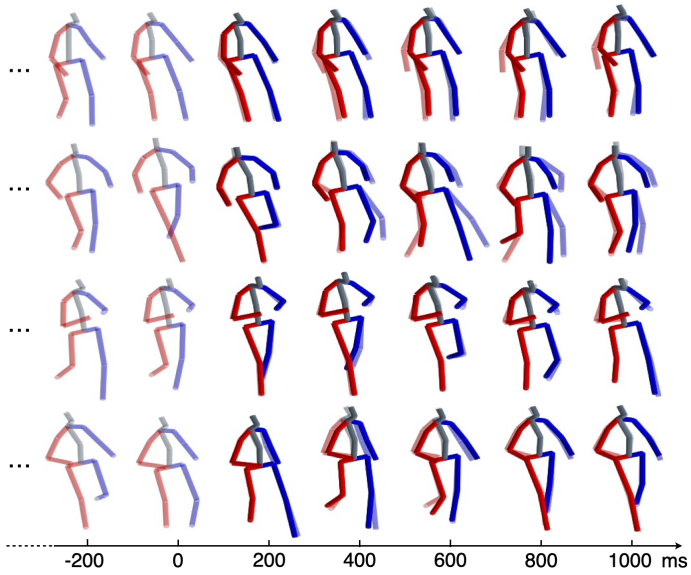


Figure 5.3: Qualitative results of our method siMLPe. The skeletons in light colors are the input (before 0ms) and the ground-truth (after 0ms). Those with dark colors represent the predicted motions. Our prediction results are close to the ground-truth.

Action	walking				eating				smoking				discussion			
Time (ms)	80	400	560	1000	80	400	560	1000	80	400	560	1000	80	400	560	1000
Res-RNN † [182]	23.2	66.1	71.6	79.1	16.8	61.7	74.9	98.0	18.9	65.4	78.1	102.1	25.7	91.3	109.5	131.8
convSeq2Seq † [145]	17.7	63.6	72.2	82.3	11.0	48.4	61.3	87.1	11.6	48.9	60.0	81.7	17.1	77.6	98.1	129.3
LTD-50-25 † [178]	12.3	44.4	50.7	60.3	7.8	38.6	51.5	75.8	8.2	39.5	50.5	72.1	11.9	68.1	88.9	118.5
LTD-10-10 † [178]	11.1	42.9	53.1	70.7	7.0	37.3	51.1	78.6	7.5	37.5	49.4	71.8	10.8	65.8	88.1	121.6
Hisrep † [177]	10.0	39.8	47.4	58.1	6.4	36.2	50.0	75.7	7.0	36.4	47.6	69.5	10.2	65.4	86.6	119.8
MSR-GCN ⋆ [43]	10.8	42.4	53.3	63.7	6.9	36.0	50.8	75.4	7.5	37.5	50.5	72.1	10.4	65.0	87.0	116.8
ST-DGCN-10-25 ⋆ [172]	11.2	42.8	49.6	58.9	6.5	36.8	50.0	74.9	7.3	37.5	48.8	69.9	10.2	64.4	86.1	116.9
siMLPe (Ours)	**9.9**	**39.6**	**46.8**	**55.7**	**5.9**	**36.1**	**49.6**	**74.5**	**6.5**	**36.3**	**47.2**	**69.3**	**9.4**	**64.3**	**85.7**	**116.3**
Action	directions				greeting				phoning				posing			
Time (ms)	80	400	560	1000	80	400	560	1000	80	400	560	1000	80	400	560	1000
Res-RNN † [182]	21.6	84.1	101.1	129.1	31.2	108.8	126.1	153.9	21.1	76.4	94.0	126.4	29.3	114.3	140.3	183.2
convSeq2Seq † [145]	13.5	69.7	86.6	115.8	22.0	96.0	116.9	147.3	13.5	59.9	77.1	114.0	16.9	92.9	122.5	187.4
LTD-50-25 † [178]	8.8	58.0	74.2	105.5	16.2	82.6	104.8	136.8	9.8	50.8	68.8	105.1	12.2	79.9	110.2	174.8
LTD-10-10 † [178]	8.0	54.9	76.1	108.8	14.8	79.7	104.3	140.2	9.3	49.7	68.7	105.1	10.9	75.9	109.9	171.7
Hisrep † [177]	7.4	56.5	73.9	106.5	13.7	78.1	101.9	138.8	8.6	49.2	67.4	105.0	10.2	75.8	107.6	178.2
MSR-GCN ⋆ [43]	7.7	56.2	75.8	105.9	15.1	85.4	106.3	**136.3**	9.1	49.8	67.9	104.7	10.3	75.9	112.5	176.5
ST-DGCN-10-25 ⋆ [172]	7.5	56.0	73.3	**105.9**	14.0	77.3	100.2	136.4	8.7	48.8	66.5	**102.7**	10.2	73.3	102.8	**167.0**
siMLPe (Ours)	**6.5**	**55.8**	**73.1**	106.7	**12.4**	**77.3**	**99.8**	137.5	**8.1**	**48.6**	**66.3**	103.3	**8.8**	**73.8**	**103.4**	168.7
Action	purchases				sitting				sittingdown				takingphoto			
Time (ms)	80	400	560	1000	80	400	560	1000	80	400	560	1000	80	400	560	1000
Res-RNN † [182]	28.7	100.7	122.1	154.0	23.8	91.2	113.7	152.6	31.7	112.0	138.8	187.4	21.9	87.6	110.6	153.9
convSeq2Seq † [145]	20.3	89.9	111.3	151.5	13.5	63.1	82.4	120.7	20.7	82.7	106.5	150.3	12.7	63.6	84.4	128.1
LTD-50-25 † [178]	15.2	78.1	99.2	134.9	10.4	58.3	79.2	118.7	17.1	76.4	100.2	143.8	9.6	54.3	75.3	118.8
LTD-10-10 † [178]	13.9	75.9	99.4	135.9	9.8	55.9	78.5	118.8	15.6	71.7	96.2	142.2	8.9	51.7	72.5	116.3
Hisrep † [177]	13.0	73.9	95.6	134.2	9.3	56.0	76.4	115.9	14.9	72.0	97.0	143.6	8.3	51.5	72.1	115.9
MSR-GCN ⋆ [43]	13.3	77.8	99.2	134.5	9.8	55.5	77.6	115.9	15.4	73.8	102.4	149.4	8.9	54.4	77.7	121.9
ST-DGCN-10-25 ⋆ [172]	13.2	74.0	95.7	**132.1**	9.1	54.6	75.1	114.8	14.7	**70.0**	**94.4**	**139.0**	8.2	**50.2**	**70.5**	112.9
siMLPe (Ours)	**11.7**	**72.4**	**93.8**	132.5	**8.6**	**55.2**	**75.4**	**114.1**	**13.6**	70.8	95.7	142.4	**7.8**	50.8	71.0	**112.8**
Action	waiting				walkingdog				walkingtogether				average			
Time (ms)	80	400	560	1000	80	400	560	1000	80	400	560	1000	80	400	560	1000
Res-RNN † [182]	23.8	87.7	105.4	135.4	36.4	110.6	128.7	164.5	20.4	67.3	80.2	98.2	25.0	88.3	106.3	136.6
convSeq2Seq † [145]	14.6	68.7	87.3	117.7	27.7	103.3	122.4	162.4	15.3	61.2	72.0	87.4	16.6	72.7	90.7	124.2
LTD-50-25 † [178]	10.4	59.2	77.2	108.3	22.8	88.7	107.8	156.4	10.3	46.3	56.0	65.7	12.2	61.5	79.6	112.4
LTD-10-10 † [178]	9.2	54.4	73.4	107.5	20.9	86.6	109.7	150.1	9.6	44.0	55.7	69.8	11.2	58.9	78.3	114.0
Hisrep † [177]	8.7	54.9	74.5	108.2	20.1	86.3	108.2	146.9	8.9	41.9	52.7	64.9	10.4	58.3	77.3	112.1
MSR-GCN ⋆ [43]	10.4	62.4	74.8	105.5	24.9	112.9	107.7	145.7	9.2	43.2	56.2	69.5	11.3	61.9	80.0	112.9
ST-DGCN-10-25 ⋆ [172]	8.7	53.6	71.6	**103.7**	20.4	84.6	105.7	145.9	8.9	43.8	54.4	64.6	10.6	57.9	76.3	109.7
siMLPe (Ours)	**7.8**	**53.2**	**71.6**	104.6	**18.2**	**83.6**	**105.6**	**141.2**	**8.4**	**41.2**	**50.8**	**61.5**	**9.6**	**57.3**	**75.7**	**109.4**

Table 5.2: Action-wise results on Human3.6M for different prediction time steps (ms). Lower is better. 256 samples are tested for each action. † indicates that the results are taken from the paper [177], ⋆ indicates that the results are taken from the paper [172].

Dataset	AMASS-BMLrub								3DPW							
Time (ms)	80	160	320	400	560	720	880	1000	80	160	320	400	560	720	880	1000
convSeq2Seq [145]	20.6	36.9	59.7	67.6	79.0	87.0	91.5	93.5	18.8	32.9	52.0	58.8	69.4	77.0	83.6	87.8
LTD-10-10 [178]	**10.3**	**19.3**	36.6	44.6	61.5	75.9	86.2	91.2	**12.0**	**22.0**	38.9	46.2	59.1	69.1	76.5	81.1
LTD-10-25 [178]	11.0	20.7	37.8	45.3	57.2	65.7	71.3	75.2	12.6	23.2	39.7	46.6	57.9	65.8	71.5	75.5
Hisrep [177]	11.3	20.7	35.7	42.0	51.7	58.6	63.4	67.2	12.6	23.1	39.0	45.4	56.0	63.6	69.7	73.7
siMLPe (Ours)	10.8	19.6	**34.3**	**40.5**	**50.5**	**57.3**	**62.4**	**65.7**	12.1	22.1	**38.1**	**44.5**	**54.9**	**62.4**	**68.2**	**72.2**

Table 5.3: Results on AMASS and 3DPW for different prediction time steps (ms). We report the MPJPE error in *mm*. Lower is better. The model is trained on the AMASS dataset. The results of the previous methods are taken from [177].

Dataset	Human3.6M								AMASS-BMLrub							
Time (ms)	80	160	320	400	560	720	880	1000	80	160	320	400	560	720	880	1000
STS-GCN [234]	10.1	17.1	33.1	38.3	50.8	60.1	68.9	75.6	10.0	12.5	21.8	24.5	31.9	38.1	42.7	45.5
STG-GCN [299]	10.1	16.9	32.5	38.5	50.0	-	-	72.9	10.0	11.9	20.1	24.0	30.4	-	-	43.1
siMLPe (Ours)	**4.5**	**9.8**	**22.0**	**28.1**	**39.3**	**49.2**	**57.8**	**63.7**	**6.1**	**10.8**	**19.1**	**22.8**	**29.5**	**35.1**	**39.7**	**42.7**

Table 5.4: Average results for different prediction time periods on Human3.6M and AMASS. These results are obtained following the evaluation method of STS-GCN [234] and STG-GCN [299], instead of the standard evaluation protocol adopted in [178, 177, 172].

Nb. Blocks	# Param.(M) ↓	MPJPE (mm) ↓							
		80	160	320	400	560	720	880	1000
1	0.012	12.7	28.5	59.7	72.1	93.6	107.0	116.8	123.6
2	0.014	10.9	24.9	52.3	64.0	83.2	97.3	108.4	115.4
6	0.025	10.2	23.1	48.8	60.1	79.0	93.3	105.1	112.6
12	0.041	9.9	22.4	47.2	58.3	77.1	91.5	103.3	110.9
24	0.073	9.7	22.0	46.8	57.7	76.4	90.8	102.6	110.3
48 (Ours)	0.138	**9.6**	**21.7**	**46.3**	**57.3**	**75.7**	**90.1**	**101.8**	**109.4**
64	0.180	9.6	21.8	46.5	57.5	76.0	90.1	101.9	109.7
96	0.266	9.7	21.9	46.7	57.8	76.3	90.5	102.1	109.8

Table 5.5: Ablation of the number of MLP blocks on Human3.6M. The network achieves the best performance with 48 MLP blocks.

5.3.6 Ablation Study

We evaluate below the influence of the different components of our approach on the Human3.6M dataset.

Ablation	80	160	320	400	560	720	880	1000
w/o 1D Conv	23.8	43.0	73.4	85.2	102.0	116.3	125.3	131.9
w/o FC	9.9	22.4	47.2	58.4	77.2	91.1	102.8	110.5
w/o LN	12.7	29.0	62.3	76.2	97.4	111.6	121.6	127.3
w/o DCT	9.9	22.4	47.3	58.4	76.9	91.2	102.8	110.5
siMLPe (ours)	**9.6**	**21.7**	**46.3**	**57.3**	**75.7**	**90.1**	**101.8**	**109.4**

Table 5.6: Ablation of different components of our network on Human3.6M. For the case of *'w/o 1D Conv'*, we replace the 1D convolutional layers with fully connected layers to maintain the network complexity. Similarly, in the case of *'w/o FC'*, we replace the fully connected layers with 1D convolutional layers with kernel size 1.

	80	160	320	400	560	720	880	1000
w/o aug	10.0	22.6	48.3	59.7	78.2	92.0	103.4	110.8
w aug	**9.6**	**21.7**	**46.3**	**57.3**	**75.7**	**90.1**	**101.8**	**109.4**

Table 5.7: Ablation of data augmentation on Human3.6M. We only use front-back flip as our data augmentation, i.e., we randomly invert the motion sequence during the training.

$\mathcal{L}_{re}$	$\mathcal{L}_v$	80	160	320	400	560	720	880	1000
✓		9.6	21.8	46.5	57.5	76.7	91.5	103.5	111.3
✓	✓	**9.6**	**21.7**	**46.3**	**57.3**	**75.7**	**90.1**	**101.8**	**109.4**

Table 5.8: Ablation of different loss terms on Human3.6M. The network performs better with both positional loss $\mathcal{L}_{re}$ and velocity loss $\mathcal{L}_v$.

Number of MLP blocks. We ablate the number of MLP blocks m in Table 5.5. Our proposed architecture already achieves good performance using only 2 MLP blocks with $0.014M$ parameters. The network achieves its best performance with 48 MLP blocks.

Network architecture. In Table 5.6, we ablate the different components of our network. As the table shows, temporal feature fusion and layer normalization are both of vital importance to our network. If the network just operates along the spatial dimension of the motion sequence without merging any information across different frames, it will lead to degraded results. However, if the network just operates along the temporal dimension, the

Residual	80	160	320	400	560	720	880	1000
w/o residual	12.4	25.1	50.7	61.6	80.1	93.9	105.5	113.0
Consecutive	9.7	22.0	46.8	57.8	76.5	90.7	102.4	110.1
Before IDCT	10.4	23.0	48.2	59.1	77.9	91.8	103.2	110.5
siMLPe (ours)	**9.6**	**21.7**	**46.3**	**57.3**	**75.7**	**90.1**	**101.8**	**109.4**

Table 5.9: Analysis of different types of residual displacement on Human3.6M. siMLPe predicts the differences of each future frame with the last observation (after IDCT). *'Before IDCT'* learns the residual before applying the IDCT transformation. *'Consecutive'* learns the velocity between consecutive frames. *'w/o residual'* predicts directly the absolute 3D poses.

network will still achieve comparable performance. Besides, the use of DCT transformation can further improve performance slightly.

Data augmentation. In Table 5.7, we ablate the use of front-back flip data augmentation and find that the data augmentation slightly improves the performance.

Losses. In Table 5.8, we evaluate the importance of different loss terms used during training. As shown in the table, with the help of the velocity loss $\mathcal{L}_v$, the network achieves better performance on long-term predictions while maintaining the same performance on the short-term.

Learning residual displacement. In Table 5.9, we analyze the importance of the proposed residual displacement and compare it to other types of residual used in previous works [182, 178]. Our method aims to predict the differences between each future pose and the last observed pose, after the IDCT transformation. When predicting directly the absolute 3D pose ('w/o residual'), the performance drops dramatically. We also test other types of residual by either learning the residual in the DCT space, before applying the IDCT transformation ('Before IDCT') following [178], or learning the velocity of the motion ('consecutive') following [182], and both achieve inferior performance compared to our proposed residual displacement.

5.4 CONCLUSION

We presented SIMLPE, a simple yet effective network for human motion prediction. SIMLPE is composed of fully connected layers, layer normalization, and 1D convlutional layer with kernel size 1, and achieves state-of-the-art performance on various benchmarks while using much fewer parameters compared to other state-of-the-art methods. Our reported ablation study highlights the importance of temporal information fusion in this task, and the simplicity of SIMLPE can serve as a baseline for future research. We hope that our work will inspire the community to rethink the problem of human motion prediction and encourage the development of simpler and more efficient models.

