LEARNED MOTION MODELS FOR THE PERCEPTION AND GENERATION OF DYNAMIC HUMANS AND OBJECTS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Davis Rempe
March 2023

This dissertation is online at: https://purl.stanford.edu/kc338bg9787

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Leonidas Guibas, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jeannette Bohg

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Karen Liu

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format.

Abstract

Understanding the motion of humans and objects is key for intelligent systems. Motions are the result of physics, but non-physical dynamics also play an important role, for example, social norms and traffic laws determine how pedestrians and vehicles behave. The ability to perceive and generate these motions enables important applications, such as autonomous robots that operate in the real world, mixed reality that augments the real world, and animation and simulation that imitate the real world. Despite often being approached as separate problems, the perception and generation of motion both fundamentally rely on having an accurate model of dynamics for humans and objects in a scene. Perception problems like pose estimation, tracking, and shape estimation require motion understanding to reason about occlusions and noise from partial and ambiguous inputs. Generation problems such as forecasting future motion rely entirely on being able to predict motion. A promising avenue to solve these problems is learning models of motion, however, it is challenging to develop models that accurately reflect the real world, capture the diversity of motion due to inherent uncertainty, and robustly generalize to many possible scenarios.

This thesis explores how to effectively learn models of motion to solve important perception and generation problems. We propose several data-driven methods to accurately capture the dynamics of humans, objects, and how they interact with each other and their environment. In the first part of the thesis, we introduce two methods for perceiving 3D human pose and 3D object shape, respectively. The first uses a robust generative model of 3D human pose transitions, while the second learns a continuous motion representation entirely from point cloud observations. The second part of the thesis focuses on motion models for synthesizing high-level human behavior in the form of 2D top-down trajectories. In these works, we introduce two new generative models that handle complex multi-agent interactions and can be controlled by a user to produce trajectories with desirable properties. We show this is useful to create rare scenarios for testing autonomous vehicles and to animate crowds of pedestrians. Finally, the thesis ends with a discussion of important future directions to continue improving learned models of motion for humans and objects.

Acknowledgments

This thesis would not have been possible without the support, mentorship, and encouragement from so many people over the last six years. First and foremost, I want to thank Leo Guibas for being an incredible advisor and mentor throughout my PhD. It has been a long and difficult journey at times, but was an extremely rewarding experience in Leo's lab. He is always someone to look up to, both as a researcher and a person, and has been supportive, patient, and kind the whole way. I appreciate that he let me work independently and do things in my own way, but still knew when to step in and give me a nudge in the right direction. As a researcher, he taught me to think deeply about problems, see the bigger picture, and find connections and interests across a large variety of topics.

Thank you to Prof. Jeannette Bohg and Prof. Karen Liu for being on my reading committee and for several insightful discussions about projects and ideas over the years. Thank you to Prof. Marco Pavone for being the chair of my defense committee. I really enjoyed getting to work with Marco and his team on interesting problems in traffic modeling during my internships at NVIDIA. Also thank you to Prof. Sanja Fidler for being on my defense committee and for being a great mentor at NVIDIA. Sanja has a tangible excitement for interesting research problems that is extremely motivating. She also really helped me to be more collaborative and put together a larger vision for my research.

I want to thank my incredible and inspiring postdoc mentors at Stanford, who were a huge help especially early in my PhD. Srinath Sridhar was the first person I worked with in Leo's lab, and he was such a calming presence and incredibly patient when I knew absolutely nothing working on my first few projects. He taught me so many research fundamentals, how to communicate ideas clearly, how to persevere through paper rejections and to believe in my work, and was always available to chat about research or otherwise. Also thank you to Tolga Birdal for pushing me to think more deeply and fundamentally about the problems we worked together on. Tolga has a theoretical viewpoint that always brought a fresh and useful perspective, and his interest in so many types of problems and knowledge in general was an inspiration for me. Outside of research, it was also fun to play music together.

During my PhD, I had impactful internships thanks to fantastic mentors. Thank you to Jimei Yang for mentoring me at Adobe, getting me interested in human motion problems, and helping to push me over the learning curve to have successful projects in these areas. Jimei's passion for projects to enable creative applications has been very inspiring for my PhD research and for my future goals as a researcher. I would also like to thank Or Litany who has been a pleasure to work with over the last couple years at NVIDIA. Or is always able to offer creative insights on problems and I have really enjoyed our discussions on such a wide range of topics. He has also been a reliable source of help for everything from research, to navigating collaborations, to life decisions.

I am very lucky to have collaborated with so many amazing people both through Stanford and internships. They helped me immensely to grow as a researcher, collaborator, and mentor, and offered a diversity of perspectives on both research and life that I really appreciate. Thank you especially to my co-authors (roughly chronological order): He Wang, Julien Valentin, Sofien Bouaziz, Aaron Hertzmann, Bryan Russell, Ruben Villegas, Yongheng Zhao, Zan Gojcic, Ali Kashefi, Jonah Philion, Colton Stearns, Jie Li, Rares Ambrus, Sergey Zakharov, Vitor Guizilini, Yanchao Yang, Boxiao Pan, Will Shen, Despoina Paschalidou, Kaichun Mo, Ziyuan Zhong, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, Zhengyi Luo, Jason Peng, Ye Yuan, Kris Kitani, and Karsten Kreis.

Thank you to all of my labmates that have come and gone over the years. They have been a great source of encouragement, feedback, discussion, and friendship to get my mind off research every once in a while. Thank you to Prof. Ron Fedkiw and Prof. Doug James for getting my PhD off to a great start with rotations in their labs. During those rotations, Zhenglin Geng, Jenny Jin, and Jui-Hsien Wang were extremely helpful to get me oriented with research and life at Stanford. Thank you to the administrative staff in the CS Department that have been so helpful during my time at Stanford, especially Carrie Petersen and Jayanthi Subramanian. Also thank you to Steve Reichenbach back at the University of Nebraska who helped nurture my interest in research during my undergraduate years.

Finally, thank you to my amazing wife Allie who has given me endless love, support, and patience throughout my PhD. She has kept me sane through long days and deadlines, and been a constant reminder to enjoy life outside of research and to go on vacation. Thank you to my mom, dad, brother, and sister for their encouragement and support. Without the hard work of my parents, I never could have made it to Stanford in the first place. They have been positive role models all my life, and enabled me to have the dedication and motivation to pursue a PhD. Thank you to my friends Drew Dupont, Nam Tran, and Cale Neelly who helped me to have fun and relax even when I could only see them a few times a year.

Contents

Abstract iv

Acknowledgments v

1 Introduction 1

1.1 Motion Modeling for Perception and Generation . . . 3

1.1.1 Modeling Motion and Accompanying Challenges . . . 3

1.1.2 Perceiving Motion . . . 5

1.2 Thesis Outline and Contributions . . . 7

2 3D Human Motion Models for Pose Estimation 11

2.1 Introduction . . . 11

2.2 Related Work . . . 13

2.3 HuMoR: 3D Human Dynamics Model . . . 14

2.3.1 Training . . . 17

2.4 Test-time Motion Optimization . . . 18

2.4.1 Optimization Variables . . . 18

2.4.2 Objective & Optimization . . . 19

2.5 Experimental Results . . . 21

2.5.1 Datasets . . . 21

2.5.2 Baselines and Evaluation Metrics . . . 22

2.5.3 Generative Model Evaluation . . . 23

2.5.4 Estimation from 3D Observations . . . 24

2.5.5 Estimation from RGB(-D) Observations . . . 26

2.6 Discussion . . . 28

2.7 Additional Related Contributions . . . 29
2.7.1 Physics-Based Human Motion Modeling . . . 30

3 Modeling 3D Object Motion for Point Cloud Perception 31
3.1 Introduction . . . 31
3.2 Related Work . . . 33
3.3 Background . . . 35
3.4 Method . . . 36
3.4.1 Network Architecture . . . 38
3.5 Experimental Evaluations . . . 40
3.5.1 Evaluations and Applications . . . 41
3.6 Discussion . . . 50
3.7 Additional Related Contributions . . . 51
3.7.1 Spatiotemporal Modeling for 3D Object Tracking . . . 51
3.7.2 Predicting the Future Motion of 3D Objects . . . 52

4 Learned Traffic Model for Scenario Generation 54
4.1 Introduction . . . 55
4.2 Related Work . . . 57
4.3 Challenging Scenario Generation . . . 58
4.3.1 Modeling "Realism": Learned Traffic Model . . . 60
4.3.2 Adversarial Optimization . . . 62
4.4 Analyzing and Using Generated Scenarios . . . 65
4.4.1 Filtering and Collision Classification . . . 65
4.4.2 Improving the Planner . . . 66
4.5 Experiments . . . 67
4.5.1 Scenario Generation Evaluation . . . 67
4.5.2 Analyzing Generated Scenarios . . . 70
4.5.3 Improving Rule-Based Planner . . . 71
4.5.4 Traffic Model Prediction Evaluation . . . 73
4.6 Discussion . . . 74

5 Controllable Trajectory Generation 77
5.1 Introduction . . . 77

5.2 Related Work . . . 80
5.3 Method . . . 81
5.3.1 Controllable Trajectory Diffusion . . . 82
5.3.2 Physics-Based Pedestrian Animation . . . 86
5.3.3 Controllable Pedestrian Animation System . . . 87
5.4 Experiments . . . 88
5.4.1 Augmenting Crowd Simulation . . . 89
5.4.2 Real-world Data Evaluation . . . 91
5.4.3 Controllable Pedestrian Animation . . . 93
5.5 Discussion . . . 94
5.6 Additional Related Contributions . . . 95
5.6.1 Controllable Traffic Generation . . . 95