CHAPTER 6

CONDITIONAL MOTION SYNTHESIS FROM SPARSE SIGNALS

Beyond forecasting human motion from historical observations as described in Chapter 5, the prediction of whole body motion from partially observed keypoint movements also presents a practical scenario, particularly in the realm of human-robot interaction. Robots may be able to access motion data from a selection of keypoints via wearable or carry-on devices. In this chapter, we introduce AGRoL, a novel conditional diffusion model specifically designed to track full bodies given sparse upper-body tracking signals. Based on SIMLPE, our model adopts also a multi-layer perceptron (MLP) architecture and a novel conditioning scheme for motion data. It can predict accurate and smooth full-body motion, particularly the challenging lower body movement. Our compact architecture can run in real-time, making it suitable for online body-tracking applications.

6.1 INTRODUCTION

Tracking full-body movement is in high demand for AR/VR and robotic applications, where humans are the primary actors. While common approaches are capable of accurately tracking upper bodies, such as in [283] and [118], moving towards full-body tracking would unlock more engaging experiences where users can interact with the virtual environment with an increased sense of presence.

However, in the typical AR/VR setting there is no strong tracking signal for the entire human body – only the head and hands are usually tracked by means of Inertial Measurement Unit (IMU) sensors embedded in Head Mounted Displays (HMD) and hand controllers. Some works suggest adding additional IMUs to track the lower body joints [105, 118], those additions come at higher costs and the expense of the users comfort [124, 117]. In an ideal setting, we want to enable high-fidelity full-body tracking using the standard three inputs (head and hands) provided by most HMDs.

Given the limited tracking signal of only the head and both hands, predicting full-body pose is inherently an underconstrained problem. To address this challenge, different methods rely on generative models such as normalizing flows [226] and Variational Autoencoders (VAE)[52] to synthesize lower body motions. However, in recent years, diffu-

sion models have shown impressive results in image and video generation[235, 100, 197], especially for conditional generation. This motivates us to employ the diffusion model to generate fully-body poses conditioned on sparse tracking signals. To the best of our knowledge, no existing work leverages the diffusion model solely for motion reconstruction from sparse tracking information.

While the diffusion model has shown promise for conditional generation, its implementation in the task of motion synthesis presents challenges. Current approaches for using diffusion models in cross-modal conditional generation are not directly transferable to motion synthesis due to significant differences in data representations. As a result, novel methods need to be developed to adapt diffusion models for effective use in motion synthesis.

In this chapter, we propose a novel diffusion architecture – *Avatars Grow Legs* (AGRoL), which is specifically tailored for the task of conditional motion synthesis. Inspired by the siMLPe network proposed in Chapter 5, which uses an MLP-based architecture, we find that a carefully designed MLP network can achieve comparable performance to the state-of-the-art methods. However, we discovered that the predicted motions of MLP networks may contain jittering artifacts. To address this issue and generate smooth realistic full body motion from sparse tracking signals, we design a novel lightweight diffusion model powered by our MLP architecture. Diffusion models require time step embedding [100, 196] to be injected in the network during training and inference; however, we found that our MLP architecture is not sensitive to the positional embedding in the input. To tackle this problem, we propose a novel strategy to effectively inject the time step embedding during the diffusion process. With the proposed strategy, we can significantly mitigate the jittering issues and further improve the performance of the model and robustness against the loss of tracking signal. Our model accurately predicts full-body motions, outperforming state-of-the-art methods as demonstrated by the experiments on AMASS [174], large motion capture dataset. Please refer to our project page for more visual results.

We summarize our contributions as follows:

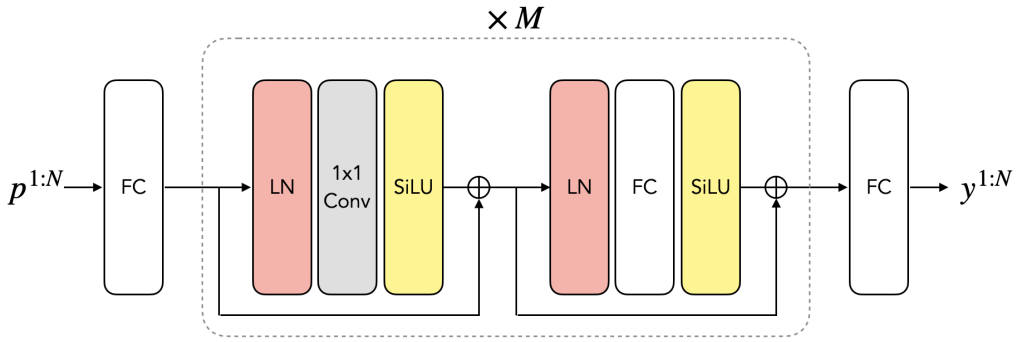


Figure 6.1: The architecture of our MLP-based network. *FC*, *LN*, and *SiLU* denote the fully connected layer, the layer normalization, and the SiLU activation layer respectively. *1 × 1 Conv* denotes the 1D convolution layer with kernel size 1. Note that *1 × 1 Conv* here is equivalent to a fully connected layer operating on the first dimension of the input tensor $\mathbb{R}^{N\times D}$, while the *FC* layers operate on the last dimension. N denotes the temporal dimension and D denotes the dimension of the latent space. The middle block is repeated M times. The first *FC* layer projects input data to a latent space $\mathbb{R}^{N\times D}$ and the last one converts from latent space to the output space of full-body poses $\mathbb{R}^{N\times S}$.

- We propose AGRoL, a conditional diffusion model specifically designed for full-body motion synthesis based on sparse IMU tracking signals. AGRoL is a simple and yet efficient MLP-based diffusion model with a lightweight architecture. To enable gradual denoising and produce smooth motion sequences we propose a block-wise injection scheme that adds diffusion timestep embedding before every intermediate block of the neural network. With this timestep embedding strategy, AGRoL achieves state-of-the-art performance on the full-body motion synthesis task without any extra losses that are commonly used in other motion prediction methods.

- We show that our lightweight diffusion-based model AGRoL can generate realistic smooth motions while achieving real-time inference speed, making it suitable for online applications. Moreover, it is more robust against tracking signals loss then existing approaches.

6.2 METHOD

6.2.1 PROBLEM FORMULATION

Our goal is to predict the whole body motion given sparse tracking signals, i.e. the orientation and translation of the headset and two hand controllers. To achieve this, we use a sequence of N observed joint features $p^{1:N} = \{p^i\}_{i=1}^N \in \mathbb{R}^{N \times C}$ and aim to predict the corresponding whole-body poses $y^{1:N} = \{y^i\}_{i=1}^N \in \mathbb{R}^{N \times S}$ for each frame. The dimensions of the input/output joint features are represented by C and S, respectively. We utilize the SMPL [163] model in this thesis to represent human poses and follow the approach outlined in [117, 52] to consider the first 22 joints of the SMPL model and disregard the joints on the hands and face. Thus, $y^{1:N}$ reflects the global orientation of the pelvis and the relative rotation of each joint. Following [117], during inference, we initially pose the human model using the predicted rotations. Next, we calculate the global translation by accounting for the known head translation and subtracting the offset between the root joint and the head joint.

In the following section, we first introduce a simple MLP-based network for full-body motion synthesis based on sparse tracking signals. Then, we show how we further improve the performance by leveraging the proposed MLP-based architecture to power the conditional generative diffusion model, termed AGRoL.

6.2.2 MLP-BASED NETWORK

Our network architecture comprises only four types of components commonly employed in the realm of deep learning: fully connected layers (FC), SiLU activation layers [220], 1D convolutional layers [140] with kernel size 1 and an equal number of input and output channels, as well as layer normalization (LN) [11]. It is worth noting that the 1D convolutional layer with a kernel size of 1 can also be interpreted as a fully connected layer operating along a different dimension. The details of our network architecture are demonstrated in Figure 6.1. Each block of the MLP network contains one convolutional

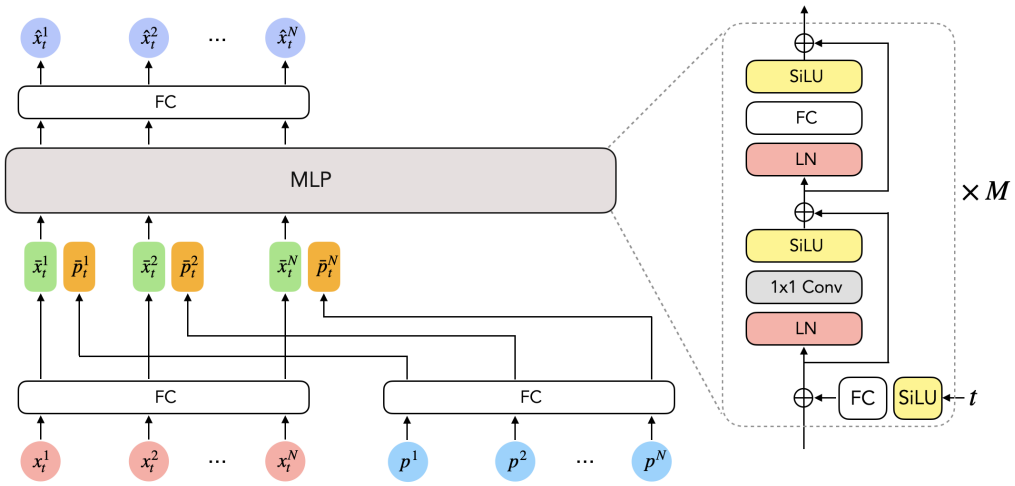


Figure 6.2: The architecture of our MLP-based diffusion model. t is the noising step. $x_t^{1:N}$ denotes the motion sequence of length N at step t, which is pure Gaussian noises when $t = T$. $p^{1:N}$ denotes the sparse upper body signals of length N. $\hat{x}_t^{1:N}$ denotes the denoised motion sequence at step t.

and one fully connected layer, which is responsible for temporal and spatial information merging respectively. We use skip-connections as in ResNets [95] with Layer Norm [11] as pre-normalization of the layers. First, we project the input data $p^{1:N}$ to a higher dimensional latent space using a linear layer. And the last layer of the network projects from the latent space to the output space of full-body poses $y^{1:N}$.

6.2.3 Diffusion Model

Diffusion model [100, 235] is a type of generative model which learns to reverse random Gaussian noise added by a Markov chain to recover desired data samples from the noise. In the forward diffusion process, given a sample motion sequence $x_0^{1:N} \sim q(x_0^{1:N})$ from the data distribution, the Markovian noising process can be written as:

$$q(x_t^{1:N}|x_{t-1}^{1:N}) := \mathcal{N}(x_t^{1:N}; \sqrt{\alpha_t}x_{t-1}^{1:N}, (1-\alpha_t)I), \tag{6.1}$$

where $\alpha_t \in (0, 1)$ is constant hyper-parameter and I is the identity matrix. $x_T^{1:N}$ tends to an isotropic Gaussian distribution when $T \to \infty$. Then, in the reverse diffusion process, a model p_θ with parameters θ is trained to generate samples from input Gaussian noise

$x_T \sim \mathcal{N}(0, I)$ with variance σ_t^2 that follows a fixed schedule. Formally,

$$p_\theta(x_{t-1}^{1:N}|x_t^{1:N}) := \mathcal{N}(x_{t-1}^{1:N}; \mu_\theta(x_t, t), \sigma_t^2 I), \tag{6.2}$$

where μ_θ could be reformulated [100] as

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \tag{6.3}$$

where $\bar{\alpha}_t = \alpha_1 \cdot \alpha_2 ... \cdot \alpha_t$. So the model has to learn to predict noise $\epsilon_\theta(x_t, t)$ from x_t and timestep t.

In our case, we want to use the diffusion model to generate sequences of full-body poses conditioned on the sparse tracking of joint features $p^{1:N}$. Thus, the reverse diffusion process becomes conditional: $p_\theta(x_{t-1}^{1:N}|x_t^{1:N}, p^{1:N})$. Moreover, we follow [223] to directly predict the clean body poses $x_0^{1:N}$ instead of predicting the residual noise $\epsilon_\theta(x_t, t)$. The objective function is then formulated as

$$\mathcal{L}_{dm} = \mathbb{E}_{x_0^{1:N} \sim q(x_0^{1:N}), t}\left[\| x_0^{1:N} - \hat{x}_0^{1:N} \|_2^2\right] \tag{6.4}$$

where the $\hat{x}_0^{1:N} = f_\theta(x^{1:N}, p^{1:N}, t)$ denotes the output of our model f_θ.

We use the MLP architecture proposed in Sect. 6.2.2 as the backbone for the model f_θ that predicts the full-body poses. At time step t, the motion features $x_t^{1:N}$ and the observed joints feature $p^{1:N}$ are first passed separately through a fully connected layer to obtain the latent features $\bar{x}_t^{1:N}$ and $\bar{p}^{1:N}$:

$$\bar{x}_t^{1:N} = \text{FC}_0(x_t^{1:N}), \tag{6.5}$$

$$\bar{p}^{1:N} = \text{FC}_1(p^{1:N}). \tag{6.6}$$

Then these features are concatenated together and fed to the MLP backbone: $\hat{x}_0^{1:N} = \text{MLP}(\text{Concat}(\bar{x}_t^{1:N}, \bar{p}^{1:N}), t)$.

When utilizing diffusion models, the embedding of the timestep t is often included as an additional input to the network. To achieve this, a common approach is to concatenate the timestep embedding with the input, similar to positional embedding used in transformer-based methods [254, 55]. However, since our network is mainly composed of FC layers, that mix the input features indiscriminately [246], the time step embedding information can easily be lost after several layers, which hinders learning the denoising process and results in predicted motions with severe jittering artifacts, as shown in Section 6.3.6. In order to address the issue of losing time step embedding information in our network, we introduce a novel strategy that repetitively injects the time step embedding into every block of the MLP network. This process involves projecting the timestep embedding to match the input feature dimensions through a fully connected layer and a SiLU activation layer. The details of our pipeline are shown in Figure 6.2. Unlike previous work, such as [100], which predicts a scale and shift factor for each block from the timestep embedding, our proposed approach directly adds the timestep embedding projections to the input activations of each block. Our experiments in Sect. 6.3 validate that this approach significantly reduces jittering issues and enables the synthesis of smooth motions.

6.3 Experiments

Our models are trained and evaluated on the AMASS dataset [174]. To compare with previous methods, we use two different settings for training and testing. In the first setting, we follow the approach of [117], which utilizes three subsets of AMASS: CMU [251], BMLr [248], and HDM05 [192]. In the second setting, we adopt the data split employed in several recent works, including [52, 6, 224]. This approach employs a larger set of training data, including CMU [251], MPI Limits [4], Total Capture [250], Eyes Japn [62], KIT [176], BioMotionLab [248], BMLMovi [74], EKUT [176], ACCAD [67], MPI Mosh [162], SFU [252], and HDM05 [192] as training data, while HumanEval [232] and Transition [174] serve as testing data.

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter	Upper Jitter	Lower Jitter
Final IK	16.77	18.09	59.24	-	-	-	-	-	-	-
LoBSTr	10.69	9.02	44.97	-	-	-	-	-	-	-
VAE-HMD	4.11	6.83	37.99	-	-	-	-	-	-	-
AvatarPoser*	3.08	4.18	27.70	2.12	1.81	7.59	**3.34**	14.49	7.36	24.81
MLP (Ours)	2.69	3.93	22.85	2.62	1.89	6.88	3.35	13.01	9.13	18.61
AGRoL (Ours)	**2.66**	**3.71**	**18.59**	**1.31**	**1.55**	**6.84**	3.36	**7.26**	**5.88**	**9.27**
GT	0	0	0	0	0	0	0	4.00	3.65	4.52

Table 6.1: Comparison of our approach with state-of-the-art methods on a subset of AMASS dataset following [117]. We report *MPJPE* [cm], *MPJRE* [deg], *MPJVE* [cm/s], Jitter [10^2m/s^3] metrics. AGRoL achieves the best performance on *MPJPE*, *MPJRE* and *MPJVE*, and outperforms other models, especially on the *Lower PE* (Lower body Position Error) and *Jitter* metrics, which shows that our model generates accurate lower body movement and smooth motions.

Method	MPJRE	MPJPE	MPJVE	Jitter
VAE-HMD† [52]	-	7.45	-	-
HUMOR† [224]	-	5.50	-	-
FLAG† [6]	-	4.96	-	-
AvatarPoser*	4.70	6.38	34.05	10.21
MLP (Ours)	4.33	6.66	33.58	21.74
AGRoL (Ours)	**4.30**	**6.17**	**24.40**	**8.32**
GT	0	0	0	2.93

Table 6.2: Comparison of our approach with state-of-the-art methods on AMASS dataset following the protocol of [52, 224, 6]. We report the MPJPE [cm], MPJRE [deg], MPJVE [cm/s], and Jitter [10^2m/s^3] metrics. The * denotes that we retrained the AvatarPoser using public code. † denotes methods that use pelvis location and rotation during inference, which are not directly comparable to our method, as we assume that the pelvis information is not available during the training and the testing. The best results are in bold, and the second-best results are underlined.

In both settings, we adopt the SMPL [163] human model for the human pose representation and train our model to predict the global orientation of the root joint and relative rotation of the other joints.

6.3.1 Implementation Details

We represent the joint rotations by the 6D reparametrization [301] due to its simplicity and continuity. Thus, for the sequences of body poses $y^{1:N} \in \mathbb{R}^{N \times S}$, $S = 22 \times 6$. The observed

joint features $p^{1:N} \in \mathbb{R}^{N \times C}$ consists of the orientation, translation, orientation velocity and translation velocity of the head and hands in global coordinate system. Additionally, we adopt 6D reparametrization for the orientation and orientation velocity, thus $C = 18 \times 3$. Unless otherwise stated, we set the frame number N to 196.

6.3.2 MLP Network

We build our MLP network using 12 blocks ($M = 12$). All latent features in the MLP network have the same shape of $N \times 512$. The network is trained with batch size 256 and Adam optimizer [127]. The learning rate is set to 3e-4 at the beginning and drops to 1e-5 after 200000 iterations. The weight decay is set to 1e-4 for the entire training. During inference, we apply our model in an auto-regressive manner for the longer sequences.

6.3.3 MLP-based Diffusion Model (AGRoL)

We keep the MLP network architecture unchanged in the diffusion model. To inject the time step embedding used in the diffusion process in the network, in each MLP block, we pass the time step embedding to a fully connected layer and a SiLU activation layer [220] and sum it with the input feature. The network is trained with exactly the same hyperparameters as the MLP network, with the exception of using the AdamW [164] as optimizer. During training, we set the sampling step to 1000 and employ a cosine noise schedule [197]. However, to expedite the inference speed, we leverage the DDIM [236] technique, which allows us to sample only 5 steps instead of 1000 during inference. All experiments were carried out on a single NVIDIA V100 graphics card, using the PyTorch framework [206].

6.3.4 Evaluation Metrics

In line with previous works [117, 289, 52, 224], we adopt nine evaluation metrics that we group into three categories.

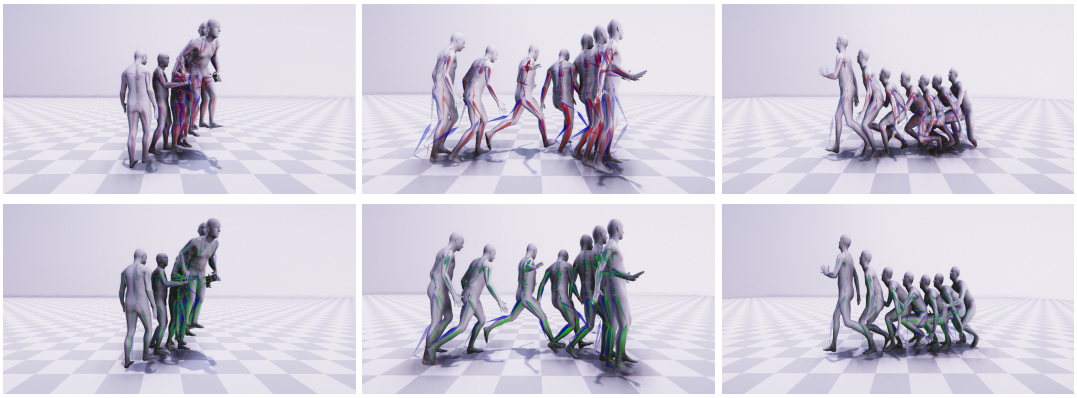


Figure 6.3: Qualitative comparison between AGRoL (top) and AvatarPoser [117] (bottom) on test sequences from AMASS dataset. We visualize the predicted skeletons and render human body meshes. **Top:** AvatarPoser predictions in red. **Bottom:** AGRoL predictions in green. In both rows, the blue skeletons denote the ground truth motion. We observe that motions predicted by AGRoL are closer to ground truth compared to the predictions of AvatarPoser.

Rotation-related metric. Mean Per Joint Rotation Error [degrees] (*MPJRE*) measures the average relative rotation error for all joints.

Velocity-related metrics. These include Mean Per Joint Velocity Error [cm/s] (*MPJVE*) and *Jitter*. *MPJVE* measures the average velocity error for all joints, while *Jitter* [289] evaluates the mean jerk (change in acceleration over time) of all body joints in global space, expressed in 10^2m/s^3. *Jitter* is an indicator of motion smoothness.

Position-related metrics. Mean Per Joint Position Error [cm] (*MPJPE*) quantifies the average position error across all joints. *Root PE* assesses the position error of the root joint, whereas *Hand PE* calculates the average position error for both hands. *Upper PE* and *Lower PE* estimate the average position error for joints in the upper and lower body, respectively.

6.3.5 Evaluation Results

We evaluate our method on the AMASS dataset with two different protocols. As shown in Table 6.1 and Table 6.2, our MLP network can already surpass most of the previous meth-

Method	#Params	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
AGRoL-AvatarPoser	2.89M	4.31	6.71	27.65	1.47	2.56	12.69	6.69	9.57
AGRoL-AvatarPoser-Large	7.63M	2.86	4.04	21.90	**1.29**	1.62	7.53	3.64	9.94
AGRoL-Transformer	7.03M	3.01	4.41	20.33	2.97	2.13	7.71	3.88	**6.45**
AGRoL (Ours)	7.48M	**2.66**	**3.71**	**18.59**	1.31	**1.55**	**6.84**	**3.36**	7.26

Table 6.3: Ablation study of network architectures in our diffusion model. We replace the proposed MLP backbone with other architectures and train several versions of the diffusion model with the same hyperparameters. The *AvatarPoser-Large* denotes the backbone with the same architecture as AvatarPoser [117] but with more transformer layers. *AGRoL-Transformer* is the AGRoL version with the transformer backbone from [244]. The *AGRol (ours)* with our MLP backbone outperforms all other backbones on most of the metrics.

ods and achieves comparable results with the state-of-the-art method [117], demonstrating the effectiveness of the proposed simple MLP architecture. By leveraging the diffusion process and the proposed MLP backbone, the AGRoL model remarkably boosts the performance of the MLP network, surpassing all previous methods in all metrics (except for insignificant 0.2 mm difference in Root PE in Table 6.1). Moreover, our proposed AGRoL model significantly improves the smoothness of the generated motion, as reflected by the reduced *Jitter* error compared to other methods. We visualize some examples in Figure 6.3 and Figure 6.4. In Figure 6.3 we show the comparison of the reconstruction error between AGRoL and AvatarPoser. In Figure 6.4, by visualizing the pose trajectories, we demonstrate the comparison of the smoothness and foot contact quality between AGRoL and AvatarPoser.

6.3.6 Ablation Studies

In this section, we ablate our methods on AMASS dataset. We first compare our proposed MLP architecture with other backbones in the context of the diffusion model in Section 6.3.6 to highlight the superiority of our MLP network. Then we investigate the importance of time step embedding for our diffusion model and evaluate different strategies for adding the time step embedding in Section 6.3.6. Finally, we analyze the impact of the number of sampling steps used during inference in Section 6.3.6.

Network architecture. To validate the effectiveness of our proposed MLP backbone in the diffusion model setup, we conduct experiments where we replace our MLP network in AGRoL with other types of backbones and compare them. Specifically, we consider two alternative backbone architectures: the network from AvatarPoser [117] and the transformer network from Tevet et al. [244]. In transformer networks, instead of repetitively injecting the time positional embedding to every block, we concatenate the time positional embedding with the input features $\bar{x}^{1:N}$ and $\bar{p}^{1:N}$ before being fed to transformer layers. We apply the same technique to the AvatarPoser backbone as this model is also based on transformer layers. To ensure a fair comparison, we train AGRoL with two versions of AvatarPoser backbone. The first one, AGRoL-AvatarPoser, uses the architecture that follows exactly the same settings as described in the original paper [117], while the second one, AGRoL-AvatarPoser-Large, incorporates additional transformer layers to achieve a comparable size to our AGRoL model with MLP backbone. Similarly, we increase the number of layers in the transformer backbone [244] and train AGRoL-Transformer. As shown in Table 6.3, the AGRoL diffusion model with the proposed MLP backbone achieves superior results compared to the versions with other backbones.

Diffusion time step embedding. We study the importance of time step embedding. Time step embedding is often used in diffusion-based models [49, 294] to indicate the noise level t during the diffusion process. We use the sinusoidal positional embedding [254] as the time step embedding. Although the AGRoL without time step embedding (see Table 6.4) can still attain reasonable performance on metrics related to position errors and rotation errors, the performance on metrics related to velocity errors (*MPJVE* and *Jitter*) is severely degraded. This outcome is expected as the absence of the time step embedding implies that the model is not aware of the current denoising step, rendering it unable to denoise accurately.

We now ablate three strategies for utilizing the time step embedding in our network: *Add*, *Concat*, and *RepIn*. *RepIn* (Repetitive Injection) repetitively passes the time step embedding through a linear layer and injects the results into every block of the MLP network.

In contrast, *Add* and *Concat* inject the time step embedding only once at the beginning of the network. Before inputting it into the network, the time step embedding is passed through a fully connected layer and a SiLU activation to obtain a latent feature $u_t \in \mathbb{R}^{1\times D}$. *Add* sums the u_t with the input features $\bar{x}_t^{1:N}$ and $\bar{p}^{1:N}$, the output of the network then becomes $\hat{x}_0^{1:N} = \text{MLP}(\text{Concat}(\bar{x}_t^{1:N}, \bar{p}^{1:N}) + u_t)$, where vector u_t is broadcasted along the first dimension.*Concat* concatenates the u_t with the input features $\bar{x}_t^{1:N}$ and $\bar{p}^{1:N}$, resulting in $\hat{x}_0^{1:N} = \text{MLP}(\text{Concat}(\bar{x}_t^{1:N}, \bar{p}^{1:N}, u_t))$. ***RepIn*** is our proposed strategy for injecting the time step embedding. Specifically, for each block of the MLP network, we project the time step embedding separately using a fully connected layer and a SiLU activation, then we add the obtained features $u_{t,j}$ to the input features of the correspondent block, where $j \in [0, ..M]$ and M is the number of blocks. As shown in Table 6.4, our proposed strategy can largely improve the velocity-related metrics and alleviate the jittering issues and generate smooth motion.

Number of sampling steps during inference. We ablate the number of sampling steps that we used during inference. In Table 6.5, we take the AGRoL model trained with 1000 sampling steps and test it with a subset of diffusion steps during inference. We opted to use 5 DDIM [236] sampling steps as it enabled our model to achieve superior performance on most of the metrics while also being faster.

Sampling steps during training. In Table 6.8 we ablate the number of sampling steps T during training. Surprisingly, even when training with merely 10 sampling steps, the model can achieve decent performance. Although we notice that the model converges to a worse local minimum when only a few sampling steps is used. To achieve the best results, more sampling steps is required.

Additional losses. In addition to $\mathcal{L}_{dm}$, we explore three other geometric losses during the training like previous works [210, 231]:

$$\mathcal{L}_{pos} = \frac{1}{N}\sum_{i=1}^{N} \parallel \text{FK}(y_0^i) - \text{FK}(\hat{x}_0^i) \parallel_2^2 \tag{6.7}$$

$$\mathcal{L}_{vel} = \frac{1}{N-1} \sum_{i=1}^{N-1} \parallel (\text{FK}(y_0^{i+1}) - \text{FK}(y_0^i)) - (\text{FK}(\hat{x}_0^{i+1}) - \text{FK}(\hat{x}_0^i)) \parallel_2^2 \tag{6.8}$$

$$\mathcal{L}_{foot} = \frac{1}{N-1} \sum_{i=1}^{N-1} \parallel (\text{FK}(y_0^i) - \text{FK}(\hat{x}_0^i)) \cdot m_i \parallel_2^2, \tag{6.9}$$

where $\text{FK}(\cdot)$ is the forward kinematics function which takes local human joint rotations as input and outputs these joint positions in the global coordinate space. Here $y_0^{1:N} = x_0^{1:N}$ is the original data without noise. $\mathcal{L}_{pos}$ represents the position loss the of joints, $\mathcal{L}_{vel}$ represents the velocity loss of the joints in 3D space, and $\mathcal{L}_{foot}$ is the foot contact loss, which enforces static feet when there is no feet movement. $m_i \in \{0, 1\}$ denotes the binary mask and equals to 0 when the feet joints have zero velocity. We train our model with different combinations of extra losses, setting their weights equal to 1. As shown in Table 6.7. In contrast to previous works [117], the extra geometric losses do not bring additional performance to our diffusion model. Our model can achieve good results when trained solely with the denoising objective function $\mathcal{L}_{dm}$ from Eq. (4). We hypothesize that the lack of improvement in the performance of AGRoL with additional losses is due to the intricacies of the reverse diffusion process. This process may not synergize effectively with extra geometrical losses without appropriate adjustments.

Input/Output length. The proposed AGRoL model takes a sequence of sparse tracking signals as input and predicts the full body motion of the same length. In Table 6.3.6 we ablate the input & output length N of the AGRoL model. Our model benefits from longer input sequences, especially decreasing the mean per joint velocity error and jitter. But the performance saturates after the length of $N = 196$. In Table 6.10 we further compare our method with AvatarPoser [117] by varying its input length. Note that with longer input sequences our model can achieve significantly lower errors on velocity-related metrics like MPJVE and jitter, while AvatarPoser still has large MPJVE and jitter even with longer input length, thus failing to fully leverage the temporal information to generate smooth

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
w/o Time	2.68	**3.63**	22.80	1.36	**1.54**	**6.67**	**3.25**	15.23
Add	2.80	4.01	23.60	1.40	1.64	7.44	3.59	15.02
Concat	2.72	3.79	21.99	1.31	1.57	7.00	3.43	13.30
RepIn (Ours)	**2.66**	3.71	**18.59**	**1.31**	1.55	6.84	3.36	**7.26**

Table 6.4: Ablation of the time step embedding. *w/o Time* denotes the results of AGRoL without time step embedding. *Add* sums up the features from time step embedding with the input features. *Concat* concatenates the features from time step embedding with the input features. In *Add* and *Concat*, the time step embedding is only fed once at the top of the network. *RepIn* (Repetitive Injection) denotes our strategy to inject the time step embedding into every block of the network. The time step embedding mainly affects the *MPJVE* and *Jitter* metrics. Omiting the timestep embedding or adding it improperly results in high MPJVE and causes severe jittering issues.

motions.

Number of blocks in the MLP network. In Table 6.11 we evaluate the impact of varying the number of blocks (described in Sect. 3.2) in the MLP network. The performance of the model consistently improves as more blocks are added. However, the performance gains approach a plateau when more than 12 blocks are used.

Predicting noise. Our diffusion model AGRoL follows [223] and directly predicts the clean signal $\hat{x}_0^{1:N}$ in contrast to the original Denoising Diffusion Probabilistic Model (DDPM) formulation [100], where the model predicts residual noise $\epsilon_\theta(x_t, t)$ at every step. We further discuss the experiment presented in Table 6.12, where we implemented a version of AGRoL model ("AGRoL - pred noise") that predicts the residual noise $\epsilon_\theta(x_t, t)$. Similar to [223], we also find it better to predict the unnoised $\hat{x}_0^{1:N}$ directly, which is demonstrated by the results in Table 6.12. Since our simple MLP network can already produce reasonable estimations of the full body motion using only one forward pass, we hypothesize that the DDPM formulation of Ramesh et al. [223] allows to exploit the full capacity of the network *at every sampling step*, in contrast to the original formulation of [100].

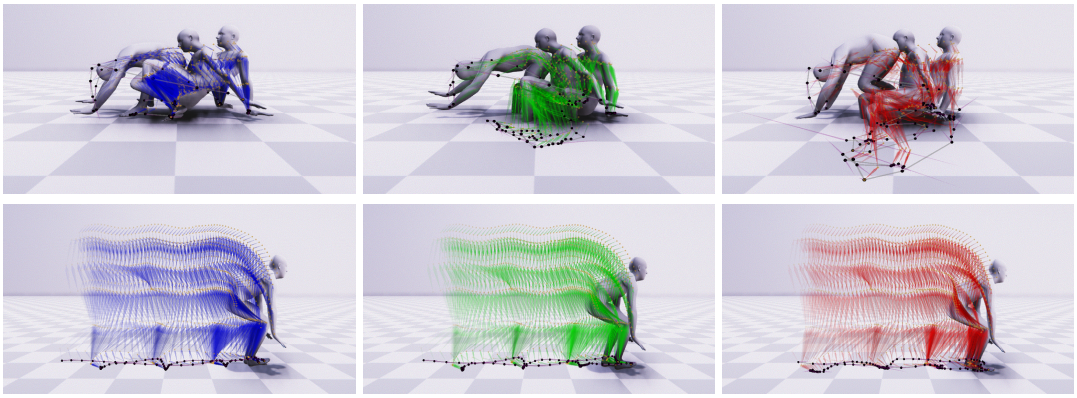


Figure 6.4: Motion trajectory visualization for predicted motions. **(a)** The ground truth motion with blue skeletons; **(b)** motion predicted by AGRoL with green skeletons; **(c)** motion predicted by AvatarPoser with red skeletons. The purple vectors denote the velocity vectors of the corresponding joints. Observing the motion trajectories, we can see jittering and foot sliding issues more clearly. Smooth motion typically exhibits regular pose trajectories with the velocity vector of each joint changing steadily. The density of joint trajectories varies with walking speed; trajectories become denser as the individual slows down. Therefore, in the absence of foot sliding, we should observe a significantly high density of points when a foot makes contact with the ground. The black dots in the bottom row represent the trajectories of the foot joints. We notice more pronounced spikes in the density of foot trajectories for AGRoL compared to AvatarPoser.