6 Conclusion and Future Vision **97**

Bibliography **101**

List of Tables

1.1 Summary of thesis contributions 9

2.1 HuMoR generation results 23
2.2 Motion and shape estimation from 3D observations 24
2.3 Motion and shape from RGB video on i3DB 25
2.4 Plausibility evaluation on videos in PROX 27

3.1 Canonicalization performance. 42
3.2 Partial surface sequence reconstruction results 43
3.3 Pose estimation using T-NOCS 45
3.4 Non-rigid reconstruction and correspondences 46
3.5 Segmentation label propagation performance 50

4.1 Evaluation of generated challenging scenarios 69
4.2 Scenario generation for Replay planner 69
4.3 Improving Rule-Based planner 72
4.4 Learned traffic model future prediction accuracy 74
4.5 Traffic model ablation study 74

5.1 Guidance evaluation on ORCA-Maps dataset 90
5.2 Guidance evaluation on nuScenes 91
5.3 Closed-loop animation results 93
5.4 Effect of value function guidance 94

List of Figures

1.1 Problems involving modeling motion 2
1.2 Thesis overview 8

2.1 HuMoR method overview 12
2.2 HuMoR CVAE architecture 15
2.3 Fitting to partial 3D keypoints results 23
2.4 Results on fitting to noisy 3D joints 24
2.5 Comparison fitting to 3D keypoints 25
2.6 Qualitative comparison fitting to RGB video 26
2.7 Fitting to RGB-D results 27
2.8 Fitting example on dynamic dancing data 28
2.9 Failure cases 29
2.10 Overview of physics-based human motion from video 30

3.1 CaSPR overview 32
3.2 Architecture and applications of CaSPR 37
3.3 Architecture of point-set canonicalization network 39
3.4 Canonicalization applications 40
3.5 Reconstruction results 43
3.6 Canonicalization, aggregation, and dense reconstruction results 44
3.7 Continuous interpolation results 45
3.8 Additional examples of spatiotemporal interpolation 46
3.9 Reconstruction results on Warping Cars data 47
3.10 Cross-instance correspondences 48
3.11 Disentanglement examples on warping cars data 48
3.12 Propagating segmentation labels over time and instances 49

3.13 Failure cases of CaSPR 50
3.14 SpOT sequence refinement results 51
3.15 Overview of predicting future 3D object motion 52

4.1 STRIVE overview 56
4.2 Test-time architecture of the learned traffic model 60
4.3 Adversarial optimization step 63
4.4 Qualitative results on the Rule-based planner 68
4.5 Qualitative comparison of generated scenarios for the Replay planner 70
4.6 Analysis of generated scenarios 71
4.7 Proof-of-concept results on pedestrian and cyclist adversaries 75
4.8 STRIVE failure cases 76

5.1 Pedestrian animation system overview 78
5.2 Trajectory diffusion model 83
5.3 Pedestrian animation controller pipeline 86
5.4 Guidance results on ORCA-Maps 89
5.5 nuScenes results demonstrating flexibility of TRACE 92
5.6 Animation system capabilities 93
5.7 Overview of controllable traffic generation method 95

Chapter 1

Introduction

Humans and objects are in constant motion and frequently interact in interesting and complex ways. *Humans* move by articulating joints over time based on their intention to perform certain actions. This motion depends on their physical capabilities, which are determined by body shape and strength. *Objects* encapsulate a wider variety of motions. For example, cars move based on accelerations and steering, which are the result of human behavior and control. Household objects move rigidly (*e.g.*, a bottle tipping over) or articulate (*e.g.*, a pair of scissors), while clothing and food may deform in complicated ways. Finally, *interactions* determine how humans and objects move together, and how motion is affected by the surrounding environment. For example, pedestrians move in crowds to avoid collisions, while vehicles respect the rules of the road including traffic laws and lane markers.

The motion of humans and objects fundamentally follows the laws of *physics*, which determine dynamics based on physical forces (gravity, joint torques), properties (mass, friction), and the environment (collisions). However, at a more abstract or *semantic* level, motions are the result of intangible interactions that we consider on a daily basis. Social "forces" determine how pedestrians interact with each other based on cultural norms, which change if a person is walking alone, with a colleague, or with a close friend. Traffic laws constrain how we drive, but may be selectively ignored by more aggressive drivers. Objects have certain semantics and affordances that determine how humans move during interactions: we usually sit on a couch but lay down on a bed, even though each object supports both actions.

The work in this thesis is motivated by the fact that intelligent computational systems require an *understanding* of both the physical and semantic aspects of motion to operate successfully. Applications of these systems range from physical systems in the real world to completely digital ones, and they face several challenges inherent to motion. For example, autonomous systems like

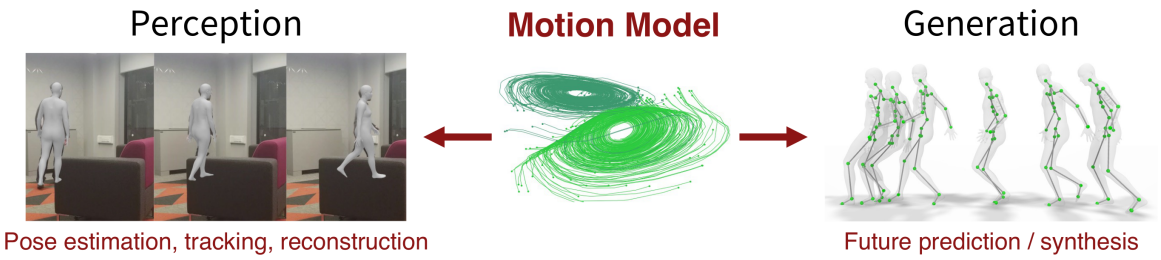

Figure 1.1: Motion *understanding* encapsulates fundamental problems in both perception (left) and generation (right). In this example for human motion, both sets of problems require knowing how human pose is likely to change over time, which can be captured by a model of motion. The spiral trajectory image was taken from [37].

self-driving vehicles and home robots move around the real world based on visual inputs, much like humans. They need to see moving entities around them from noisy sensors, which may be partially or even fully occluded. They must anticipate how surrounding entities will move in the future while accounting for potential uncertainty, and then decide how to proceed to achieve their goals. Importantly, these systems must deal with a variety of motions from rigid or articulated bodies to non-rigid deformation.

There are also applications that represent the dynamic world digitally, such as animating characters or simulating traffic. These systems must synthesize realistic and diverse motion for a variety of characters and objects. They must also resolve interactions between humans and objects, for example to simulate motions that avoid collisions and follow social norms. Often, these applications are inherently creative and therefore require that users have some controllability over motion.

Across these motivating applications, motion *understanding* encapsulates a wide range of capabilities that includes fundamental tasks in computer vision. These problems are both discriminative (*i.e.*, involve *perception*) and generative (*i.e.*, involve *prediction* or *synthesis*). Perception is the ability to parse the contents of a dynamic scene from visual inputs like an RGB video or point cloud sequence. For example, estimating the 3D pose of objects and humans, tracking how the pose evolves over time, and reconstructing their 3D geometry. Prediction, on the other hand, determines how entities are likely to move in the future, subject to constraints from their environment and interactions with each other. This kind of forecasting could be for top-down 2D trajectories, full 3D poses, or 3D geometry (*e.g.*, a point cloud sequence).

The work in this thesis tackles both perception and generation problems, but takes a unified approach that is built on learned models of motion. We will propose several learning-based methods that deliberately model the underlying dynamics of humans and objects to accurately capture their

motions and how they interact with each other and their surroundings. We will see how these learned models can be leveraged for perception problems like 3D pose estimation and reconstruction, and for generative problems such as creating traffic scenarios and simulating crowds. To begin, Sec. 1.1 will define motion modeling, discuss the key challenges, and place the thesis work broadly in the context of prior approaches (an in depth discussion of related work for specific applications and tasks is relegated to later chapters). Then, Sec. 1.2 will outline the thesis organization and discuss how the works are connected by a throughline of effective motion modeling.

1.1 Motion Modeling for Perception and Generation

Perceiving motion and generating motion are often tackled independently using distinct approaches. As an example, let us consider 3D human motion (*i.e.*, the change in joint pose over time). 3D human motion can be generated by learning to animate characters [147, 77, 103, 196]. On the perception side, there are several recent learned approaches to recover 3D human motion from RGB video [114, 231, 126]. Despite both of these problems relying on the same underlying motion understanding – knowing how humans are expected to move – animation is achieved with generative or predictive models, while perception uses discriminative feed-forward networks that, at best, use a motion prior only during training.

In contrast, the work in this thesis views the the generation and perception of motion as two sides of the same problem: as shown in Fig. 1.1, both rely on a detailed understanding of motion. Therefore, modeling the underlying dynamics of humans and objects in a scene should be useful for solving both sets of problems. This kind of *motion model* directly solves motion generation problems where the goal is to synthesize or predict realistic future motions. For perception, the motion model indicates how a person or object is expected to move, which provides a strong prior on the pose and geometry that is being estimated. So how should we approach modeling motion in a way that is useful for both perception and generation? And how do motion models help to tackle perception challenges compared to other approaches?

1.1.1 Modeling Motion and Accompanying Challenges

At a high level, the goal of a *motion model* is to predict future states for a human or object based on past states and the surrounding context, such as the static environment or motion of other dynamic entities. The state space of the motion and representation of the context varies based on the domain. Throughout this thesis, we will primarily discuss three different domains: (**1**) 3D humans, where

motion is the change in articulated pose over time, (**2**) 3D objects represented as point clouds, where motion is the change in object shape, *i.e.*, evolution of surface points, and (**3**) vehicle and pedestrian behavior, where motion is represented using 2D bird's-eye view trajectories.