# Sampling Steps	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
2	3.17	4.93	20.03	2.19	2.12	8.98	4.61	**6.90**
5	**2.66**	3.71	**18.59**	**1.31**	**1.55**	6.84	3.36	7.26
10	2.68	**3.69**	19.55	1.39	**1.55**	**6.77**	**3.31**	7.51
100	2.84	3.93	23.50	1.62	1.67	7.19	3.51	9.64
1000	2.97	4.14	27.25	1.82	1.78	7.55	3.66	12.79

Table 6.5: Ablation of the number of DDIM [236] sampling steps during inference. The input and output length is fixed to $N = 196$. To achieve superior performance while being fast, we choose to use 5 sampling steps during inference.

6.3.7 Robustness to Tracking Loss

In this section, we study the robustness of our model against input joint tracking loss. In VR applications, it is common for the joint tracking signal to be lost on some frames when hands or controllers move out of the field of view, causing temporal discontinuity in the input signals. We evaluate the performance of all available methods on tracking loss by

Methods	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
AvatarPoser	5.69	10.34	572.58	8.98	5.49	17.34	8.83	762.79
MLP	5.37	10.76	107.82	12.43	6.48	16.94	8.74	92.51
Transformer	4.44	8.62	135.99	7.29	5.28	13.44	10.32	147.09
AGRoL (Ours)	**4.20**	**6.38**	**96.85**	**5.27**	**3.86**	**10.03**	**6.67**	**33.35**

Table 6.6: Robustness of the models to joints tracking loss. We evaluate the methods by randomly masking a portion (10%) of input frames during the inference on AMASS dataset. We test each method 5 times and take the average results. AGRoL achieves the best performance among all the methods, which shows the robustness of our method against joint tracking loss.

$\mathcal{L}_{pos}$	$\mathcal{L}_{vel}$	$\mathcal{L}_{foot}$	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
			2.66	**3.71**	**18.59**	**1.31**	**1.55**	**6.84**	**3.36**	**7.26**
		✓	2.83	4.07	20.66	1.58	1.70	7.49	3.66	9.20
✓			2.81	4.06	21.85	1.75	1.73	7.43	3.72	12.16
✓	✓		2.73	3.92	20.55	1.72	1.68	7.15	3.52	10.16
✓	✓	✓	2.89	4.16	20.58	1.73	1.76	7.63	3.81	8.98

Table 6.7: Ablation of the additional losses used during training.

# Sampling Steps	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Jitter
10	2.69	3.70	19.41	1.47	1.55	6.80	7.63
100	2.65	3.62	18.74	1.33	1.52	6.66	6.71
1000 (Ours)	2.66	3.71	18.59	1.31	1.55	6.84	7.26

Table 6.8: Ablation of the number of sampling steps during training the AGRoL model. The results become worse when the number of sampling steps is too small. More sampling steps is beneficial during training the network.

Input & Output Length	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Jitter
41	2.59	3.64	23.24	1.28	1.50	6.73	13.67
98	2.61	3.70	20.71	1.58	1.57	6.76	10.59
196 (Ours)	2.66	3.71	18.59	1.31	1.55	6.84	7.26
256	2.81	3.81	19.05	1.27	1.57	7.03	7.76

Table 6.9: Ablation of the input & output length of the AGRoL model. Our model can benefit from larger input length.

randomly masking 10% of input frames during inference, and present results in Table 6.6. We observe that the performance of all previous methods is significantly degraded, indi-

Methods	Input Length	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Jitter
AvatarPoser [117]	41	3.08	4.18	27.70	2.12	1.81	7.59	14.49
AGRoL	41	2.59	3.64	23.24	1.28	1.50	6.73	13.67
AvatarPoser [117]	196	3.05	4.20	28.71	1.61	1.70	7.82	16.96
AGRoL (Ours)	196	2.66	3.71	18.59	1.31	1.55	6.84	7.26

Table 6.10: Comparison between AGRoL and AvatarPoser [117] while varying the number of input frames. Our method can benefit from longer inputs and generate smoother motion. In contrast, AvatarPoser fails to gain consistent improvement from longer input sequences and even degrades in some metrics, including MPJVE, Lower PE, and Jitter.

#Blocks	#Params	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
2	2.08M	4.54	7.39	34.38	3.10	3.13	13.55	7.28	22.03
6	4.35M	2.89	4.12	19.51	1.38	1.69	7.62	3.72	6.29
12 (Ours)	7.48M	**2.66**	3.71	18.59	1.31	1.55	6.84	3.36	7.26
24	14.53M	2.73	**3.61**	**18.33**	**1.08**	**1.50**	**6.65**	**3.28**	**7.23**

Table 6.11: Ablation study of the number of blocks in the proposed MLP network. The performance of the AGRoL model benefits keeps improving with more blocks and reaches a plateau when the number of blocks reaches 12.

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
AGRoL - pred noise ϵ_θ	5.41	8.88	28.67	4.38	3.91	16.06	8.76	9.80
AGRoL (Ours)	**2.66**	**3.71**	**18.59**	**1.31**	**1.55**	**6.84**	**3.36**	**7.26**

Table 6.12: Ablating different formulations of the diffusion model: Predicting clean signal directly (Ours) *vs* predicting noise $\epsilon_\theta(x_t, t)$. The AGRoL model that learns to predict clean body motion at every diffusion step is substantially better in every metric.

cating their lack of robustness against tracking loss. In comparison, AGRoL shows less degradation in accuracy, suggesting that our approach can accurately model motion even with highly sparse tracking inputs.

6.3.8 Inference Speed.

Our AGRoL model achieves real-time inference speed due to a lightweight architecture combined with DDIM sampling. A single AGRoL generation, that runs 5 DDIM sampling steps, produces 196 output frames in 35 ms on a single NVIDIA V100 GPU. Our predictive MLP model takes 196 frames as input and predicts a final result of 196 frames

in a single forward pass. It is even faster and requires only 6 ms on a single NVIDIA V100 GPU. We first show extra ablation experiments of our method on AMASS [174] dataset following the protocol proposed in [117]. Then we show extra qualitative results and comparison between our method and the state-of-the-art method [117].

6.3.9 Extra Datasets

In addition to the AMASS [174] dataset, we also evaluate the performance of our approach on AIST++ [150] dataset. AIST++ dataset contains in total 5.1 hours of dancing movements performed by professional dancers. The dataset has 10 genres of dances, including some dances containing complicated movements like breakdancing, jazz etc. We follow the train/test splits proposed in [150]. The global rotation and translation of the hands and head are calculated using the SMPL human model [163] with the provided model parameters. Compared to the AMASS dataset, which contains mostly everyday life motions, the motions in the AIST++ dataset are much more diverse and challenging. As shown in Table 6.13, the AGRoL achieves superior performance in all the metrics and produces smoother motions compared to the AvatarPoser and the predictive MLP model. While there is still room for improvement on such a challenging dataset, the proposed AGRoL method significantly reduces the MPJVE, Jitter and lower body positional error (Lower PE) compared to the AvatarPoser.

Method	MPJRE	MPJPE	MPJVE	Hand PE	Upper PE	Lower PE	Root PE	Jitter
AvatarPoser	4.37	9.11	97.24	4.31	3.32	17.47	8.11	65.18
MLP (Ours)	3.63	7.33	74.90	3.86	2.69	14.03	5.48	47.16
AGRoL (Ours)	**3.56**	**6.83**	**65.58**	**2.17**	**2.04**	**13.74**	**4.91**	**41.95**
GT	0	0	0	0	0	0	0	30.48

Table 6.13: Comparison of our approach with the competitors on AIST++ [150] dataset. The AGRoL performs better than other methods.

6.4 CONCLUSION

In this chapter, we presented a simple yet efficient MLP-based architecture with carefully designed building blocks which achieves competitive performance on the full-body motion synthesis task. Then we introduced AGRoL, a conditional diffusion model for full-body motion synthesis based on sparse tracking signal. AGRoL leverages a simple yet efficient conditioning scheme for structured human motion data. We demonstrated that our lightweight diffusion-based model generates realistic and smooth human motions while achieving real-time inference speed, making it suitable for online AR/VR applications. A notable limitation of our and related approaches is occasional floor penetration artifacts. Future work involves investigating this issue and integrating additional physical constraints into the model.

PART III:

ROBOTS LEARN FROM HUMAN

In this part of the thesis, we tackle the challenge of transferring human motion to robot movements, specifically, we focus on the task of converting human grasp demonstration to any multi-fingered grippers. In Chapter 7, we propose a novel optimization-based approach, which produces robotic grasps that mimic the human hand orientation and the contact area with the object, while alleviating interpenetration. Extensive experiments show that our method leads to grasps more similar to the human demonstration than existing approaches, without requiring any gripper-specific tuning. We confirm these findings through a user study and validate the applicability of our approach on a real robot.

CHAPTER 7

GRASPING LIKE HUMANS

Beyond simply understanding human motion, our objective extends to enabling robots to learn directly from human demonstrations. This step brings us closer to creating intelligent machines that can observe, understand, and replicate complex human actions, thereby improving their functionality and integration into human-centric environments.

7.1 INTRODUCTION

To effectively aid individuals with their daily tasks, a robot must possess the ability to manipulate objects in ways that are specific to the task at hand. This involves handling objects differently depending on the current objective. For instance, a robot should not handle a knife in the same manner when cutting vegetables as it would when passing it to someone with limited mobility. A robot can learn such specificities by being taught by a human through task demonstrations. In this study, we concentrate on the task of grasping rigid objects, as depicted in the Figure 7.1. Reproducing exactly a human grasp is impossible for a robot, because a robotic gripper is usually quite different from a human hand: different size, number of fingers, actuation, *etc.* (see Figure 7.2 for a comparison).

Rather than simply handling objects in a predetermined way, the robot should be capable of grasping objects in a manner that emulates that of a human. However, most existing grasp retargeting techniques [90, 216, 138] rely on manually crafted correspondences between the robot gripper and the human hand. These correspondences are usually based on joint angles, surface contact points, or other key vectors, and they do not take the object being grasped into account. Other methods, such as ContactGrasp [18] refine the grasps produced by GraspIt! [189] through contact surface optimization and reranking. However, this approach is time-consuming and requires a significant amount of effort to generate and refine grasp candidates. Additionally, it may result in significant differences from the human grasp.

Defining the concept of "grasping like a human" is a challenging task, as it lacks a precise and well-defined meaning. However, in this research, we present some general indicators of grasp similarity, namely the *contact* surface and the grasp *orientation*. In-

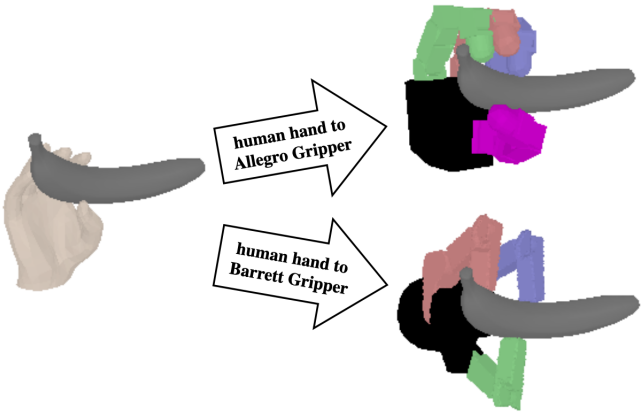


Figure 7.1: Given an input human grasp (left), our method outputs a configuration of a multi-fingered gripper grasping the same object *like the human* demonstration. We experiment with the Allegro (top) and BarrettHand (bottom) grippers.

deed, the affordance [75] of a grasped object is typically dependent on the open space surrounding this object. In general, replicating a human grasp of an object by gripping it from a similar orientation and using a similar contact surface on the object should allow the robot to perform similar actions with the object as the human did. In this study, we examine the effectiveness of utilizing these grasp similarity indicators in generating robot grasps that closely resemble human demonstrations.

Our research proposes a multi-step optimization-based approach that utilizes a human grasp demonstration, represented by a 3D mesh of the object and a parametric MANO model [227] of the hand pose, to generate a corresponding robotic grasp configuration. We formulate an objective function that promotes similarity in contact surfaces and global orientation between the human and robotic grasps, while penalizing collisions between the gripper and the object. To prevent getting stuck in local minima, we perform a multi-stage optimization process where the global position and orientation of the gripper are initialized to match those of the human demonstration. Next, the fingers are closed by minimizing the distance between the fingertips and the contact areas on the objects, followed by optimizing for the complete objective function in the final step. To validate the genericity of our approach, we experiment with two off-the-shelf robotic hands: the *Allegro* [1] and the *BarrettHand* [2] grippers (see Figure 7.2).

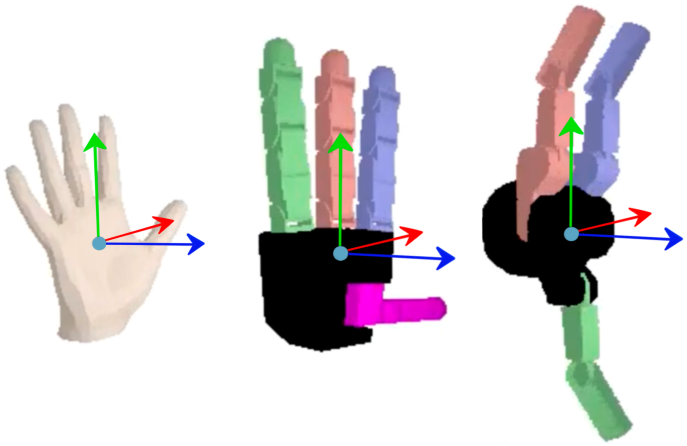


Figure 7.2: Comparison between different grippers at the same scale with a human hand (left), Allegro (middle) and BarrettHand (right). Note that the size of the gripper and in particular the fingers are significantly different. The blue vector represents the normal vectors of the human hand and robot hands, the green vector represents the forward vector (best seen in color).

We evaluate our method using human grasps from the YCB-affordance dataset [41] with various quality metrics. In addition to the proposed optimization-based method, we conducted a user study to qualitatively compare our approach with related techniques. The results from both evaluations demonstrate that our approach generates reasonable grasps that are superior to those produced by existing state-of-the-art grasp retargeting methods, and are more similar to the human demonstration. Ultimately, we validate the practicality of our approach by applying it to a Panda robotic arm in real-world scenarios.

In summary, the main contributions of this work are: (1) A novel objective function consisting of four losses which encourages a valid grasp while capturing the similarity between the human hand grasp and robot gripper grasp. (2) A novel multi-step optimization-based pipeline to transfer a human grasp demonstration to any multi-fingered gripper. (3) An extensive quantitative and qualitative evaluation and comparison between our approach and other related methods.

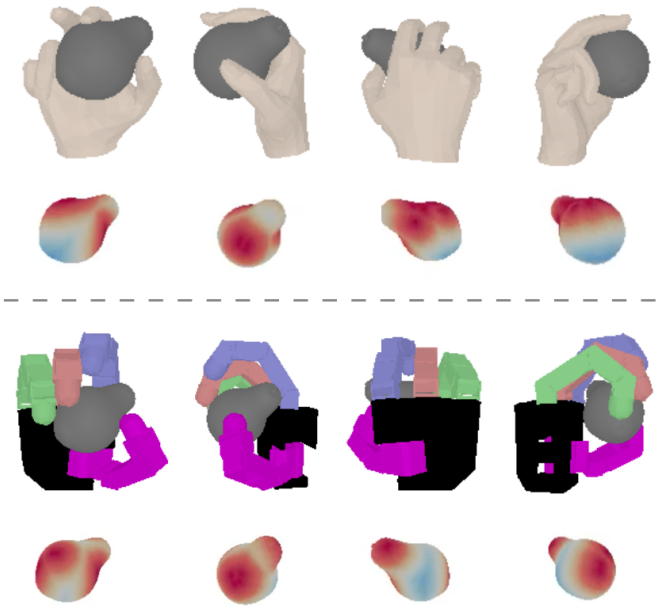


Figure 7.3: Contact heatmaps on the object mesh corresponding to a human (top) and robotic (bottom) grasp. Our optimization-based approach tries to minimize the discrepancy between these heatmaps. Red color denotes regions close to the hand/gripper while blue color denotes regions far from the hand/gripper.