Across these domains, there are key features that a motion model must possess to be useful in downstream applications. Motion models must be:

- *Accurate* – Motions should reflect real-world data and exhibit both physical and semantic plausibility. This means predicting motions that respect the laws of physics (accelerations, contacts, etc.) along with non-physical constraints (*e.g.*, social norms and road rules).
- *Diverse* – The model should reflect uncertainty by capturing the full distribution of possible outcomes. This is necessary because motion prediction is usually an under-constrained problem with many correct answers. For example, the (unobservable) intent of a person has a large effect on how they move.
- *Robust* – The model should make reasonable predictions from partial or ambiguous inputs. For example, 3D sensors provide partial and noisy point cloud observations, from which we want to recover past motion or predict future motion.
- *Generalizable* – The model should be accurate for all instances within a class of motions. For example, a model to predict 3D human poses should work for all body shapes and not just the average person.
- *Interaction-aware* – Motion predictions must account for surrounding context. This means successfully modeling both physical and semantic interactions with the static environment and with other dynamic entities.
- *Controllable* – A user should be able to constrain output motions to meet objectives. For creative applications, this could be hitting specific keyframe poses or traveling to goal locations. This is also useful for perception, in which case motion must be constrained to match observations.

One way to model motion is by explicitly leveraging the laws of physics. If we have the full state of the system being modeled, including the state of the human/object, their surroundings, and physical properties like mass and friction, we can simulate motion using an explicit physics-based model to get accurate and realistic dynamics [111, 148, 64, 66, 23, 22]. As discussed in Sec. 2.7.1, this kind of approach can be successfully used for perception tasks like animating 3D characters

from video [208]. It also nicely generalizes to different humans and objects since the underlying physics does not change. However, using explicit physics relies on knowing information that is not always given or observable from visual inputs like video. In addition to physical properties like mass and friction, contacts with the environment must be known or inferred, along with the forces being exerted by the person which result from high-level intent. Moreover, even though a physics-based model gives *physically*-realistic motion, it will not necessarily be *semantically* plausible. To get semantics, a high-level controller must determine forces that will, *e.g.*, fulfill the intent of a person or follow social norms.

This motivates using learned motion models, which have the potential to overcome the issues of explicit physics while still implicitly capturing physically-plausible motion. Data-driven models are attractive because, with enough data, they can learn not only physical dynamics, but also motion resulting from semantic [242, 92] or social interactions (*e.g.*, social distancing in crowds [298, 228, 4, 31]). They can also be formulated probabilistically to handle noise and partiality, capture uncertainty, and learn semantic plausibility by training with maximum likelihood objectives.

To effectively leverage the capabilities of learned models, there are key technical design challenges that must be solved, which will be discussed extensively in this thesis. For example:

- *The architecture and state space* of the model must be carefully designed for each domain to be accurate and expressive. Unlike physics-based models that have a clearly-defined state space and dynamics function, neural network motion models can benefit from additional state information [311] and network structures that reflect inductive bias in a domain [29, 131].
- *Generalizing* learned motion models to unseen humans and objects at test time requires carefully-designed architectures and large and diverse training datasets. Even then, ensuring models maintain accuracy outside of the training domain can be tricky, especially for closed-loop motion synthesis such as traffic or crowd simulation.
- *Incorporating constraints on motion* in a flexible way at test time is not straightforward for black-box neural networks. This is problematic for multi-agent interactions (*e.g.*, collision avoidance) [163, 164] and user controllability (*e.g.*, keyframe constraints) that inherently require hard constraints.

1.1.2 Perceiving Motion

Considering the numerous challenges outlined for effectively modeling motion, is it worthwhile to use motion models for perception problems? After all, there are discriminative approaches to 3D

pose estimation, tracking, and reconstruction that do not require developing a fully-featured model of motion. For example, due to the rapid improvement of deep learning methods that estimate pose or shape from a single frame of data (*e.g.*, an RGB image or single point cloud) [129, 113, 241], it is tempting to simply apply these *static* methods to dynamic data on a per-frame basis to solve perception tasks. However, this approach cannot guarantee temporal coherence due to inherent uncertainty: entities are often occluded or disappear completely and, in the case of image observations, 2D projection introduces ambiguity. Therefore, it is important to consider a wide temporal context to produce plausible and accurate estimations across a sequence [58].

For neural network methods, a simple approach is to take the entire observation sequence as input (*e.g.*, RGB video or point cloud sequence) and make an estimation at all frames simultaneously. The hope is that during training, the model will learn temporal correlations present in the data and produce plausible outputs. Alternatively, these motion patterns can be explicitly encouraged during training using loss functions for smoothness [104] or physical plausibility, or using adversarial training that employs a discriminator to increase the realism of recovered motions [126]. Though at test time, there is no guarantee that plausibility will be maintained: the black-box network will learn some implicit model of motion that may be entangled with other factors such as appearance. This makes generalization to novel inputs potentially challenging, and limits the amount of controllability available to the user.

Explicitly modeling the underlying motion of observed entities can more effectively inform perception. Intuitively, if the model understands how humans and objects are *expected* to move, it should be easy to determine whether estimated pose, tracks, and shape are plausible. This can be seen as *analysis-by-synthesis*, and has several advantages over direct discriminative approaches. For example, if the motion model captures realistic motion, it can be used to regularize or even directly parameterize the outputs of the perception model. This will ensure that outputs are constrained to the valid motion manifold, even at test time, which is particularly useful when observations are noisy or contain occlusions. In these cases, the motion model can continue to track how humans/objects are evolving even if they completely disappear from view. Similarly, the motion model can anticipate future motions to inform perception in later observations. Finally, if the motion model parameterizes the output of perception, then the motion will be disentangled from distracting features like appearance.

A simple example of explicitly leveraging a motion model is track-by-detection frameworks [293]. In these methods, object detections at a single step are given as input, and the goal is to associate detections across time. Several methods explicitly model object states and use a linear dynamics

model to estimate future states and thereby inform the association across frames [280]. These simple dynamics models are a good start but will not be expressive enough for many applications, especially those dealing with complicated articulation and deformation. More sophisticated motion models have been employed as optimization priors to recover 3D human and object pose sequences. This includes both physics-based models [208, 25, 93, 173] along with probabilistic models learned from data [258, 275]. Using highly expressive neural network models as motion priors is a relatively recent direction [159], and will be discussed extensively in Chapters 2 and 3. Developing effective and expressive motion models for perception inherits the challenges outlined in Sec. 1.1.1; robustness and generalizability are particularly important. This is because input observations may be noisy and irregular, and if a learned motion prior heavily overfits to a specific motion, it will limit the ability to perceive more diverse or unexpected motions.

1.2 Thesis Outline and Contributions

This thesis addresses several problems in motion perception and generation. In particular, we focus on developing data-driven models of motion and providing insights into the most effective ways to achieve the capabilities outlined in Sec. 1.1.1. By learning motion models for 3D human pose, 3D object shape, and 2D vehicle and pedestrian trajectories, several generative applications are enabled such as character animation, traffic and crowd simulation, and traffic scenario editing. Moreover, the proposed methods promote the analysis-by-synthesis perception paradigm by using learned models of motion to inform perception tasks like 3D pose estimation and shape reconstruction. The contributions of the thesis to the area of learned motion modeling are summarized in Tab. 1.1.

The thesis is organized into two main parts, as shown in Fig. 1.2. Chapters 2 and 3 focus on motion models for *perception* problems. Specifically, they tackle perceiving the pose of 3D humans and shape of 3D objects that are relatively isolated (*i.e.*, have few environment or multi-agent interactions). Chapters 4 and 5 look at *generative* applications of motion modeling centered around human behavior. In these works, motion takes the form of 2D vehicle and pedestrian trajectories that contain considerable interactions between agents and with the environment.

Chapter 2 introduces the HuMoR model for 3D human motion [206]. HuMoR is an expressive generative model in the form of a conditional variational autoencoder (VAE), which learns a distribution of the change in pose at each step of a motion sequence. As a latent variable model, HuMoR accounts for the uncertainty inherent to human motion and captures a diverse set of possible futures. Its architecture design mirrors that of a traditional physics-based model, enabling generalization to a

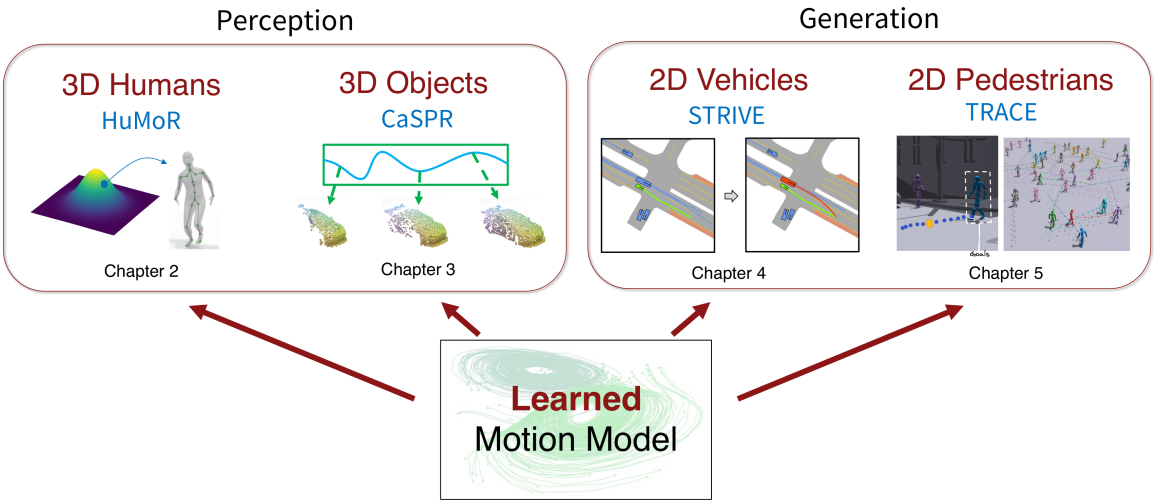

Figure 1.2: This thesis introduces several learned models of motion for solving important problems in perception and generation. The first two chapters focus on perceiving 3D humans and objects, while the latter two model vehicle and pedestrian behavior for traffic and crowd simulation using 2D trajectories.

variety of motions and body shapes after training on a large motion capture dataset. For perception, HuMoR is used as a motion prior to robustly estimate plausible 3D human pose, shape, and ground contacts from noisy and ambiguous observations like RGB videos and point clouds. This is achieved through a flexible optimization approach that *controls* motion using the HuMoR latent space to best fit the partial observations.

Humans can be treated as articulated rigid bodies for the sake of pose estimation, so HuMoR predicts human motion in the intuitive and low-dimensional pose space. However, general 3D objects may experience motion that is rigid, articulated, and even deformable. How can motion be effectively modeled for such 3D objects with no obvious state space?

Chapter 3 introduces CaSPR [207], a method to learn representations for dynamic 3D point clouds that more generally capture spatiotemporal changes in object shape. The goal of such a representation is to enable information aggregation over time and the interrogation of object state at any spatiotemporal neighborhood in the past, observed or not. CaSPR first maps an input point cloud sequence to a spatiotemporally-canonicalized object space. This canonicalization is leveraged by a continuous model of object motion built on the latent neural ordinary differential equations (ODE) framework. Using a latent ODE for motion enables learning a state space best suited for a particular object motion, and therefore is flexible to rigid and deformable objects alike. Experiments demonstrate the effectiveness of CaSPR on several perception applications including

Method	Accurate & Diverse	Robust & General	Interactions	Controllable
HuMoR (Chapter 2)	• Conditional VAE • Human pose state	• Single-step design • Latent variable model	• Ground contacts	• Latent optim
CaSPR (Chapter 3)	• Neural ODE • Latent state space	• Canonicalization	–	–
STRIVE (Chapter 4)	• Conditional VAE	• Bicycle model output	• Graph structure • Collision penalty	• Latent optim
TRACE (Chapter 5)	• Diffusion model	• Agent-centric • Classifier-free sampling	• Map feature grid	• Guidance

Table 1.1: Summary of the thesis contributions to the area of learned motion modeling. Several of the challenges outlined in Sec. 1.1.1 are addressed within each of the presented works.

shape reconstruction, camera pose estimation, continuous spatiotemporal sequence reconstruction, and correspondence estimation from irregularly or intermittently sampled point cloud observations.

In Chapters 4 and 5, learned models are used to simulate high-level human behavior, which is rich with multi-agent interactions. In these works, motion is represented as 2D trajectories – a temporal series of 2D waypoints that a human will either drive or walk through. Modeling motion as trajectories is useful for autonomous vehicles, where populating a simulation with realistic drivers and pedestrians helps to test and train self-driving systems.

Chapter 4 tackles the problem of scalably creating long-tail (*i.e.*, rare) traffic scenarios [210], which are crucial to ensuring autonomous planners are safe. We introduce a method called STRIVE to automatically generate challenging scenarios that cause a given planner to produce undesirable collision behaviors. To maintain scenario plausibility, STRIVE leverages a learned model of traffic motion in the form of a graph-based conditional VAE. The VAE operates at the *scene level*, using a graph structure to resolve *interactions* between vehicles and training with a collision penalty to encourage realistic interactions. The output of the model is an action sequence, which goes through the kinematic bicycle model to ensure realistic vehicle dynamics are predicted. Similar to HuMoR, the VAE learns a latent space that can be exploited via optimization to *control* the future trajectories of vehicles in a scenario. Scenario generation is formulated as a latent space optimization that perturbs an initial real-world scene to produce trajectories that collide with a given planner. A subsequent optimization is used to find a "solution" to the scenario, ensuring it is useful to improve the planner. Experiments show that STRIVE successfully generates a diverse set of realistic and challenging scenarios to attack, and thereby improve, two planners.

STRIVE and HuMoR demonstrate the ability to *control* the outputs of a VAE motion model

by optimizing in the latent space. However, this latent traversal can be expensive and produces deterministic results, even if there are several plausible ways to meet the desired objectives. Chapter 5 explores an alternative approach to controlling motion generation through diffusion modeling [209]. We introduce a method called TRACE for generating realistic pedestrian trajectories that can be controlled to meet user-defined goals. TRACE is a guided diffusion model that allows users to constrain trajectories through target waypoints, speed, collision avoidance, and specified social groups while accounting for the surrounding environment context. The denoising architecture employs a learned map feature grid to resolve local interactions between pedestrians and obstacles, and classifier-free training is used to ensure the model is flexible to guidance. Guidance perturbs generated trajectories as part of the denoising process at test time, such that the model does not need to be re-trained to work with new user controls. TRACE is integrated with a physics-based humanoid controller to form a closed-loop, full-body pedestrian animation system capable of placing large crowds in a simulated environment with varying terrains.

Chapter 6 concludes the thesis by summarizing the main contributions and connecting them to several future directions to continue improving learned motion models for perception and generation.

Chapter 2

3D Human Motion Models for Pose Estimation

We begin by tackling one of the most fundamental perception problems involving humans: 3D pose estimation. As introduced next, the key component is a learned model of 3D human pose transitions called HuMoR, which was originally published in ICCV 2021 [206].

2.1 Introduction

As humans, we are constantly moving in, interacting with, and manipulating the world around us. Thus, applications such as action recognition [265, 266] or holistic dynamic indoor scene understanding [39] require accurate perception of 3D human pose, shape, motion, contacts, and interaction. Extensive previous work has focused on estimating 2D or 3D human pose [27, 167, 168], shape [187, 88, 226], and motion [126] from videos. These are challenging problems due to the large space of articulations, body shape, and appearance variations. Even the best methods struggle to accurately capture a wide variety of motions from varying input modalities, producing noisy or overly-smoothed motions (especially at ground contact, *i.e.*, footskate), and struggle with occlusions (*e.g.*, walking behind a couch as in Fig. 2.1).

In this chapter, we focus on the problem of building a robust human motion model that can address these challenges. To date, most motion models directly represent sequences of likely poses — *e.g.*, in PCA space [181, 259, 235] or via future-predicting autoregressive processes [250, 258, 191]. However, purely pose-based predictions either make modeling environment interactions and generalization beyond training poses difficult, or quickly diverge from the space of realistic motions.

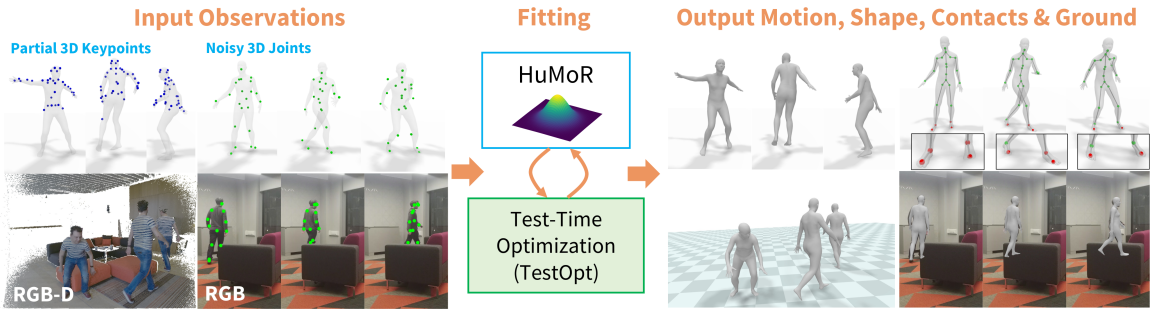

Figure 2.1: HuMoR is a 3D **Hu**man **Mo**tion model for **R**obust estimation of temporal pose formulated as a conditional variational autoencoder. (Left) The proposed approach can operate on many input modalities and is designed to handle partial and noisy observations. (Middle/Right) A test-time optimization fits 3D motion and shape to an input sequence using HuMoR as a prior; additional outputs include the ground and person-ground contacts (colored as ground plane and contacts).

On the other hand, explicit physical dynamics models [208, 145, 233, 195, 25, 24] are resource intensive and require knowledge of unobservable physical quantities. While generative models potentially offer the required flexibility, building an *expressive*, *generalizable* and *robust* model for *realistic* 3D human motions remains an open problem.

To address this, we introduce a learned, autoregressive, generative model that captures the *dynamics* of 3D human motion, *i.e.*, how pose changes over time. Rather than describing likely poses, the **Hu**man **Mo**tion Model for **R**obust Estimation (HuMoR) models a probability distribution of possible *pose transitions*, formulated as a conditional variational autoencoder [239]. Though not explicitly physics-based, its components correspond to a physical model: the latent space can be interpreted as generalized forces, which are inputs to a dynamics model with numerical integration (the decoder). Moreover, ground contacts are explicitly predicted and used to constrain pose estimation at test time.