7.2 METHOD

In this section, we describe our optimization-based approach to generate a robot grasp 'similar' to a given human grasp. After formalizing the problem and notations (Section 7.2.1), we introduce the optimized objective function in Section 7.2.2 and detail all the steps of our approach in Section 7.2.3.

7.2.1 PROBLEM AND NOTATIONS

We consider as input a rigid object grasped by a human hand. We represent the object by a 3D mesh $\mathcal{M}_{object}$, and we adopt the MANO [227] model to represent the pose of the hand by a global rigid transformation $(R_{hand}, t_{hand}) \in SO(3) \times \mathbb{R}^3$ relative to the object and by its local joints configuration $\theta_{hand} \in SO(3)^{20}$. Similarly, we assume that a kinematic model of the robotic gripper is available. We aim to predict a global pose $(R_{robot}, t_{robot}) \in SO(3) \times \mathbb{R}^3$ relative to the object and some joints configuration $\theta_{robot} \in \mathbb{R}^n$ describing a

static grasp with this gripper similar to the human demonstration ($n = 16$ for Allegro, $n = 7$ for Barrett). We formulate this as an optimization problem and minimize an objective function $\mathcal{L}(R_{robot}, t_{robot}, \theta_{robot})$ representing the *dissimilarity* of the robotic grasp with the human demonstration.

7.2.2 Objective Function

Our objective function $\mathcal{L}$ is composed of a contact-heatmap loss $\mathcal{L}_C$ that incites contacts on the object to be similar, a hand orientation loss $\mathcal{L}_O$, as well as losses that penalize interpenetration with the object $\mathcal{L}_I$ and self-penetration of the gripper $\mathcal{L}_S$, *i.e.*:

$$\mathcal{L} = \lambda_C \mathcal{L}_C + \lambda_O \mathcal{L}_O + \lambda_I \mathcal{L}_I + \lambda_S \mathcal{L}_S \tag{7.1}$$

with weights experimentally set to $\lambda_C = 10, \lambda_O = 10, \lambda_I = 0.5$ and $\lambda_S = 1$. We detail these terms in the following paragraphs.

Contact heatmap loss $\mathcal{L}_C$. Intuitively, grasps are similar if their contact regions on the target object are similar. Based on this observation, we propose an object-centric contact heatmap loss term, which encourages the contact regions of the input human hand and the robotic gripper on the object to be similar. Specifically, we represent the contact regions of the human hand and the robot gripper by scalar contact heatmaps H on the object. At each vertex $o_i \in \mathcal{M}_{object}$ of the object mesh, we define the values of these heatmaps as

$$\begin{cases} H^{hand}(o_i) = \exp(-d(o_i, \mathcal{M}_{hand})/\tau) \\ H^{robot}(o_i) = \exp(-d(o_i, \mathcal{M}_{robot})/\tau) \end{cases} \tag{7.2}$$

where $d(o_i, \mathcal{M})$ denotes the L_2-distance of o_i to the set of vertices of the mesh $\mathcal{M}$, and where τ is a constant used to define contacts in a soft manner (*i.e.* $H(o_i) = 1$ when $d(o_i, \mathcal{M}) = 0$, and $H(o_i) \approx 0$ when $d(o_i, \mathcal{M}) >> \tau$). In our experiments, we use uniformly sampled meshes and choose $\tau = 0.01m$. Figure 7.3 shows examples of contact heatmaps for different human grasps. We define our object-centric contact heatmap loss

as the L_1-distance between these generated heatmaps:

$$\mathcal{L}_C = \sum_{o_i \in \mathcal{M}_{object}} |H^{hand}(o_i) - H^{robot}(o_i)| \tag{7.3}$$

Hand orientation loss $\mathcal{L}_O$. Grasps have high similarity if the hands are oriented similarly towards the object, thus resulting in a similar free space around the object, and thus potentially to a similar affordance. Therefore, we introduce a loss to encourage the orientation of the hand and the gripper to be similar. To this end, for each human/robot hand model, we define two unit vectors which are inherent to the model, the forward vector f and the normal vector n. Examples of these two vectors for different models are shown in Figure 7.2. The normal vector n is defined as the unit normal vector of the palm surface. The forward vector f is defined as the unit vector that is parallel to the palm surface and pointing to the 'pushing' direction. We define the hand orientation loss as the L_1-distance between these two unit vectors:

$$\mathcal{L}_O = |n_{robot} - n_{hand}| + |f_{robot} - f_{hand}| \tag{7.4}$$

Gripper-object interpenetration loss $\mathcal{L}_I$. To avoid interpenetration while ensuring realistic contacts between the robotic gripper and the object, we take inspiration from Müller *et al.* [191] and add a loss

$$\mathcal{L}_I = \alpha_1 \mathcal{L}_{push} + \beta_1 \mathcal{L}_{pull} + \gamma_1 \mathcal{L}_{normal} \tag{7.5}$$

to our objective function. It consists of three weighted terms. The first term $\mathcal{L}_{push}$ aims at avoiding interpenetration by pushing the penetrated parts of the robotic gripper towards their nearest surface on the object mesh. To do so, we consider $\mathcal{U}_{robot} \subset \mathcal{M}_{robot}$ the set of vertices on the robotic gripper mesh that are inside the object mesh, and $\mathcal{U}_{object} \subset \mathcal{M}_{object}$ the set of vertices on the object mesh that are inside the robotic gripper mesh. In practice, we detect these two sets of vertices using the generalized winding numbers [109]. We define

$$\mathcal{L}_{push} = \sum_{o_i \in \mathcal{U}_{object}} \tanh\left(\frac{d(o_i, \mathcal{M}_{robot})}{\alpha_2}\right) + \sum_{r_k \in \mathcal{U}_{robot}} \tanh\left(\frac{d(r_k, \mathcal{M}_{object})}{\alpha_2}\right) \tag{7.6}$$

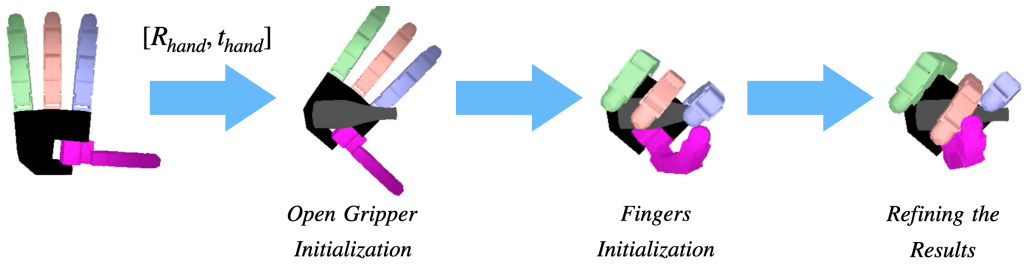


Figure 7.4: Overview of our pipeline for transferring human hand grasp to robot gripper grasp. We first initialize the gripper with open fingers at the location of the hand. We then initialize the fingers position on the object surface by minimizing the distance between the fingertips and the contact regions of the human demonstration. At last, we refine the grasp by minimizing the overall objective function.

to penalize interpenetration. The second term $\mathcal{L}_{pull}$ encourages contacts for points of the gripper closer than a threshold δ to the object, while being constant for points farther away:

$$\mathcal{L}_{pull} = \sum_{r_k \in \mathcal{M}_{robot}} \tanh\left(\frac{\min(d(r_k, \mathcal{M}_{object}), \delta)}{\beta_2}\right) \tag{7.7}$$

We use $\delta = 2mm$ in practice. To further ensure realistic contacts, a third term is added that encourages normals of both meshes to be opposite at contact locations $\mathcal{V} = \{r_k \in \mathcal{M}_{robot} | d(r_k, \mathcal{M}_{object}) < \delta\}$:

$$\mathcal{L}_{normal} = \sum_{r_k \in \mathcal{V}} 1 + \langle N(r_k), N(o_i^k) \rangle \tag{7.8}$$

where $N(\cdot)$ denotes the unit normal vector at a given vertex, and $o_i^k = arg\,min_{o_i \in \mathcal{M}_{object}} d(r_k, o_i)$ denotes the closest point on the object for any vertex $r_k \in \mathcal{V}$. Hyperparameters values are experimentally set to $\alpha_1 = 2.4$, $\beta_1 = 7$, $\gamma_1 = 0.001$, $\alpha_2 = 4cm$, $\beta_2 = 6cm$.

Gripper self-penetration loss $\mathcal{L}_S$. $\mathcal{L}_I$ considers griper-object penetration, but some configurations of the gripper could also lead to self-penetration between different gripper components such as its fingers. We thus add a loss to avoid self-penetration. To this end, we use the exact same loss as $\mathcal{L}_{push}$ but apply it between the gripper mesh and itself, resulting in a loss $\mathcal{L}_S$.

7.2.3 Optimization Pipeline

Our objective function of Equation (7.1) admits many local minima, and several optimization terms admit zero gradient when the gripper is far from the object. Having a good initialization is therefore important, and we thus propose a multi-step optimization pipeline whose overview is shown in Figure 7.4. It consists of 3 steps: (a) initializing the robotic gripper with open fingers around the same location as the human hand, (b) closing the fingers until contact with object, (c) refining all degrees of freedom.

(i) Open gripper initialization. Because of the hand orientation loss $\mathcal{L}_O$, the optimal global position and orientation of the gripper (R_{robot}, t_{robot}) is likely to be close to the global position and orientation of the human hand (R_{hand}, t_{hand}) in the object coordinate system. This is why we initialize the gripper position and orientation at the same position and orientation as the human hand. At this stage, we assume that the rest of the parameters, *i.e.*, the angle of the finger joints correspond to a fully-open position and we thus refer to this stage as 'open gripper initialization'.

(ii) Fingers initialization. To initialize the fingers and make the fingers touch the object at the right place, we first detect the contact region of the human grasp, then we minimize the distance between the fingertips and their nearest contact region using the gripper-object interpenetration loss $\mathcal{L}_I$ defined in Equation (7.5) with the self-penetration loss $\mathcal{L}_S$. In this way, we can put the fingers of the robot hand to their closest region of contact and at the same time avoid gripper-object interpenetration and self-penetration.

(iii) Refining the results. We finally run the full optimization from this initialization. We use AdamW [165] as our optimizer, the initial learning rate is set to 0.001 for the translation T_{robot} and 0.01 for rotation R_{robot} and pose parameters θ_{robot}. Each grasp is optimized for 100 iterations. The learning rate decreases by 10 at iteration #50. During the optimization, we use the rotation parametrization introduced in [301].

		Grasp ϵ-quality ↑	Max Penetration Depth (cm)↓	Penetration Volume (cm^3)↓	Orientation Difference ↓	Contact Heatmap Difference ↓
Allegro Hand	DexPilot† [90]	**0.535**	2.91	5.04	0.011	0.176
	ContactGrasp† [18]	0.460	3.53	6.94	1.818	0.195
	GraspIt! (best $\mathcal{L}_C + \mathcal{L}_O$) [189]	0.345	2.76	**1.27**	0.420	0.254
	Ours	0.466	**2.57**	4.89	**0.001**	**0.153**
Barrett Hand	ContactGrasp† [18]	0.523	4.65	6.28	2.003	0.225
	GraspIt! (best $\mathcal{L}_C + \mathcal{L}_O$) [189]	0.354	4.52	**0.88**	0.714	0.258
	Ours	**0.566**	**4.09**	2.91	**0.001**	**0.166**

Table 7.1: Comparison of our approach with state-of-the-art methods. † indicates methods that use different hyperparameters for different grippers. For GraspIt!, we generated 100 grasps per human demonstration and selected the one with the lowest orientation difference and contact heatmap difference, *i.e.*, the lowest $\mathcal{L}_C + \mathcal{L}_O$ loss.

7.3 Experiments

In this section, after presenting datasets and metrics (Section 7.3.1), we provide the results of an ablation study in Section 7.3.3 and a comparison to the state of the art in Section 7.3.4. We then describe a user study (Section 7.3.6) that validates that our approach leads to grasps more similar to the human demonstrations than the state of the art.

7.3.1 Datasets

To measure performance, we consider the human grasps from the YCB-Affordance dataset [41]. For the 52 objects of the YCB-Objects [27], different types of human grasps are manually annotated and refined using GraspIt! [189]. This leads to a diversity in terms of grasps, including not only power grasps but also pinch grasps, *etc.*, as shown by the annotations of the grasp categories provided with the dataset [41] and illustrated in the top row of Figure 7.5.

7.3.2 Metrics

For evaluation, we measure both the grasps quality using the Grasp ϵ-quality metric, which corresponds to the radius of the largest ball centered at the origin which can

be enclosed by the convex hull of the wrench space [188], as commonly used in, *e.g.*, [66, 170, 27]. We also report the *Max Penetration Depth*, *i.e.*, the maximum distance between a vertex of the gripper that is inside the object and its closest vertex on the object, and the *Penetration Volume*, *i.e.*, the estimated volume of penetration between the two meshes. In addition to these grasp quality metrics, we propose two metrics that are used to measure the similarity between the human hand grasp and the robot gripper grasp: the *Contact Heatmap Difference* that measures contact similarity and the *Orientation Difference* that measures similarity in terms of contact angle. We use the metric $\mathcal{L}_C$ introduced in Section 7.2.2 to evaluate numerically the *Contact Heatmap Difference* and $\mathcal{L}_O$ for the *Orientation Difference*.

7.3.3 Ablations

We first ablate our approach in Table 7.2 for the Allegro gripper. To start with, we replace the second step of our optimization pipeline by another finger closing strategy inspired by [41]: starting from an *open* configuration of the gripper, we discretize the gripper configuration space and pick iteratively for each finger joint the bin corresponding to the most *closed* configuration that does not penetrate the object. We observe that this significantly degrades the grasp ϵ-quality, the penetration volume and the contact heatmap similarity.

We then ablate the losses of the final optimization step of our approach by removing them one by one. Removing the contact similarity loss $\mathcal{L}_C$ significantly degrades the contact heatmap difference from 0.153 to 0.189, while also impacting negatively the grasp ϵ-quality metric and the interpenetration. Additionally removing the loss on the angle similarity $\mathcal{L}_O$ leads to grasps that are even less similar and leads to higher penetration volume. Furthermore, removing the self-penetration loss $\mathcal{L}_I$ degrades the grasp ϵ-quality even more. We finally evaluate the performance of our approach without the global optimization (the third step of our pipeline) in the last row, to show its importance for achieving good grasps.

Fingers Init. (step 2)	$\mathcal{L}_I$	$\mathcal{L}_S$	$\mathcal{L}_O$	$\mathcal{L}_C$	Grasp ϵ-quality ↑	Max Penetration Depth (cm)↓	Penetration Volume (cm^3)↓	Orientation Difference ↓	Contact Heatmap Difference ↓
Discrete init.	✓	✓	✓	✓	0.294	2.59	5.83	0.011	0.213
Contact optim.	✓	✓	✓	✓	**0.466**	2.57	**4.89**	**0.001**	**0.153**
	✓	✓	✓		0.438	2.70	6.32	**0.001**	0.189
	✓	✓			0.467	**2.55**	6.60	0.041	0.190
	✓				0.411	2.69	6.39	0.031	0.187
					0.387	2.96	8.52	0.011	0.170

Table 7.2: Ablation study of our approach with the Allegro gripper. In the first row, we replace the second step of our pipeline with contact optimization for the fingers initialization by a discrete closing strategy. In the rows below, we remove the losses one by one. The last row without any loss corresponds to the absence of Step 3 of our optimization pipeline.

7.3.4 Comparison with Other Methods

We compare our approach to the state of the art and report performances for the Allegro gripper as well as for the BarrettHand gripper in Table 7.1. We also show various examples in Figure 7.5. As a first approach, we compare to a manually-defined mapping between the human hand and the Allegro gripper using a re-implementation of DexPilot [90]. Note that this approach would require new manual annotations for another type of robotic gripper. As a second approach, we use ContactGrasp [18] that proposes to refine and rerank the grasps generated by GraspIt! by exploiting contact information. For each object, GraspIt! generates grasps from different directions around the object (we consider 100 grasps in practice). The generated grasps are fed to the ContactGrasp pipeline with the contact region heatmaps generated using the code provided by the authors. In the end, we consider the best-ranked robotic grasp for each reference human grasp. As a third approach, we use GraspIt! [189] to generate 100 different grasps for an object and select the one that minimizes the sum of *Orientation Difference* and *Contact Heatmap Difference*. While DexPilot obtains a higher ϵ-quality metric than our approach with the Allegro gripper, the grasps are actually less realistic as there are larger interpenetrations with the object, as illustrated in the left example of Figure 7.5. Additionally, the grasp similarity in terms of both orientation and contact heatmap is lower than our approach. Note also that DexPilot is not generic in that it has been handcrafted for the

Allegro gripper. We also compare our method to the best grasp from ContactGrasp [18], which uses different hyperparameters for the two grippers. We observe that our approach leads to higher quality grasps, with less interpenetrations, and with higher similarity with the input human grasps. This is particularly true for the orientation similarity as illustrated in the examples of Figure 7.5: while ContactGrasp optimizes for similar contact regions, it does not enforce the gripper to approach the object from a similar direction, which can lead to grasps with significantly different properties than the human grasps in terms of free space around the object. Lastly, our method outperforms GraspIt! on every metric except for the penetration volume.