After training on the large AMASS motion capture dataset [162], we use HuMoR as a *motion prior* at test time for 3D human perception from noisy and partial observations across different input modalities such as RGB(-D) video and 2D or 3D joint sequences, as illustrated in Fig. 2.1 (left). In particular, we introduce a robust test-time optimization strategy which interacts with HuMoR to estimate the parameters of *3D motion*, *body shape*, the *ground plane*, and *contact points* as shown in Fig. 2.1 (middle/right). This interaction happens in two ways: (i) by parameterizing the motion in the latent space of HuMoR, and (ii) using HuMoR priors in order to regularize the optimization towards the space of plausible motions.

Comprehensive evaluations reveal that our method surpasses the state-of-the-art on a variety of visual inputs in terms of accuracy and physical plausibility of motions under partial and severe

occlusions. We further demonstrate that our motion model generalizes to diverse motions and body shapes on common generative tasks like sampling and future prediction. In a nutshell, our contributions are:

- HuMoR, a generative 3D human motion prior modeled by a novel conditional VAE which enables expressive and general motion reconstruction and generation,
- A subsequent robust test-time optimization approach that uses HuMoR as a strong motion prior jointly solving for pose, body shape, and ground plane / contacts,
- The capability to operate on a variety of inputs, such as RGB(-D) video and 2D/3D joint position sequences, to yield accurate and plausible motions and contacts, exemplified through extensive evaluations.

Our work, more generally, suggests that neural nets for dynamics problems can benefit from architectures that model transitions, allowing control structures that emulate classical physical formulations.

2.2 Related Work

Much progress has been made on building methods to recover 3D joint locations [190, 168, 167] or parameterized 3D pose and shape (*i.e.*, SMPL [155]) from observations [263]. We focus primarily on motion and shape estimation.

Learning-Based Estimation. Deep learning approaches have shown success in regressing 3D shape and pose from a single image [129, 113, 188, 84, 83, 300, 41]. This has led to developments in predicting *motion* (pose sequences) and shape directly from RGB video [114, 305, 231, 247, 58]. Most recently, VIBE [126] uses adversarial training to encourage plausible outputs from a conditional recurrent motion generator. MEVA [159] maps a fixed-length image sequence to the latent space of a pre-trained motion autoencoder. These methods are fast and produce accurate root-relative joint positions for video, but motion is globally inconsistent and they struggle to generalize, *e.g.*, under severe occlusions. Other works have addressed occlusions but only on static images [17, 309, 217, 75, 127]. Our approach resolves difficult occlusions in video and other modalities by producing plausible and expressive motions with HuMoR.

Optimization-Based Estimation. One may directly optimize to more accurately fit to observations (images or 2D pose estimators [27]) using human body models [71, 8, 19]. SMPLify [19] uses the

SMPL model [155] to fit pose and shape parameters to 2D keypoints in an image using priors on pose and shape. Later works consider body silhouettes [134] and use a learned variational pose prior [187]. Optimization for motion sequences has been explored by several works [6, 108, 151, 301, 283] which apply simple smoothness priors over time. These produce reasonable estimates when the person is fully visible, but with unrealistic dynamics, *e.g.*, overly smooth motions and footskate.

Some works employ human-environment interaction and contact constraints to improve shape and pose estimation [93, 151, 94] by assuming scene geometry is given. iMapper [172] recovers both 3D joints and a primitive scene representation from RGB video based on interactions by motion retrieval, which may differ from observations. In contrast, our approach optimizes for pose and shape by using an expressive generative model that produces more natural motions than prior work with realistic ground contact.

Human Motion Models. Early sophisticated motion models for pose tracking used a variety of approaches, including mixtures-of-Gaussians [106], linear embeddings of periodic motion [181, 259, 235], nonlinear embeddings [62], and nonlinear autoregressive models [250, 274, 258, 191]. These methods operate in pose space, and are limited to specific motions. Models based on physics can potentially generalize more accurately [208, 145, 233, 195, 25, 24, 297], while also estimating global pose and environmental interactions. However, general-purpose physics-based models are difficult to learn, computationally intensive at test-time, and often assume full-body visibility to detect contacts [208, 145, 233].

Many motion models have been learned for computer animation [21, 132, 219, 142, 148, 103, 242] including recent recurrent and autoregressive models [89, 77, 98, 285, 147]. These often focus on visual fidelity for a small set of characters and periodic locomotions. Some have explored generating more general motion and body shapes [311, 189, 3, 46], but in the context of short-term future prediction. HuMoR is most similar to Motion VAE [147], however we make crucial contributions to enable generalization to unseen, non-periodic motions on novel body shapes.

2.3 HuMoR: 3D Human Dynamics Model

The goal of this work is to build an *expressive* and *generalizable* generative model of 3D human motion learned from real human motions, and to show that this can be used for robust test-time optimization (TestOpt) of pose and shape. In this section, we first describe the model, HuMoR.

State Representation. We represent the state of a moving person as a matrix $\mathbf{x}$ composed of a root translation $\mathbf{r} \in \mathbb{R}^3$, root orientation $\Phi \in \mathbb{R}^3$ in axis-angle form, body pose joint angles $\Theta \in \mathbb{R}^{3\times 21}$

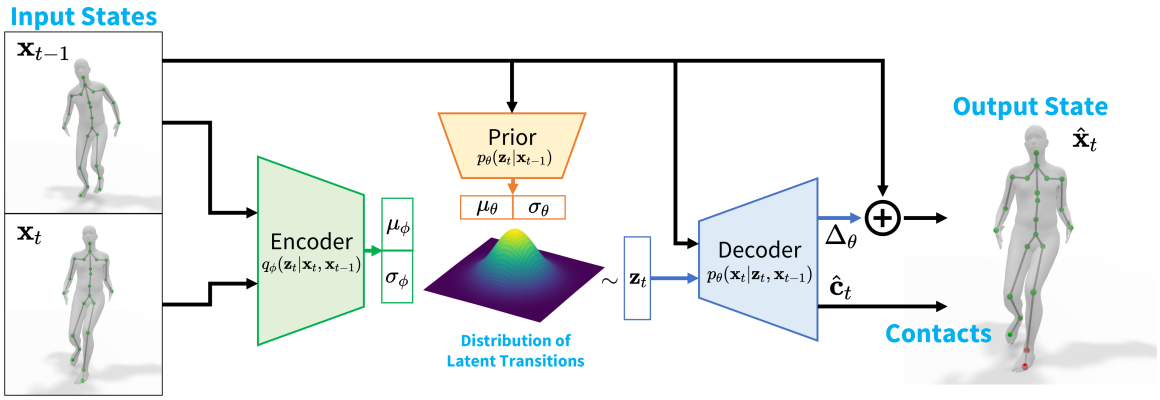

Figure 2.2: The HuMoR CVAE architecture is shown. During training, given the previous state $\mathbf{x}_{t-1}$ and ground truth current state $\mathbf{x}_t$, the model reconstructs $\hat{\mathbf{x}}_t$ by sampling from the **encoder** distribution. At test time we can (i) *generate* the next state from $\mathbf{x}_{t-1}$ by sampling from the **prior** distribution and **decoding**, (ii) *infer* a latent transition $\mathbf{z}_t$ with the **encoder**, or (iii) evaluate the *likelihood* of a given $\mathbf{z}_t$ with the **conditional prior**.

and joint positions $\mathbf{J} \in \mathbb{R}^{3\times 22}$:

$$\mathbf{x} = [\quad \mathbf{r} \quad \dot{\mathbf{r}} \quad \Phi \quad \dot{\Phi} \quad \Theta \quad \mathbf{J} \quad \dot{\mathbf{J}} \quad], \tag{2.1}$$

where $\dot{\mathbf{r}}, \dot{\Phi}$ and $\dot{\mathbf{J}}$ denote the root and joint velocities, respectively, giving $\mathbf{x} \in \mathbb{R}^{3\times 69}$. Part of the state, $(\mathbf{r}, \Phi, \Theta)$, parameterizes the SMPL body model [155, 218] which is a differentiable function $M(\mathbf{r}, \Phi, \Theta, \beta)$ that maps to body mesh vertices $\mathbf{V} \in \mathbb{R}^{3\times 6890}$ and joints $\mathbf{J}^{\text{SMPL}} \in \mathbb{R}^{3\times 22}$ given shape parameters $\beta \in \mathbb{R}^{16}$. Our over-parameterization allows for two ways to recover the joints: (i) explicitly from $\mathbf{J}$, (ii) implicitly through the SMPL map $M(\cdot)$.

Latent Variable Dynamics Model. We are interested in modeling the probability of a time sequence of states

$$p_\theta(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) = p_\theta(\mathbf{x}_0) \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{2.2}$$

where each state is assumed to be dependent on only the previous one and θ are learned parameters. Then $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ must capture the *plausibility* of a transition.

We propose a **conditional variational autoencoder (CVAE)** which formulates the motion $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ as a latent variable model as shown in Fig. 2.2. Following the original CVAE derivation [239], our model contains two main components. First, conditioned on the previous state $\mathbf{x}_{t-1}$,

the distribution over possible latent variables $\mathbf{z}_t \in \mathbb{R}^{48}$ is described by a learned **conditional prior**:

$$p_\theta(\mathbf{z}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \mu_\theta(\mathbf{x}_{t-1}), \sigma_\theta(\mathbf{x}_{t-1}))\,, \tag{2.3}$$

which parameterizes a Gaussian distribution with diagonal covariance via a neural network. Intuitively, the latent variable $\mathbf{z}_t$ represents the transition to $\mathbf{x}_t$ and should therefore have different distributions given different $\mathbf{x}_{t-1}$. For example, an idle person has a large variation of possible next states while a person in midair is on a nearly deterministic trajectory. Learning the conditional prior, rather than assuming $p_\theta(\mathbf{z}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \mathbf{0}, \mathbf{I})$, significantly improves the ability of the CVAE to generalize to diverse motions and empirically stabilizes both training and TestOpt.

Second, conditioned on $\mathbf{z}_t$ and $\mathbf{x}_{t-1}$, the **decoder** produces two outputs, Δ_θ and $\mathbf{c}_t$. The *change in state* Δ_θ defines the output distribution $p_\theta(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1})$ through

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta_\theta(\mathbf{z}_t, \mathbf{x}_{t-1}) + \boldsymbol{\eta}, \qquad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.4}$$

We find the additive update Δ_θ improves predictive accuracy compared to direct next-step prediction. The person-ground contact $\mathbf{c}_t$ is the probability that each of 8 body joints (*left and right toes*, *heels*, *knees*, and *hands*) is in contact with the ground at time t. Contacts are *not* part of the input to the conditional prior, only an output of the decoder. The contacts enable environmental constraints in TestOpt.

The complete probability model for a transition is then:

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}) = \int_{\mathbf{z}_t} p_\theta(\mathbf{z}_t|\mathbf{x}_{t-1}) p_\theta(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1}). \tag{2.5}$$

Given an initial state $\mathbf{x}_0$, one can sample a motion sequence by alternating between sampling $\mathbf{z}_t \sim p_\theta(\mathbf{z}_t|\mathbf{x}_{t-1})$ and sampling $\mathbf{x}_t \sim p_\theta(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1})$, from $t = 1$ to T. This model parallels a conventional stochastic physical model. The conditional prior can be seen as a controller, producing "forces" $\mathbf{z}_t$ as a function of state $\mathbf{x}_{t-1}$, while the decoder acts like a combined physical dynamics model and Euler integrator of generalized position and velocity in Eq. (2.4).

In addition to this nice physical interpretation, our model is motivated by Motion VAE (MVAE) [147], which has recently shown promising results for single-character locomotion animation, also using a VAE for $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$. However, we find that directly applying MVAE for estimation does not give good results (Sec. 2.5). We overcome this by additionally learning a *conditional* prior, modeling the *change in state* and contacts, and encouraging *consistency* between joint position and angle

predictions (Sec. 2.3.1).

Rollout. We use our model to define a deterministic *rollout function*, which is key to TestOpt. Given an initial state $\mathbf{x}_0$ and a sequence of latent transitions $\mathbf{z}_{1:T}$, we define a function $\mathbf{x}_T = f(\mathbf{x}_0, \mathbf{z}_{1:T})$ that deterministically maps the motion "parameters" $(\mathbf{x}_0, \mathbf{z}_{1:T})$ to the resulting state at time T. This is done through autoregressive rollout which decodes and integrates $\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta_\theta(\mathbf{z}_t, \mathbf{x}_{t-1})$ at each timestep.

Initial State GMM. We model $p_\theta(\mathbf{x}_0)$ with a Gaussian mixture model (GMM) containing $K = 12$ components with weights γ^i, so that $p_\theta(\mathbf{x}_0) = \sum_{i=1}^{K} \gamma^i \mathcal{N}(\mathbf{x}_0; \mu_\theta^i, \sigma_\theta^i)$.

2.3.1 Training

Our CVAE is trained using pairs of $(\mathbf{x}_{t-1}, \mathbf{x}_t)$. We consider the usual variational lower bound:

$$\log p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}) \geq \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1})] - D_{\text{KL}}(q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{x}_{t-1}) \parallel p_\theta(\mathbf{z}_t|\mathbf{x}_{t-1})). \tag{2.6}$$

The expectation term measures the reconstruction error of the **decoder**. The **encoder**, *i.e.* approximate posterior, is introduced for training and parameterizes a Gaussian distribution $q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \mu_\phi(\mathbf{x}_t, \mathbf{x}_{t-1}), \sigma_\phi(\mathbf{x}_t, \mathbf{x}_{t-1}))$. The KL divergence $D_{\text{KL}}(\cdot \parallel \cdot)$ regularizes its output to be near the **prior**. Therefore, we seek the parameters (θ, ϕ) that minimize the loss function

$$\mathcal{L}_{\text{rec}} + w_{\text{KL}}\mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{reg}} \tag{2.7}$$

over all training pairs in our dataset, where $\mathcal{L}_{\text{rec}} + w_{\text{KL}}\mathcal{L}_{\text{KL}}$ is the lower bound in Eq. (2.6) with weight w_{KL}, and $\mathcal{L}_{\text{reg}}$ contains additional regularizers.

For a single training pair $(\mathbf{x}_{t-1}, \mathbf{x}_t)$, the reconstruction loss is computed as $\mathcal{L}_{\text{rec}} = ||\mathbf{x}_t - \hat{\mathbf{x}}_t||^2$ from the decoder output $\hat{\mathbf{x}}_t = \mathbf{x}_{t-1} + \Delta_\theta(\mathbf{z}_t, \mathbf{x}_{t-1})$ with $\mathbf{z}_t \sim q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{x}_{t-1})$. Gradients are backpropagated through this sample using the reparameterization trick [122]. The regularization loss contains two terms: $\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{SMPL}} + w_{\text{contact}}\mathcal{L}_{\text{contact}}$. The SMPL term $\mathcal{L}_{\text{SMPL}} = \mathcal{L}_{\text{joint}} + \mathcal{L}_{\text{vtx}} + \mathcal{L}_{\text{consist}}$ uses the output of the body model with the estimated parameters and ground truth shape $[\hat{\mathbf{J}}_t^{\text{SMPL}}, \hat{\mathbf{V}}_t] = M(\hat{\mathbf{r}}_t, \hat{\Phi}_t, \hat{\Theta}_t, \beta)$:

$$\mathcal{L}_{\text{joint}} = ||\mathbf{J}_t^{\text{SMPL}} - \hat{\mathbf{J}}_t^{\text{SMPL}}||^2 \tag{2.8}$$

$$\mathcal{L}_{\text{vtx}} = ||\mathbf{V}_t - \hat{\mathbf{V}}_t||^2 \tag{2.9}$$

$$\mathcal{L}_{\text{consist}} = ||\hat{\mathbf{J}}_t - \hat{\mathbf{J}}_t^{\text{SMPL}}||^2. \tag{2.10}$$

The loss $\mathcal{L}_{\text{consist}}$ encourages consistency between regressed joints and those of the body model. The contact loss $\mathcal{L}_{\text{contact}} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{vel}}$ contains two terms. The first supervises ground contact classification with a typical binary cross entropy; the second regularizes joint velocities to be consistent with contacts $\mathcal{L}_{\text{vel}} = \sum_j \hat{c}_t^j ||\hat{\mathbf{v}}_t||^2$ with $\hat{\mathbf{v}}_t \in \hat{\dot{\mathbf{J}}}_t$ and $\hat{c}_t^j \in \hat{\mathbf{c}}_t$ the predicted probability that joint j is in ground contact. We set $w_{\text{contact}} = 0.01$ and $w_{\text{KL}} = 4e^{-4}$.

The initial state GMM is trained separately with expectation-maximization on the same dataset used to train the CVAE.

Implementation Details. To ease learning and improve generalization, our model operates in an aligned canonical coordinate frame at each step. Specifically, based on $\mathbf{x}_{t-1}$ we apply a rotation around the up $(+z)$ axis and translation in x, y such that the x and y components of $\mathbf{r}_{t-1}$ are 0 and the person's body right axis (w.r.t. Φ_{t-1}) is facing the $+x$ direction. All networks are 4 or 5 layer MLPs with ReLU activations and group normalization [282]. The latent transition $\mathbf{z}_t \in \mathbb{R}^{48}$ is skip-connected to every layer of the decoder in order to emphasize its importance and help avoid posterior collapse [147].

A common difficulty in training VAEs is posterior collapse [158] – when the learned latent encoding $\mathbf{z}_t$ is effectively ignored by the decoder. This problem is exacerbated in CVAEs since the decoder receives additional conditioning [147, 239]. To combat posterior collapse [158, 147, 239], we linearly anneal w_{KL} during training [20]. Following [147], we also use scheduled sampling [12] to enable long-term generation by making the model robust to its own errors.

2.4 Test-time Motion Optimization

We next use the space of motion learned by HuMoR as a *prior* in TestOpt to recover pose and shape from noisy and partial observations while ensuring plausibility.

2.4.1 Optimization Variables

Given a sequence of observations $\mathbf{y}_{0:T}$, either as 2D/3D joints, 3D point clouds, or 3D keypoints, we seek the shape β and a sequence of SMPL pose parameters $(\mathbf{r}_{0:T}, \Phi_{0:T}, \Theta_{0:T})$ which describe the underlying motion being observed. We parameterize the optimized motion **using our CVAE** by the initial state $\mathbf{x}_0$ and a sequence of latent transitions $\mathbf{z}_{1:T}$. Then at T (and any intermediate steps) $\mathbf{x}_T = f(\mathbf{x}_0, \mathbf{z}_{1:T})$ is determined through model *rollout* using the **decoder** as previously detailed. Compared to directly optimizing SMPL [6, 19, 108], this motion representation naturally encourages plausibility and is compact in the number of variables. To obtain the transformation

between the canonical coordinate frame in which our CVAE is trained and the observation frame used for optimization, we additionally optimize the ground plane of the scene $\mathbf{g} \in \mathbb{R}^3$ with $\mathbf{g} = d\hat{\mathbf{n}}$ where $\hat{\mathbf{n}}$ is the ground unit normal vector and d the plane offset. All together, we simultaneously optimize initial state $\mathbf{x}_0$, a sequence of latent variables $\mathbf{z}_{1:T}$, ground $\mathbf{g}$, and shape β. We assume a static camera with known intrinsics.