7.3.5 Running time

We evaluated all the methods on a machine with 20 Intel(R) Core(TM) i9-9900X CPUs and one NVidia GeForce RTX 2080Ti card. Our approach takes about 1 minute for a given grasp. For comparison, our implementation of DexPilot takes about 1 second but does not consider the geometry of the object, and ContactGrasp takes on average 43 minutes for one input human grasp as it first requires to generate 100 robotic gripper grasps using GraspIt! before refining all of them.

7.3.6 User Study

We aim at enabling robots to *grasp like humans*, but the metrics above do not necessarily express this notion well. We therefore conducted a user study to better evaluate the similarity between human and robotic grasps. It is difficult for people to quantitatively evaluate this similarity, thus we resorted to a comparative evaluation. We randomly selected 120 human grasps from the YCB-Affordance [41] dataset. For each human grasp, we generated corresponding robotic grasps using different methods and asked participants to select the one which – in their opinion – is the most similar to the human demonstration. For the Allegro gripper, we compared our method with ContactGrasp [18] and DexPilot [90]. For BarrettHand, we compared our method with ContactGrasp [18]. We also included an additional grasp generated randomly using GraspIt! [189] as baseline.

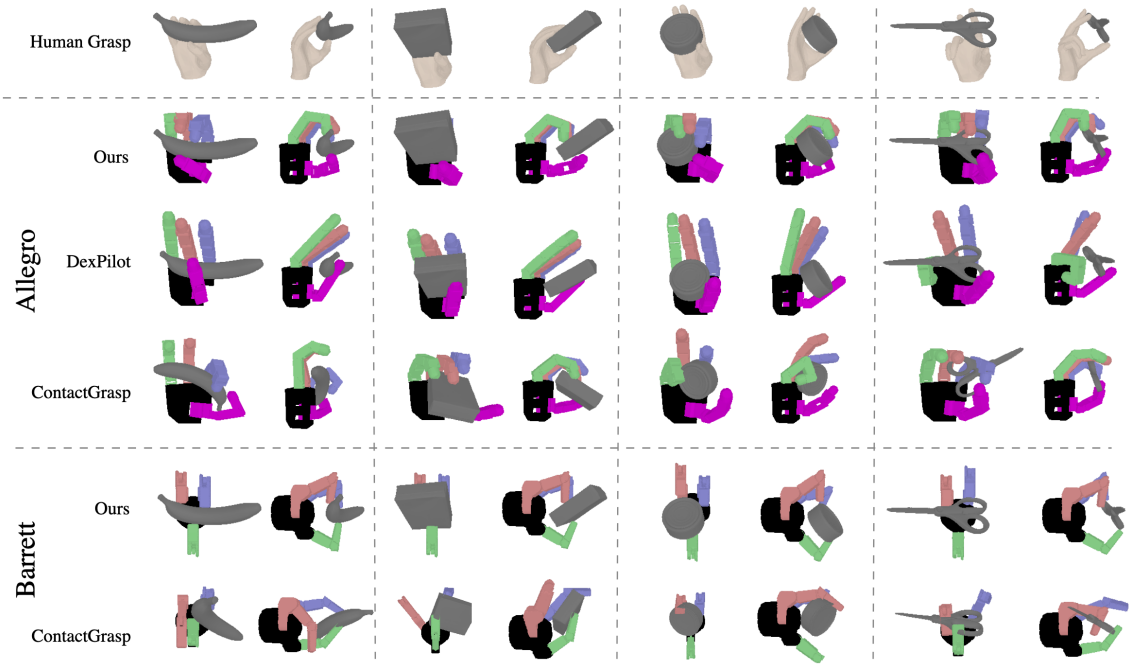


Figure 7.5: Example of generated grasp transfers for our approach, DexPilot[90] and ContactGrasp[18] for Allgro and BarrettHand grippers, our approach generated more plausible robot grasps with less penetration.

We received in total 1,392 votes from 58 participants – each participant sharing its preference regarding 24 human grasps. Results are summarized in Table 7.3. Overall, the participants favored grasps produced by our method in 51% of the cases for the Allegro gripper, and 73% of the cases for BarrettHand. These scores are way above random chance (25% for Allegro, 33% for BarrettHand), and they suggest that our generated grasps are considered significantly more similar to the human demonstrations than the grasps generated using the other evaluated methods. Further analysis of the results showed that the preference for our method could be explained in all cases by the smaller difference of global orientation between the robot and human hands when using our method.

7.3.7 Real World Experiments

We focus in this work on predicting static grasps that describe the pose and joints configuration of a robotic gripper with respect to an object. Grasping however is fundamentally

Grasp generation method	Number of votes (total: 1392)	
	Allegro	BarrettHand
GraspIt! (random) [189]	49	38
ContactGrasp [18]	117	101
DexPilot [90]	265	-
Ours	**440**	**383**

Table 7.3: User-study: "Which grasp is the most similar to the human demonstration?"

a dynamic process, involving robot motion and contact forces. To demonstrate the usability of our approach in real scenarios, we performed grasping experiments using an *Allegro* gripper mounted on a *Panda* robotic arm, from *Franka Emika*. A few examples are shown in Figure 7.6. These experiments allowed to check that grasps produced by our method are physically feasible, while being similar to the human demonstrations. Robot perception is out of the scope of this study, therefore we used as input some human demonstrations from the YCB-Affordance [27] dataset, and we manually placed the objects in known poses before attempting the grasps with the robot. We did not conduct any quantitative evaluations because of this manual step. Note that state-of-the-art methods for object [136] and hand+object pose estimation [92, 89, 12, 30, 241, 91] could be used to overcome this limitation.

7.4 Conclusion

We propose a multi-step optimization-based approach for transferring grasps from a human demonstration to a multi-fingered robotic gripper, so as to enable a robot to *grasp like a human*. The proposed approach is generic and can be applied to arbitrary multi-fingered gripper, as shown by our experimental evaluation with both Allegro and BarrettHand grippers. Our results – based on quantitative metrics and a qualitative user study – suggest that it produces grasps significantly more similar to the human demonstrations than state-of-the-art methods, and we validated its applicability in the real world using an Allegro gripper mounted on a Panda arm.

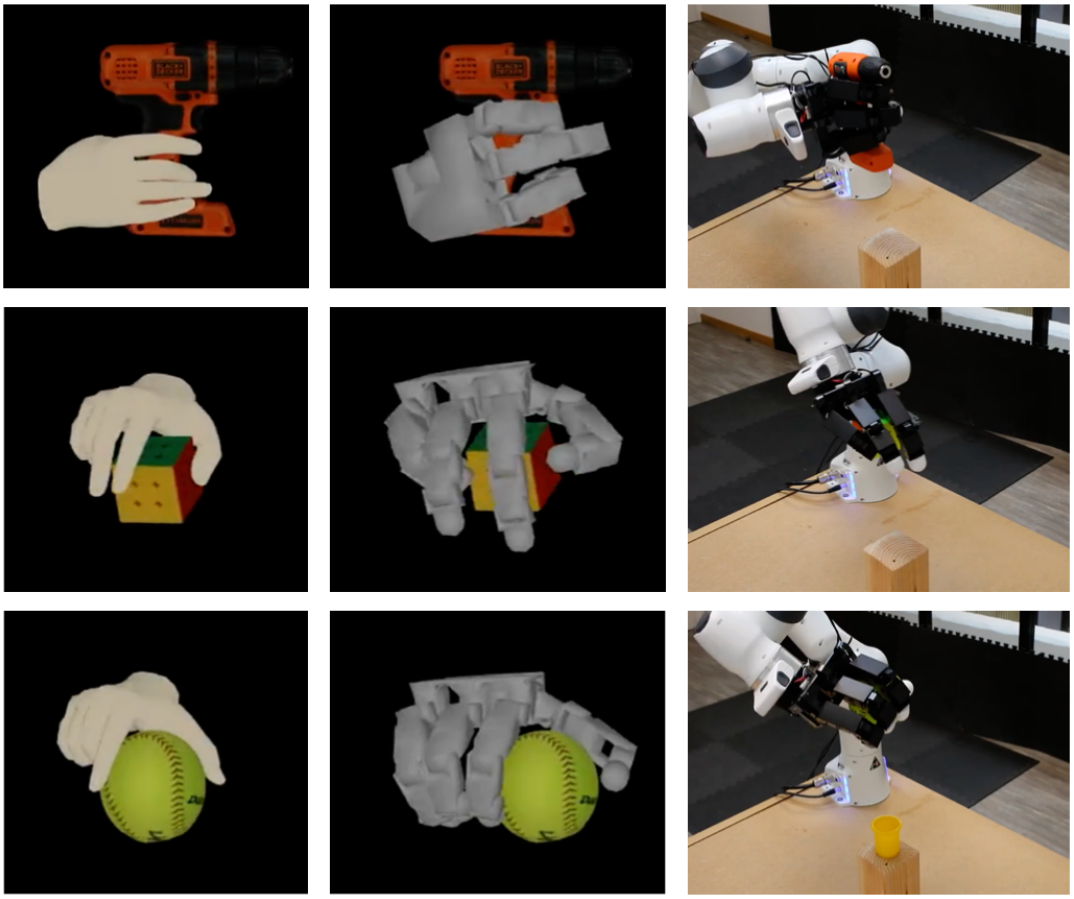


Figure 7.6: Real world examples. We performed grasping experiments using an *Allegro* gripper mounted on a *Panda* robotic arm, from *Franka Emika*. The left column shows the given human static grasp, the middle column shows the robot gripper grasp retargeted by our method. The right column shows the success grasp on the real world robot arm. *Real world experiments carried out by Dr.Romain Brégier at Naver Labs Europe.*

CHAPTER 8

CONCLUSION

8.1 SUMMARY

This thesis focuses on the development of methodologies that facilitate seamless interaction between robots and humans in complex environments. The main contributions are threefold. First, we present two innovative strategies to boost a robot's scene understanding capabilities. One approach enhances depth accuracy, while the other establishes a framework for more precise and autonomous object segmentation in open-world environments. Both methods are adaptable and can generalize to novel domains without the need for labeled data. Second, we introduce two novel techniques aimed at improving the ability of the robots to understand and predict human motion. Our approach accurately forecasts full-body human movements, regardless of whether the body joint observations are partial or complete. By utilizing these methods, robots can enhance their predictive capabilities, leading to more effective collaboration with humans. Lastly, we address the challenge of translating learned human motions into robotic movements. We propose a method that adapts human grasp demonstrations for compatibility with any multi-fingered gripper, allowing robots to manipulate objects more intuitively and efficiently.

8.2 Future Work

Several lines of future works could be explored. In this section, we discuss some potential future directions.

8.2.1 Object Discovery in 3D Data

Humans, as inhabitants of a three-dimensional world, are inherently adapted to recognize and interact with objects in 3D spaces. However, the focus of most object discovery methods has been primarily oriented towards 2D images or videos. There is a significant opportunity to broaden the scope of real-world object discovery applications by enabling robots to interpret 3D cues and directly discover objects in three-dimensional spaces. Such an advancement would not only have significant implications for a variety of applications, but it could also greatly expedite the process of automatic labeling. Nonetheless, the processing of 3D data presents a considerable challenge. The volume of 3D data is typically much larger than that of 2D data, and the efficacy of algorithms often depends heavily on the chosen method of 3D representation. Thus, an area of critical importance lies in the development of a compact and efficient system of representation for 3D data. Achieving this goal would be of significant benefit to the broader community of researchers and practitioners in this field.

8.2.2 Interactive Human Motion Understanding

Many current approaches to understanding human motion are predominantly limited to scenarios involving a single individual. However, in daily life, interactions play a pivotal role in understanding human motion. Some recent studies have addressed this aspect of interactive human motion understanding. Nevertheless, their methodologies often necessitate a fixed number of individuals or require exhaustive exploration of pairwise relationships between every pair of individuals in a scene. Thus, it is crucial that we envision a method capable of considering all individuals in a scene as a unified entity. This approach would more accurately reflect real-world situations, where human interactions are not just

pairwise, but often involve dynamic groups of varying sizes. It could potentially lead to a more holistic and sophisticated understanding of human motion, taking into account the intricate web of interpersonal dynamics that characterize human interaction.

8.2.3 Fine-grained Controllable Human Motion Synthesis

Recently, several studies have been conducted with the aim of generating human motion sequences based on controllable signals. These studies typically use text or command inputs to produce a sequence of movements that align with the given description. However, controlling the generation of fine-grained motions is still a challenge. For instance, we might want to generate a continuous motion sequence where the human character is sitting for the first three seconds, then gets up and walks for the remaining five seconds. Alternatively, we might want to generate a motion sequence in which the character places their hands on their legs while simultaneously shaking their legs. Enabling the synthesis of such fine-grained, controllable human motion would significantly contribute to sectors like the metaverse, digital human creation, and animation. This would allow for the creation of more nuanced, realistic, and dynamic virtual beings and scenarios, providing a richer and more immersive user experience. Future research should therefore prioritize the development of techniques that can handle this level of precision and flexibility in motion generation.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

[1] Allegro Hand. https://www.wonikrobotics.com/research-robot-hand, 2020.

[2] Barrett Hand. https://medical.barrett.com/about-barrethand, 2020.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC Superpixels Compared to State-Of-The-Art Superpixel Methods. *IEEE TPAMI*, 34(11), 2012.

[4] Ijaz Akhter and Michael J. Black. Pose-Conditioned Joint Angle Limits for 3D Human Pose Reconstruction. In *CVPR*, 2015.

[5] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. A Spatio-Temporal Transformer for 3D Human Motion Prediction. In *3DV*, 2021.

[6] Sadegh Aliakbarian, Pashmina Cameron, Federica Bogo, Andrew Fitzgibbon, and Thomas J Cashman. Flag: Flow-based 3d avatar generation from sparse observations. In *CVPR*, pages 13253–13262, 2022.

[7] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour Detection and Hierarchical Image Segmentation. *IEEE TPAMI*, 33(5):898–916, 2011.

[8] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 2009.

[9] Andreas Aristidou, Ariel Shamir, and Yiorgos Chrysanthou. Digital Dance Ethnography: Organizing Large Dance Collections. *J. Comput. Cult. Herit.*, 12(4), November 2019.

[10] Anurag Arnab and Philip HS Torr. Pixelwise Instance Segmentation with a Dynamically Instantiated Network. In *CVPR*, 2017.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. In *arXiv*, 2016.

[12] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *ICCV*, 2015.

[13] Jonathan T Barron and Ben Poole. The Fast Bilateral Solver. In *ECCV*, 2016.

[14] Jérôme Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple Object Tracking Using K-Shortest Paths Optimization. *IEEE TPAMI*, 33(9), 2011.

[15] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking Without Bells and Whistles. In *CVPR*, 2019.

[16] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. *Springer Handbook of Robotics*, 2008.

[17] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering Human Bodies in Motion. In *CVPR*, 2017.

[18] Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. Contactgrasp: Functional multi-finger grasp synthesis from contact. In *IROS*, 2019.

[19] Matthew Brand and Aaron Hertzmann. Style Machines. In *CGIT*, 2000.

[20] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv*, 2018.

[21] Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised Scene Decomposition and Representation. In *arXiv*, 2019.

[22] Judith Butepage, Michael J. Black, Danica Kragic, and Hedvig Kjellstrom. Deep Representation Learning for Human Motion Prediction and Classification. In *CVPR*, 2017.

[23] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. Anticipating Many Futures: Online Human Motion Prediction and Generation for Human-Robot Interaction. In *ICRA*, 2018.

[24] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-Shot Video Object Segmentation. In *CVPR*, 2017.

[25] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and Stuff Classes in Context. In *CVPR*, 2018.

[26] Yujun Cai, Lin Huang, Yiwei Wang, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Xu Yang, Yiheng Zhu, Xiaohui Shen, and Others. Learning Progressive Joint Propagation for Human Motion Prediction. In *ECCV*, 2020.

[27] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *ICAR*, 2015.

[28] John Canny. A Computational Approach to Edge Detection. *IEEE TPAMI*, 8(6), 1986.

[29] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, and Others. Shapenet: An Information-Rich 3D Model Repository. In *arXiv*, 2015.

[30] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, et al. Dexycb: A benchmark for capturing hand grasping of objects. In *CVPR*, 2021.

[31] Anargyros Chatzitofis, Leonidas Saroglou, Prodromos Boutis, Petros Drakoulis, Nikolaos Zioulis, Shishir Subramanyam, Bart Kevelham, Caecilia Charbonnier, Pablo Cesar, Dimitrios Zarpalas, and Others. HUMAN4D: A Human-Centric Multimodal Dataset for Motions and Immersive Media. In *IEEE Access*, 2020.

[32] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation. In *CVPR*, 2020.

[33] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid Task Cascade for Instance Segmentation. In *CVPR*, 2019.

[34] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D. Collins, Ekin D. Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens. Naive-Student: Leveraging Semi-Supervised Learning in Video Sequences for Urban Scene Segmentation. In *ECCV*, 2020.

[35] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A Foundation for Dense Object Segmentation. In *ICCV*, 2019.

[36] Yi-Wen Chen, Yi-Hsuan Tsai, Yen-Yu Lin, and Ming-Hsuan Yang. VOSTR: Video Object Segmentation via Transferable Representations. *IJCV*, 128(4), 2020.

[37] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-Deeplab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. In *CVPR*, 2020.

[38] Jingchun Cheng, Yi-Hsuan Tsai, Wei-Chih Hung, Shengjin Wang, and Ming-Hsuan Yang. Fast and Accurate Online Video Object Segmentation via Tracking Parts. In *CVPR*, 2018.

[39] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. SegFlow: Joint Learning for Video Object Segmentation and Optical Flow. In *ICCV*, 2017.

[40] Hsu-kuang Chiu, Ehsan Adeli, Borui Wang, De-An Huang, and Juan Carlos Niebles. Action-Agnostic Human Pose Forecasting. In *WACV*, 2019.

[41] Enric Corona, Albert Pumarola, Guillem Alenya, Francesc Moreno-Noguer, and Grégory Rogez. Ganhand: Predicting human grasp affordances in multi-object scenes. In *CVPR*, 2020.

[42] Ioana Croitoru, Simion-Vlad Bogolin, and Marius Leordeanu. Unsupervised Learning from Video to Detect Foreground Objects in Single Images. In *ICCV*, 2017.

[43] Lingwei Dang, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. MSR-GCN: Multi-Scale Residual Graph Convolution Networks for Human Motion Prediction. In *ICCV*, 2021.

[44] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards Segmenting Anything That Moves. In *ICCV*, 2019.

[45] Bert De Brabandere, D. Neven, and L. Van Gool. Semantic Instance Segmentation with a Discriminative Loss Function. In *arXiv*, 2017.

[46] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[47] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to Predict Crisp Boundaries. In *ECCV*, 2018.

[48] Akshay Dhamija, Manuel Gunther, Jonathan Ventura, and Terrance Boult. The Overlooked Elephant of Object Detection: Open Set. In *WACV*, 2020.

[49] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021.

[50] Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1), 1959.

[51] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ICLR*, 2016.

[52] Andrea Dittadi, Sebastian Dziadzio, Darren Cosker, Ben Lundell, Thomas J Cashman, and Jamie Shotton. Full-body motion from a single head-mounted device: Generating smpl poses from partial observations. In *ICCV*, pages 11687–11697, 2021.

[53] Piotr Dollár and C. Lawrence Zitnick. Structured Forests for Fast Edge Detection. In *ICCV*, 2013.

[54] Piotr Dollár and C. Lawrence Zitnick. Structured Forests for Fast Edge Detection. In *ICCV*, 2013.

[55] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 2020.

[56] Yuming Du, Robin Kips, Albert Pumarola, Sebastian Starke, Ali Thabet, and Artsiom Sanakoyeu. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model. In *CVPR*, 2023.

[57] Yuming Du, Philippe Weinzaepfel, Vincent Lepetit, and Romain Brégier. Multi-finger grasping like humans. In *IROS*, pages 1564–1570. IEEE, 2022.

[58] Yuming Du, Yang Xiao, and Vincent Lepetit. Learning to better segment objects from unseen classes with unlabeled videos. In *ICCV*, pages 3375–3384, 2021.

[59] David Eigen and Rob Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In *ICCV*, 2015.

[60] David Eigen, Christian Puhrsch, and Rob Fergus. Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network. In *NeurIPS*, 2014.

[61] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 2010.

[62] Eyes, JAPAN Co. Ltd. Eyes, Jappan.

[63] Alon Faktor and Michal Irani. Video Segmentation by Non-Local Consensus Voting. In *BMVC*, 2014.

[64] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *CVPR*, 2017.

[65] Zhibo Fan, Jin-Gang Yu, Zhihao Liang, Jiarong Ou, Changxin Gao, G. Xia, and Y. Li. FGN: Fully Guided Network for Few-Shot Instance Segmentation. In *CVPR*, 2020.

[66] Carlo Ferrari and John F Canny. Planning optimal grasps. In *ICRA*, 1992.

[67] Advanced Computing Center for the Arts and Design. ACCAD MoCap Dataset.

[68] David A Forsyth, Jitendra Malik, Margaret M Fleck, Hayit Greenspan, Thomas Leung, Serge Belongie, Chad Carson, and Chris Bregler. Finding Pictures of Objects in Large Collections of Images. In *ORCV*, 1996.

[69] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent Network Models for Human Dynamics. In *ICCV*, 2015.

[70] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. In *CVPR*, 2018.

[71] Pascal Fua. Combining Stereo and Monocular Information to Compute Dense Depth Maps That Preserve Depth Discontinuities. In *IJCAI*, pages 1292–1298, August 1991.

[72] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: the KITTI Dataset. *IJRR*, 2013.

[73] Davi Geiger, Bruce Ladendorf, and Alan Yuille. Occlusions and Binocular Stereo. *IJCV*, 14:211–226, 1995.

[74] Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F. Troje. MoVi: A Large Multipurpose Motion and Video Dataset, 2020.

[75] James Jerome Gibson. *The ecological approach to visual perception.* Houghton Mifflin, 1979.

[76] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *ICCV*, 2019.

[77] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *JMLR*, 9:249–256, 01 2010.

[78] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *AISTATS*, pages 315–323, 2011.

[79] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017.

[80] Haifeng Gong, Jack Sim, Maxim Likhachev, and Jianbo Shi. Multi-Hypothesis Motion Planning for Visual Object Tracking. In *ICCV*, 2011.

[81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[82] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-Object Representation Learning with Iterative Variational Inference. In *ICML*, 2019.

[83] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José MF Moura. Adversarial Geometry-Aware Human Motion Prediction. In *ECCV*, 2018.

[84] Wen Guo, Xiaoyu Bie, Xavier Alameda-Pineda, and Francesc Moreno-Noguer. Multi-Person Extreme Motion Prediction. In *CVPR*, 2022.

[85] Wen Guo, Yuming Du, Xi Shen, Vincent Lepetit, Xavier Alameda-Pineda, and Francesc Moreno-Noguer. Back to mlp: A simple baseline for human motion prediction. In *WACV*, pages 4809–4819, 2023.

[86] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images. In *CVPR*, 2013.

[87] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *ECCV*, 2014.

[88] Ikhsanul Habibie, Daniel Holden, Jonathan Schwarz, Joe Yearsley, and Taku Komura. A recurrent variational autoencoder for human motion synthesis. In *BMVC*, 2017.

[89] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *CVPR*, 2020.

[90] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *ICRA*, 2020.

[91] Yana Hasson, Gül Varol, Cordelia Schmid, and Ivan Laptev. Towards unconstrained joint hand-object reconstruction from rgb videos. In *3DV*, 2021.

[92] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *CVPR*, 2019.

[93] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[94] Kaiming He, Jian Sun, and Xiaoou Tang. Guided Image Filtering. *IEEE TPAMI*, 35:1397–1409, 06 2013.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[96] Philipp Heise, Sebastian Klose, Brian Jensen, and Alois Knoll. PM-Huber: Patchmatch with Huber Regularization for Stereo Matching. In *ICCV*, 2013.

[97] Minhyeok Heo, Jaehan Lee, Kyung-Rae Kim, Han-Ul Kim, and Chang-Su Kim. Monocular Depth Estimation Using Whole Strip Masking and Reliability-Based Refinement. In *ECCV*, 2018.

[98] Alejandro Hernandez, Jurgen Gall, and Francesc Moreno-Noguer. Human Motion Prediction via Spatio-Temporal Inpainting. In *ICCV*, 2019.

[99] Geoffrey E. Hinton. Products of Experts. In *ICANN*, 1999.

[100] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[101] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering Surface Layout from an Image. *IJCV*, 75(1):151–172, October 2007.

[102] Derek Hoiem, Andrew N Stein, Alexei A Efros, and Martial Hebert. Recovering Occlusion Boundaries from an Image. *IJCV*, 91(3), 2011.

[103] Ludovic Hoyet, Kenneth Ryall, Rachel Mcdonnell, and Carol O'sullivan. Sleight of Hand: Perception of Finger Motion from Reduced Marker Sets. In *SIGGRAPH*, 2012.

[104] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to Segment Every Thing. In *CVPR*, 2018.

[105] Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J Black, Otmar Hilliges, and Gerard Pons-Moll. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM TOG*, 37(6):1–15, 2018.

[106] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask Scoring R-CNN. In *CVPR*, 2019.

[107] Stephen S Intille and Aaron F Bobick. Disparity-Space Images and Large Occlusion Stereo. In *ECCV*, 1994.

[108] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *IEEE TPAMI*, 2013.

[109] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM ToG*, 2013.

[110] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial Transformer Networks. In *NeurIPS*, 2015.

[111] Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-Rnn: Deep Learning on Spatio-Temporal Graphs. In *CVPR*, 2016.

[112] Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, June 2016.

[113] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. Fusionseg: Learning to Combine Motion and Appearance for Fully Automatic Segmentation of Generic Objects in Videos. In *CVPR*, 2017.

[114] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Self-Supervised Learning of Interpretable Keypoints from Unlabelled Videos. In *CVPR*, 2020.

[115] Junho Jeon and Seungyong Lee. Reconstruction-Based Pairwise Depth Dataset for Depth Image Enhancement Using CNN. In *ECCV*, 2018.

[116] Jisoo Jeong, Seungeui Lee, Jeesoo Kim, and Nojun Kwak. Consistency-Based Semi-Supervised Learning for Object Detection. In *NeurIPS*, 2019.

[117] Jiaxi Jiang, Paul Streli, Huajian Qiu, Andreas Fender, Larissa Laich, Patrick Snape, and Christian Holz. Avatarposer: Articulated full-body pose tracking from sparse motion sensing. *ECCV*, 2022.

[118] Yifeng Jiang, Yuting Ye, Deepak Gopinath, Jungdam Won, Alexander W Winkler, and C Karen Liu. Transformer inertial poser: Attention-based real-time human motion reconstruction from sparse imus. *arXiv*, 2022.

[119] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look Deeper into Depth: Monocular Depth Estimation with Semantic Booster and Attention-Driven Loss. In *ECCV*, 2018.

[120] SouYoung Jin, Aruni Roychowdhury, Huaizu Jiang, Ashish Singh, Aditya Prasad, Deep Chakraborty, and Erik Learned-Miller. Unsupervised Hard Example Mining from Videos for Improved Object Detection. In *ECCV*, 2018.

[121] Takeo Kanade and Masatoshi Okutomi. A Stereo Matching Algorithm with an Adaptative Window: Theory and Experiment. *IEEE TPAMI*, 16(9):920–932, September 1994.

[122] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019.

[123] Kevin Karsch, Zicheng Liao, Jason Rock, Jonathan T Barron, and Derek Hoiem. Boundary Cues for 3D Object Shape Recovery. In *CVPR*, 2013.

[124] Manuel Kaufmann, Yi Zhao, Chengcheng Tang, Lingling Tao, Christopher Twigg, Jie Song, Robert Wang, and Otmar Hilliges. Em-pose: 3d human pose estimation from sparse electromagnetic trackers. In *ICCV*, 2021.

[125] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, and Others. The Kinetics Human Action Video Dataset. In *arXiv*, 2017.

[126] Jihoon Kim, Jiseob Kim, and Sungjoon Choi. Flame: Free-form language-based motion synthesis & editing. *arXiv*, 2022.

[127] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[128] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic Segmentation. In *CVPR*, 2019.

[129] Alexander Kirillov, Evgeny Levinkov, B. Andres, Bogdan Savchynskyy, and C. Rother. InstanceCut: From Edges to Instances with MultiCut. In *CVPR*, 2017.

[130] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image Segmentation as Rendering. In *CVPR*, 2020.

[131] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Korner. Evaluation of CNN-Based Single-Image Depth Estimation Methods. In *ECCV Workshop*, 2018.

[132] Yeong Jun Koh and Chang-Su Kim. Primary Object Segmentation in Videos Based on Region Augmentation and Reduction. In *CVPR*, 2017.

[133] Hema Swetha Koppula and Ashutosh Saxena. Anticipating Human Activities for Reactive Robotic Response. In *IROS*, 2013.

[134] Weicheng Kuo, A. Angelova, Jitendra Malik, and Tsung-Yi Lin. ShapeMask: Learning to Segment Novel Objects by Refining Shape Priors. In *ICCV*, 2019.

[135] Bio Motion Lab. BMLhandball Motion Capture Database.

[136] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *ECCV*, 2020.

[137] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *3DV*, 2016.

[138] Arjun Lakshmipathy, Dominik Bauer, Cornelia Bauer, and Nancy S Pollard. Contact transfer: A direct, user-driven method for human to robot transfer of grasps and manipulations. In *ICRA*, 2022.

[139] Tim Lebailly, Sena Kiciroglu, Mathieu Salzmann, Pascal Fua, and Wei Wang. Motion Prediction Using Temporal Inception Module. In *ACCV*, 2020.

[140] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[141] Jae-Han Lee and Chang-Su Kim. Monocular Depth Estimation Using Relative Depth Maps. In *CVPR*, June 2019.

[142] Youngwan Lee and Jongyoul Park. CenterMask: Real-Time Anchor-Free Instance Segmentation. In *CVPR*, 2020.

[143] Andreas M. Lehrmann, Peter V. Gehler, and Sebastian Nowozin. Efficient Nonlinear Markov Models for Human Motion. In *CVPR*, 2014.

[144] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization Using Optimization. In *ACM Transactions on Graphics*, 2004.

[145] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional Sequence to Sequence Model for Human Dynamics. In *CVPR*, 2018.

[146] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M. Rehg. Video Segmentation by Tracking Many Figure-Ground Segments. In *ICCV*, 2013.

[147] Jun Li, Reinhard Klein, and Angela Yao. A Two-Streamed Network for Estimating Fine-Scaled Depth Maps from Single RGB Images. In *ICCV*, 2017.

[148] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Symbiotic Graph Neural Networks for 3D Skeleton-Based Human Action Recognition and Motion Prediction. *IEEE TPAMI*, 2021.

[149] Maosen Li, Siheng Chen, Yangheng Zhao, Ya Zhang, Yanfeng Wang, and Qi Tian. Dynamic Multiscale Graph Neural Networks for 3D Skeleton Based Human Motion Prediction. In *CVPR*, 2020.

[150] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *ICCV*, 2021.

[151] Xueting Li, Sifei Liu, Shalini De mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-Task Self-Supervised Learning for Temporal Correspondence. In *NeurIPS*, 2019.

[152] Yandong Li, Di Huang, Danfeng Qin, Liqiang Wang, and Boqing Gong. Improving Object Detection with Selective Self-Supervised Self-Training. In *ECCV*, 2020.

[153] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsua Yang. Deep Joint Image Filtering. In *ECCV*, 2016.

[154] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017.

[155] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. PlaneNet: Piece-Wise Planar Reconstruction from a Single RGB Image. In *CVPR*, 2018.

[156] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep Convolutional Neural Fields for Depth Estimation from a Single Image. In *CVPR*, 2015.

[157] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Generating grasp poses for a high-dof gripper using neural networks. In *IROS*, 2019.

[158] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Jizheng Xu, Houqiang Li, and Yan Lu. Affinity Derivation and Graph Merge for Instance Segmentation. In *ECCV*, 2018.

[159] Zhenguang Liu, Shuang Wu, Shuyuan Jin, Qi Liu, Shijian Lu, Roger Zimmermann, and Li Cheng. Towards Natural and Accurate Future Motion Prediction of Humans and Animals. In *CVPR*, 2019.

[160] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-Centric Learning with Slot Attention. In *NeurIPS*, 2020.

[161] Matthew Loper, Naureen Mahmood, and Michael J. Black. MoSh: Motion And Shape Capture from Sparse Markers. 33(6), November 2014.

[162] Matthew Loper, Naureen Mahmood, and Michael J Black. Mosh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics (ToG)*, 33(6):1–13, 2014.

[163] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A Skinned Multi-Person Linear Model. *ACM transactions on graphics (TOG)*, 34(6), 2015.

[164] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv*, 2017.

[165] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[166] Eyes JAPAN Co Ltd. Eyes Japan MoCap Dataset.

[167] Qingkai Lu, Mark Van der Merwe, and Tucker Hermans. Multi-fingered active grasp learning. In *IROS*, 2020.

[168] Xiankai Lu, Wenguan Wang, Jianbing Shen, Yu-Wing Tai, David J. Crandall, and Steven C.H. Hoi. Learning Video Object Segmentation from Unlabeled Videos. In *CVPR*, 2020.