Observation-to-Canonical Transformation. We assume that gravity is orthogonal to the ground plane. Therefore, given the current floor $\mathbf{g}$ and root state $\mathbf{r}, \Phi$ (in the observation frame) we compute a rotation and translation to the canonical CVAE frame: after the transformation, $\hat{\mathbf{n}}$ is aligned with $+z$ and $d = 0$, Φ faces body right towards $+x$, and the x, y components of $\mathbf{r}$ are 0. With this ability, we can always compute the (observed) state at time $\mathbf{x}_t$ from $\mathbf{z}_{1:t}$, $\mathbf{x}_0$, and $\mathbf{g}$ by (i) transforming $\mathbf{x}_0$ to the canonical frame, (ii) using the CVAE to rollout $\mathbf{x}_t = f(\mathbf{x}_0, \mathbf{z}_{1:t})$, and (iii) transforming $\mathbf{x}_t$ back to the observation frame.

2.4.2 Objective & Optimization

The optimization objective can be formulated as a maximum a-posteriori (MAP) estimate, which seeks a motion that is plausible under our generative model while closely matching observations:

$$\min_{\mathbf{x}_0, \mathbf{z}_{1:T}, \mathbf{g}, \beta} \mathcal{E}_{\text{mot}} + \mathcal{E}_{\text{data}} + \mathcal{E}_{\text{reg}}. \tag{2.11}$$

We next detail each of these terms which are the motion prior, data, and regularization energies. In the following, λ are weights to determine the contribution of each term.

Motion Prior $\mathcal{E}_{\text{mot}}$. This energy measures the likelihood of the latent transitions $\mathbf{z}_{1:T}$ and initial state $\mathbf{x}_0$ under the HuMoR CVAE and GMM. It is $\mathcal{E}_{\text{mot}} = \mathcal{E}_{\text{CVAE}} + \mathcal{E}_{\text{init}}$ where

$$\mathcal{E}_{\text{CVAE}} = -\lambda_{\text{CVAE}} \sum_{t=1}^{T} \log \mathcal{N}(\mathbf{z}_t; \mu_\theta(\mathbf{x}_{t-1}), \sigma_\theta(\mathbf{x}_{t-1})) \tag{2.12}$$

$$\mathcal{E}_{\text{init}} = -\lambda_{\text{init}} \log \sum_{i=1}^{K} \gamma^i \mathcal{N}(\mathbf{x}_0; \mu_\theta^i, \sigma_\theta^i). \tag{2.13}$$

$\mathcal{E}_{\text{CVAE}}$ uses the learned **conditional prior** and $\mathcal{E}_{\text{init}}$ uses the initial state GMM.

Data Term $\mathcal{E}_{\text{data}}$. This term is the *only* modality-dependent component of our approach, requiring different losses for different inputs: 3D joints, 2D joints, and 3D point clouds. All data losses operate on SMPL joints or mesh vertices obtained through the body model $[\mathbf{J}_t^{\text{SMPL}}, \mathbf{V}_t] = M(\mathbf{r}_t, \Phi_t, \Theta_t, \beta)$

using the current shape β along with the SMPL parameters $(\mathbf{r}_t, \Phi_t, \Theta_t)$ contained in $\mathbf{x}_t = f(\mathbf{x}_0, \mathbf{z}_{1:t})$ transformed from the canonical to observation (*i.e.* camera) frame. In the simplest case, the observations $\mathbf{y}_t$ are 3D joint positions (or keypoints with known correspondences) and our energy is

$$\mathcal{E}_{\text{data}} \triangleq \mathcal{E}^{\text{3D}}_{\text{data}} = \lambda_{\text{data}} \sum_{t=0}^{T} \sum_{j=1}^{J} ||\mathbf{p}_t^j - \mathbf{y}_t^j||^2 \tag{2.14}$$

with $\mathbf{p}_t^j \in \mathbf{J}_t^{\text{SMPL}}$. For 2D joint positions, each with a detection confidence σ_t^j, we use a re-projection loss

$$\mathcal{E}_{\text{data}} \triangleq \mathcal{E}^{\text{2D}}_{\text{data}} = \lambda_{\text{data}} \sum_{t=0}^{T} \sum_{j=1}^{J} \sigma_t^j \rho(\Pi(\mathbf{p}_t^j) - \mathbf{y}_t^j) \tag{2.15}$$

with ρ the robust Geman-McClure function [19, 73] and Π the pinhole projection. If an estimated person segmentation mask is available, it is used to ignore spurious 2D joints. Finally, if $\mathbf{y}_t$ is a 3D point cloud obtained from a depth map roughly masked around the person of interest, we use the mesh vertices to compute

$$\mathcal{E}_{\text{data}} \triangleq \mathcal{E}^{\text{PC3D}}_{\text{data}} = \lambda_{\text{data}} \sum_{t=0}^{T} \sum_{i=1}^{N_t} w_{\text{bs}} \min_{\mathbf{p}_t \in \mathbf{V}_t} ||\mathbf{p}_t - \mathbf{y}_t^i||^2 \tag{2.16}$$

where w_{bs} is a robust bisquare weight [10] computed based on the Chamfer distance term.

Regularizers $\mathcal{E}_{\text{reg}}$. The additional regularization consists of four terms $\mathcal{E}_{\text{reg}} = \mathcal{E}_{\text{skel}} + \mathcal{E}_{\text{env}} + \mathcal{E}_{\text{gnd}} + \mathcal{E}_{\text{shape}}$. The first two terms encourage rolled-out motions from the CVAE to be plausible even when the initial state $\mathbf{x}_0$ is far from the optimum (*i.e.* early in optimization). The skeleton consistency term uses the joints $\mathbf{J}_t$ directly *predicted* by the decoder during rollout along with the SMPL joints:

$$\mathcal{E}_{\text{skel}} = \sum_{t=1}^{T} \left(\lambda_{\text{c}} \sum_{j=1}^{J} ||\mathbf{p}_t^j - \mathbf{p}_t^{j,\text{pred}}||^2 + \lambda_{\text{b}} \sum_{i=1}^{B} (l_t^i - l_{t-1}^i)^2 \right) \tag{2.17}$$

with $\mathbf{p}_t^j \in \mathbf{J}_t^{\text{SMPL}}$ and $\mathbf{p}_t^{j,\text{pred}} \in \mathbf{J}_t$. The second summation uses bone lengths l computed from $\mathbf{J}_t$ at each step. The second regularizer $\mathcal{E}_{\text{env}}$ ensures consistency between predicted CVAE contacts, the motion, and the environment:

$$\mathcal{E}_{\text{env}} = \sum_{t=1}^{T} \sum_{j=1}^{J} \lambda_{\text{cv}} c_t^j ||\mathbf{p}_t^j - \mathbf{p}_{t-1}^j||^2 + \lambda_{\text{ch}} c_t^j \max(|\mathbf{p}_{z,t}^j| - \delta, 0) \tag{2.18}$$

where $\mathbf{p}_t^j \in \mathbf{J}_t^{\text{SMPL}}$ and c_t^j is the contact probability output from the model for joint j. The contact height term weighted by λ_{ch} ensures the z-component of contacting joints are within δ of the floor in the canonical frame.

The final two regularizers are priors on the ground and shape. We assume the ground should stay close to initialization $\mathcal{E}_{\text{gnd}} = \lambda_{\text{gnd}}||\mathbf{g} - \mathbf{g}^{\text{init}}||^2$. Finally, β should stay near the neutral zero vector similar to [93, 187]: $\mathcal{E}_{\text{shape}} = \lambda_{\text{shape}}||\beta||^2$.

Initialization & Optimization. We initialize the temporal SMPL parameters $\mathbf{r}_{0:T}, \Phi_{0:T}, \Theta_{0:T}$ and shape β with an initialization optimization using $\mathcal{E}_{\text{data}}$ and $\mathcal{E}_{\text{shape}}$ along with two additional regularization terms. $\mathcal{E}_{\text{pose}} = \sum_t ||\mathbf{z}_t^{\text{pose}}||^2$ is a pose prior where $\mathbf{z}_t^{\text{pose}} \in \mathbb{R}^{32}$ is the body joint angles represented in the latent space of the VPoser model [187, 93]. The smoothness term $\mathcal{E}_{\text{smooth}} = \sum_{t=1}^{T}\sum_{j=1}^{J}||\mathbf{p}_t^j - \mathbf{p}_{t-1}^j||^2$ with $\mathbf{p}_t^j \in \mathbf{J}_t^{\text{SMPL}}$ smooths 3D joint positions over time. Afterwards, the initial latent sequence $\mathbf{z}_{1:T}^{\text{init}}$ is computed through inference with the CVAE **encoder**. Our optimization is implemented in PyTorch [186] using L-BFGS and *autograd*; with batching, a $3s$ RGB video takes about 5.5 min to fit. For all experiments, we optimize using the neutral SMPL+H [218] body model in 3 stages. First, only the initial state $\mathbf{x}_0$ and first 15 frames of the latent sequence $\mathbf{z}_{1:15}$ are optimized for 30 iterations in order to quickly reach a reasonable initial state. Next, $\mathbf{x}_0$ is fixed while the full latent dynamics sequence $\mathbf{z}_{1:T}$ is optimized for 25 iterations, and then finally the full sequence and initial state are tuned together for another 15 iterations. The ground $\mathbf{g}$ and shape β are optimized in every stage.