[169] Jonathon Luiten, Idil Esen Zulfikar, and Bastian Leibe. UnOVOST: Unsupervised Offline Video Object Segmentation and Tracking. In *WACV*, 2020.

[170] Jens Lundell, Enric Corona, Tran Nguyen Le, Francesco Verdoja, Philippe Weinzaepfel, Grégory Rogez, Francesc Moreno-Noguer, and Ville Kyrki. Multi-fingan: Generative coarse-to-fine sampling of multi-finger grasps. In *ICRA*, 2021.

[171] Tianyang Ma and Longin Jan Latecki. Maximum Weight Cliques with Mutex Constraints for Video Object Segmentation. In *CVPR*, 2012.

[172] Tiezheng Ma, Yongwei Nie, Chengjiang Long, Qing Zhang, and Guiqing Li. Progressively Generating Better Initial Guesses Towards Next Stages for High-Quality Human Motion Prediction. In *CVPR*, 2022.

[173] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML*, 2013.

[174] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of Motion Capture as Surface Shapes. In *ICCV*, 2019.

[175] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv*, 2021.

[176] Christian Mandery, Ömer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. The KIT Whole-Body Human Motion Database. In *ICAR*, 2015.

[177] Wei Mao, Miaomiao Liu, and Mathieu Salzmann. History Repeats Itself: Human Motion Prediction via Motion Attention. In *ECCV*, 2020.

[178] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning Trajectory Dependencies for Human Motion Prediction. In *ICCV*, 2019.

[179] Bernard Marr. How tesla is using artificial intelligence to create the autonomous cars of the future., 2022. https://bernardmarr.com/how-tesla-is-using-artificial-intelligence-to-create-the-autonomous-cars-of-the-future/.

[180] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *ICCV*, 2001.

[181] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues. *IEEE TPAMI*, 26(5), 2004.

[182] Julieta Martinez, Michael J. Black, and Javier Romero. On Human Motion Prediction Using Recurrent Neural Networks. In *CVPR*, 2017.

[183] Esteve Valls Mascaro, Shuo Ma, Hyemin Ahn, and Dongheui Lee. Robust human motion forecasting using transformer-based model. In *IROS*, pages 10674–10680. IEEE, 2022.

[184] Francisco Massa and Ross Girshick. Maskrcnn-Benchmark: Fast, Modular Reference Implementation of Instance Segmentation and Object Detection Algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018.

[185] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *ICCV*, 2016.

[186] Microsoft. What is augmented reality or ar?, 2023. https://dynamics.microsoft.com/en-us/mixed-reality/guides/what-is-augmented-reality-ar/.

[187] Anton Milan, Laura Leal-Taixé, Konrad Schindler, and Ian Reid. Joint Tracking and Segmentation of Multiple Targets. In *CVPR*, 2015.

[188] Andrew T Miller and Peter K Allen. Examples of 3d grasp quality computations. In *ICRA*, 1999.

[189] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 2004.

[190] MNichols. Robots are transforming the healthcare industry., 2019. https://community.robotshop.com/blog/show/robots-are-transforming-the-healthcare-industry.

[191] Lea Muller, Ahmed AA Osman, Siyu Tang, Chun-Hao P Huang, and Michael J Black. On self-contact and human pose. In *CVPR*, 2021.

[192] Meinard Müller, Tido Röder, Michael Clausen, Bernhard Eberhardt, Björn Krüger, and Andreas Weber. Documentation mocap database HDM05. Technical Report CG-2007-2, Universität Bonn, June 2007.

[193] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*, 2010.

[194] Natalia Neverova, David Novotny, and Andrea Vedaldi. Correlated Uncertainty for Learning Dense Correspondences from Noisy Labels. In *NeurIPS*, 2019.

[195] Van Nguyen Nguyen, Yuming Du, Yang Xiao, Michael Ramamonjisoa, and Vincent Lepetit. Pizza: A powerful image-only zero-shot zero-cad approach to 6 dof tracking. In *3DV*, 2022.

[196] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv*, 2021.

[197] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, pages 8162–8171, 2021.

[198] David Novotny, Samuel Albanie, Diane Larlus, and A. Vedaldi. Semi-Convolutional Operators for Instance Segmentation. In *ECCV*, 2018.

[199] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video Object Segmentation Using Space-Time Memory Networks. In *ICCV*, 2019.

[200] Aljoša Ošep, Paul Voigtlaender, Jonathon Luiten, Stefan Breuers, and Bastian Leibe. Large-Scale Object Mining for Object Discovery from Unlabeled Video. In *ICRA*, 2019.

[201] Aljoša Ošep, Paul Voigtlaender, Mark Weber, Jonathon Luiten, and Bastian Leibe. 4D Generic Video Object Proposals. In *ICRA*, 2020.

[202] Art B Owen. A Robust Hybrid of Lasso and Ridge Regression. *Contemp. Math.*, 443, 2007.

[203] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. 2016.

[204] Anestis Papazoglou and Vittorio Ferrari. Fast Object Segmentation in Unconstrained Video. In *ICCV*, 2013.

[205] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in Pytorch. In *NeurIPS Workshop*, 2017.

[206] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[207] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning Features by Watching Objects Move. In *CVPR*, 2017.

[208] Federico Perazzi, Jordi Pont-Tuset, Brian Mcwilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *CVPR*, 2016.

[209] Federico Perazzi, Oliver Wang, Markus Gross, and Alexander Sorkine-Hornung. Fully Connected Object Proposals for Video Segmentation. In *ICCV*, 2015.

[210] Mathis Petrovich, Michael J. Black, and Gül Varol. Action-Conditioned 3D Human Motion Synthesis with Transformer VAE. In *ICCV*, 2021.

[211] Trung Pham, Vijay BG Kumar, Thanh-Toan Do, Gustavo Carneiro, and Ian Reid. Bayesian Semantic Instance Segmentation in Open Set World. In *ECCV*, 2018.

[212] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to Refine Object Segments. In *ECCV*, 2016.

[213] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Learning Monocular Depth Estimation with Unsupervised Trinocular Assumptions. In *3DV*, pages 324–333, 2018.

[214] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 1991.

[215] Georgy Ponimatkin, Nermin Samet, Yang Xiao, Yuming Du, Renaud Marlet, and Vincent Lepetit. A simple and powerful global optimization for unsupervised video object segmentation. In *WACV*, pages 5892–5903, 2023.

[216] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *ECCV*, 2022.

[217] Xuchong Qiu, Yang Xiao, Chaohui Wang, and Renaud Marlet. Pixel-pair occlusion relationship map (p2orm): formulation, inference and application. In *ECCV*, 2020.

[218] Ilija Radosavovic, P. Dollár, Ross B. Girshick, Georgia Gkioxari, and Kaiming He. Data Distillation: Towards Omni-Supervised Learning. In *CVPR*, 2018.

[219] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.

[220] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv*, 7(1):5, 2017.

[221] Michael Ramamonjisoa, Yuming Du, and Vincent Lepetit. Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In *CVPR*, pages 14648–14657, 2020.

[222] Michael Ramamonjisoa and Vincent Lepetit. Sharpnet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation. In *ICCV Workshop*, 2019.

[223] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv*, 2022.

[224] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. In *ICCV*, 2021.

[225] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Figure/ground Assignment in Natural Images. In *ECCV*, 2006.

[226] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, pages 1530–1538. PMLR, 2015.

[227] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM ToG*, 2017.

[228] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.

[229] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, and Richard Newcombe. FroDO: From Detections to 3D Objects. In *CVPR*, 2020.

[230] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. In *CoRL*, 2021.

[231] Mingyi Shi, Kfir Aberman, Andreas Aristidou, Taku Komura, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Motionet: 3d human motion reconstruction from monocular video with skeleton consistency. *ACM TOG*, 40(1):1–15, 2020.

[232] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *IJCV*, 2010.

[233] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012.

[234] Theodoros Sofianos, Alessio Sampieri, Luca Franco, and Fabio Galasso. Space-Time-Separable Graph Convolutional Network for Pose Forecasting. In *ICCV*, 2021.

[235] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

[236] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. 2021.

[237] Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. Local motion phases for learning multi-contact character movements. *ACM TOG*, 39(4):54–1, 2020.

[238] Otilia Stretcu and Marius Leordeanu. Multiple Frames Matching for Object Discovery in Video. In *BMVC*, 2015.

[239] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *ICRA*, 2021.

[240] Richard Szeliski and Richard Weiss. Robust Shape Recovery from Occluding Contours Using a Linear Smoother. *IJCV*, 28(1):27–44, 1998.

[241] Omid Taheri, Nima Ghorbani, Michael J Black, and Dimitrios Tzionas. Grab: A dataset of whole-body human grasping of objects. In *ECCV*, 2020.

[242] Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. Modeling Human Motion Using Binary Latent Variables. In *NIPS*, 2007.

[243] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *ECCV*, 2020.

[244] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Amit H Bermano, and Daniel Cohen-Or. Human motion diffusion model. *ICLR*, 2023.

[245] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional Convolutions for Instance Segmentation. In *ECCV*, 2020.

[246] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

[247] Carlo Tomasi and Roberto Manduchi. Bilateral Filtering for Gray and Color Images. In *ICCV*, 1998.

[248] Nikolaus F. Troje. Decomposing Biological Motion: A Framework for Analysis and Synthesis of Human Gait Patterns. *Journal of Vision*, 2(5), September 2002.

[249] Matt Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors. In *BMVC*, 2017.

[250] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total capture: 3d human pose estimation fusing video and inertial sensors. In *BMVC*, 2017.

[251] Carnegie Mellon University. CMU MoCap Dataset.

[252] Simon Fraser University and National University of Singapore. SFU Motion Capture Database.

[253] Jacob Varley, Jonathan Weisz, Jared Weiss, and Peter Allen. Generating multi-fingered robotic grasps via deep learning. In *IROS*, 2015.

[254] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017.

[255] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marques, and Xavier Giro-I-Nieto. RVOS: End-To-End Recurrent Network for Video Object Segmentation. In *CVPR*, 2019.

[256] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, 2018.

[257] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. FEELVOS: Fast End-To-End Embedding Learning for Video Object Segmentation. In *CVPR*, 2019.

[258] Timo Von Marcard, Roberto Henschel, Michael J. Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering Accurate 3D Human Pose in the Wild Using IMUs and a Moving Camera. In *ECCV*, 2018.

[259] Kentaro Wada, Kei Okada, and Masayuki Inaba. Joint Learning of Instance and Semantic Segmentation for Robotic Pick-and-Place with Heavy Occlusions in Clutter. In *ICRA*, 2019.

[260] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian Process Dynamical Models. In *NIPS*, 2005.

[261] Jiashun Wang, Huazhe Xu, Jingwei Xu, Sifei Liu, and Xiaolong Wang. Synthesizing long-term 3d human motion and interaction in 3d scenes. In *CVPR*, 2021.

[262] Peng Wang, Xiaohui Shen, Bryan Russell, Scott Cohen, Brian Price, and Alan L Yuille. SURGE: Surface Regularized Geometry Estimation from a Single Image. In *NeurIPS*, 2016.

[263] Peng Wang and Alan Yuille. DOC: Deep Occlusion Estimation from a Single Image. In *ECCV*, 2016.

[264] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning Correspondence from the Cycle-Consistency of Time. In *CVPR*, 2019.

[265] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. SOLO: Segmenting Objects by Locations. In *ECCV*, 2020.

[266] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic, Faster and Stronger. In *NeurIPS*, 2020.

[267] Zhiyong Wang, Jinxiang Chai, and Shihong Xia. Combining recurrent neural networks and adversarial training for human motion synthesis and control. In *IEEE TVCG*, 2019.

[268] Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual Interaction Networks: Learning a Physics Simulator from Video. In *NeurIPS*, 2017.

[269] Alexander Winkler, Jungdam Won, and Yuting Ye. Questsim: Human motion tracking from sparse sensors with simulated avatars. In *SIGGRAPH Asia*, pages 1–8, 2022.

[270] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying Unknown Instances for Autonomous Driving. In *CoRL*, 2020.

[271] Bohan Wu, Iretiayo Akinola, Abhi Gupta, Feng Xu, Jacob Varley, David Watkins-Valls, and Peter K Allen. Generative attention learning: a "general" framework for high-performance multi-fingered grasping in clutter. *Autonomous Robots*, 2020.

[272] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast End-To-End Trainable Guided Filter. In *CVPR*, 2018.

[273] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[274] Zheng Wu, Thomas H. Kunz, and Margrit Betke. Efficient Track Linking Methods for Track Graphs Using Network-Flow and Set-Cover Techniques. In *CVPR*, 2011.

[275] Yang Xiao, Yuming Du, and Renaud Marlet. Posecontrast: Class-agnostic object viewpoint estimation in the wild with pose-aware contrastive learning. In *3DV*, 2021.

[276] Ren Xiaofeng and Liefeng Bo. Discriminatively Trained Sparse Code Gradients for Contour Detection. In *NeurIPS*, 2012.

[277] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. The Best of Both Modes: Separately Leveraging RGB and Depth for Unseen Object Instance Segmentation. In *CoRL*, 2020.

[278] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single Shot Instance Segmentation with Polar Representation. In *CVPR*, 2020.

[279] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3D: Fully Automatic 2D-To-3d Video Conversion with Deep Convolutional Neural Networks. In *ECCV*, 2016.

[280] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Multi-Scale Continuous CRFs as Sequential Deep Networks for Monocular Depth Estimation. In *CVPR*, 2017.

[281] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Monocular Depth Estimation Using Multi-Scale Continuous CRFs as Sequential Deep Networks. *IEEE TPAMI*, 2018.

[282] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and L. Lin. Meta R-CNN: Towards General Solver for Instance-Level Low-Shot Learning. In *ICCV*, 2019.

[283] Dongseok Yang, Doyeon Kim, and Sung-Hee Lee. Lobstr: Real-time lower-body pose prediction from sparse upper-body tracking signals. In *Comput. Graph. Forum*, volume 40, pages 265–275. Wiley Online Library, 2021.

[284] Linjie Yang, Yuchen Fan, and Ning Xu. Video Instance Segmentation. In *ICCV*, 2019.

[285] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. LEGO: Learning Edge with Geometry All at Once by Watching Videos. In *CVPR*, 2018.

[286] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised Learning of Geometry from Videos with Edge-Aware Depth-Normal Consistency. In *AAAI*, 2018.

[287] Yongjing Ye, Libin Liu, Lei Hu, and Shihong Xia. Neural3Points: Learning to Generate Physically Realistic Full-body Motion for Virtual Reality Users. 2022.

[288] Lin Yen-Chen, Andy Zeng, Shuran Song, Phillip Isola, and Tsung-Yi Lin. Learning to See Before Learning to Act: Visual Pre-Training for Manipulation. In *ICRA*, 2020.

[289] Xinyu Yi, Yuxiao Zhou, Marc Habermann, Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Feng Xu. Physical inertial poser (pip): Physics-aware real-time human motion tracking from sparse inertial sensors. In *CVPR*, pages 13167–13178, 2022.

[290] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing Geometric Constraints of Virtual Normal for Depth Prediction. In *ICCV*, 2019.

[291] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. GMCP-Tracker: Global Multi-Object Tracking Using Generalized Minimum Clique Graphs. In *ECCV*, 2012.

[292] Dong Zhang, Omar Javed, and Mubarak Shah. Video Object Segmentation through Spatially Accurate and Temporally Dense Extraction of Primary Object Regions. In *CVPR*, 2013.

[293] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, and Others. Resnest: Split-Attention Networks. In *arXiv*, 2020.

[294] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv*, 2022.

[295] Rufeng Zhang, Zhi Tian, Chunhua Shen, Mingyu You, and Youliang Yan. Mask Encoding for Single Shot Instance Segmentation. In *CVPR*, 2020.

[296] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, 2018.

[297] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Yan Yan, Nicu Sebe, and Jian Yang. Pattern-Affinitive Propagation Across Depth, Surface Normal and Semantic Segmentation. In *CVPR*, 2019.

[298] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network. In *CVPR*, 2017.

[299] Chongyang Zhong, Lei Hu, Zihao Zhang, Yongjing Ye, and Shihong Xia. Spatio-Temporal Gating-Adjacency GCN For Human Motion Prediction. In *CVPR*, 2022.

[300] Yanzhao Zhou, Xin Wang, Jianbin Jiao, and Trevor Darrell. Learning Saliency Propagation for Semi-Supervised Instance Segmentation. In *CVPR*, 2020.

[301] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, pages 5745–5753, 2019.

[302] Tianqiang Zhu, Rina Wu, Xiangbo Lin, and Yi Sun. Toward human-like grasp: Dexterous grasping via semantic representation of object-hand. In *ICCV*, 2021.

[303] Laurent Zwald and Sophie Lambert-Lacroix. The Berhu Penalty and the Grouped Effect. In *arXiv*, 2012.