2.5 Experimental Results

We evaluate HuMoR on (i) generative sampling tasks and (ii) as a prior in TestOpt to estimate motion from 3D and RGB(-D) inputs. We encourage viewing the **supplementary videos** to appreciate the qualitative improvement of our approach.

2.5.1 Datasets

AMASS [162] is a large motion capture database containing diverse motions and body shapes on the SMPL body model. We sub-sample the dataset to 30 Hz and use the recommended training split to train the CVAE and initial state GMM in HuMoR. We evaluate on the held out Transitions and HumanEva [236] subsets (Sec. 2.5.3 and 2.5.4).

i3DB [172] contains RGB videos of person-scene interactions involving medium to heavy occlusions. It provides annotated 3D joint positions and a primitive 3D scene reconstruction which we use to fit a ground plane for computing plausibility metrics. We run off-the-shelf 2D pose estimation [27], person segmentation [36], and plane detection [149] models to obtain inputs for our optimization.

PROX [93] contains RGB-D videos of people interacting with indoor environments. We use a subset of the qualitative data to evaluate plausibility metrics using a floor plane fit to the provided ground truth scene mesh. We obtain 2D pose, person masks, and ground plane initialization in the same way as done for i3DB. We evaluate on all videos from 4 chosen scenes (N3Office, N3Library, N0Sofa, and MPH1Library) that tend to have more dynamic motions and occlusions.

2.5.2 Baselines and Evaluation Metrics

Motion Prior Baselines. We ablate the proposed CVAE to analyze its core components: *No Delta* directly predicts the next state from the decoder rather than the change in state, *No Contacts* does not classify ground contacts, *No* $\mathcal{L}_{SMPL}$ does not use SMPL regularization in training, and *Standard Prior* uses $\mathcal{N}(\mathbf{0}, \mathbf{I})$ rather than our learned **conditional prior**. All of these ablated together recovers *MVAE* [147].

Motion Estimation Baselines. *VPoser-t* is the initialization phase of our optimization. It uses VPoser [187] and 3D joint smoothing similar to previous works [6, 108, 301]. *PROX-(RGB/D)* [93] are optimization-based methods which operate on individual frames of RGB and RGB-D videos, respectively. Both assume the full scene mesh is given to enforce contact and penetration constraints. *VIBE* [126] is a recent learned method to recover shape and pose from video.

Error Metrics. 3D positional errors are measured on joints, keypoints, or mesh vertices (**Vtx**) and compute *global* mean per-point position error unless otherwise specified. We report positional errors for all (**All**), occluded (**Occ**), and visible (**Vis**) observations separately. Finally, we report binary classification accuracy of the 8 person-ground contacts (**Contact**) predicted by HuMoR.

Plausibility Metrics. We use additional metrics to measure qualitative motion characteristics that joint errors cannot capture. Smoothness is evaluated by mean per-joint accelerations (**Accel**) [114]. Another important indicator of plausibility is ground penetration [208]. We use the true ground plane to compute the frequency (**Freq**) of foot-floor penetrations: the fraction of frames for both the left and right toe joints that penetrate more than a threshold. We measure frequency at 0, 3, 6, 9, 12, and 15 cm thresholds and report the mean. We also report mean penetration distance (**Dist**), where non-penetrating frames contribute a distance of 0 to make values comparable across differing

	Future Prediction			Diversity
Model	**Contact ↑**	**ADE ↓**	**FDE ↓**	**APD ↑**
MVAE [147]	-	25.8	50.6	85.4
HuMoR	0.88	**21.5**	**42.1**	94.9
HuMoR (Qual)	0.88	22.0	46.3	**100.0**

Table 2.1: (Left) Future prediction accuracy for $2s$ AMASS sequences. Contact classification accuracy, average displacement error (cm), and final displacement error (cm) are reported. (Right) Sampling diversity over $5s$ rollouts measured by average pairwise distance (cm).

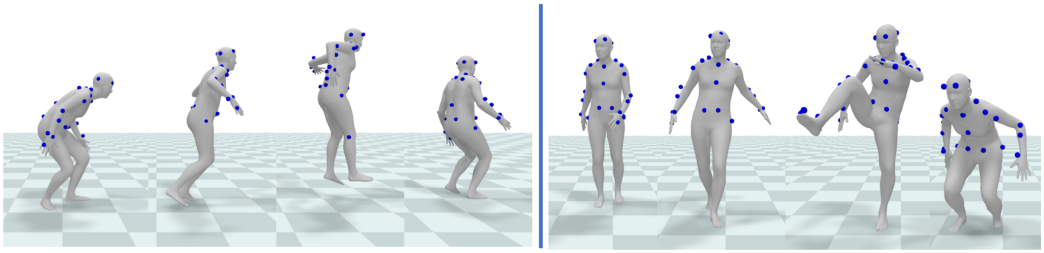

Figure 2.3: Fitting to partial 3D keypoints. HuMoR captures non-periodic motions like jumping, crouching, and kicking.

frequencies.

2.5.3 Generative Model Evaluation

We first evaluate HuMoR as a standalone generative model and show improved generalization to unseen motions and bodies compared to *MVAE* for two common tasks (see Table 2.1): future prediction and diverse sampling. We use $2s$ AMASS sequences and start generation from the first step. Results are shown for *HuMoR* and a modified *HuMoR (Qual)* that uses $\mathbf{J}^{\text{SMPL}}$ as input to each step during rollout instead of $\mathbf{J}$, thereby enforcing skeleton consistency. This version produces *qualitatively* superior results for generation, but is too expensive to use during TestOpt.

For prediction, we report average displacement error (**ADE**) and final displacement error (**FDE**) [295], which measure mean joint errors over all steps and at the final step, respectively. We sample 50 $2s$ motions for each initial state and the one with lowest **ADE** is considered the prediction. For diversity, we sample 50 $5s$ motions and compute the average pairwise distance (**APD**) [5], *i.e.* the mean joint distance between all pairs of samples.

As seen in Tab. 2.1, the base MVAE [147] does not generalize well when trained on the large AMASS dataset; our proposed CVAE improves both the accuracy and diversity of samples. *HuMoR (Qual)* hinders prediction accuracy, but gives better diversity and visual quality.

		Positional Error			Joints	Mesh			Ground Pen	
Method	**Input**	**Vis**	**Occ**	**All**	**Legs**	**Vtx**	**Contact**	**Accel**	**Freq**	**Dist**
VPoser-t	Occ Keypoints	**0.67**	20.76	9.22	21.08	7.95	-	5.71	16.77%	2.28
MVAE [147]	Occ Keypoints	2.39	19.15	9.52	16.86	8.90	-	7.12	**3.15%**	0.30
HuMoR (Ours)	Occ Keypoints	1.46	**17.40**	**8.24**	**15.42**	**7.56**	0.89	**5.38**	3.31%	**0.26**
VPoser-t	Noisy Joints	-	-	3.67	4.47	4.98	-	**4.61**	1.35%	0.07
MVAE [147]	Noisy Joints	-	-	2.68	3.21	4.42	-	6.5	1.75%	0.11
HuMoR (Ours)	Noisy Joints	-	-	**2.27**	**2.61**	**3.55**	0.97	5.23	**1.18%**	**0.05**

Table 2.2: Motion and shape estimation from 3D observations: partially occluded keypoints (top) and noisy joints (bottom). *Positional Error* (cm) is reported w.r.t. the input modality. Acceleration is m/s^2 and penetration distance in cm.

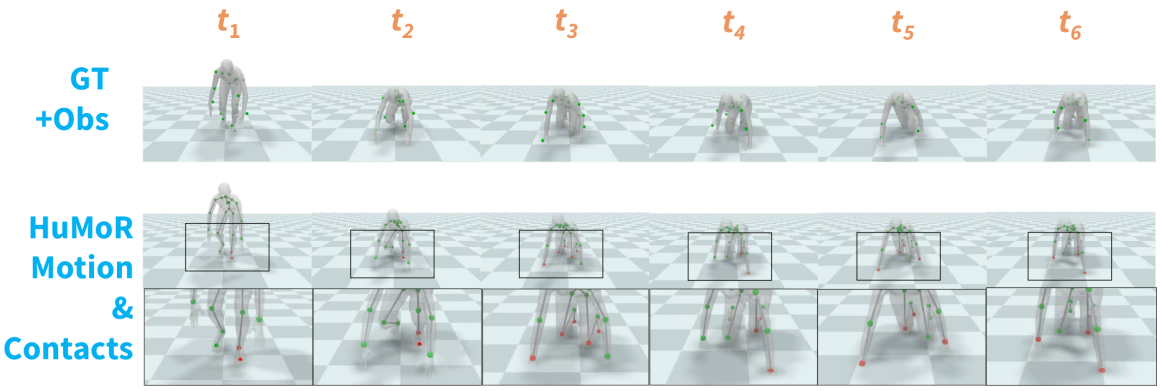

Figure 2.4: TestOpt with HuMoR recovers complex contact patterns involving feet, knees, and hands from noisy 3D joint observations (shown on top along with the ground truth motion).

2.5.4 Estimation from 3D Observations

Next, we show that HuMoR also generalizes better when used in TestOpt for fitting to 3D data, and that using a motion prior is crucial to plausibly handling occlusions. $3s$ AMASS sequences are used to demonstrate key abilities: (i) fitting to partial data and (ii) denoising. For the former, TestOpt fits to 43 keypoints on the body that resemble motion capture markers; keypoints that fall below $0.9m$ at each timestep are "occluded", leaving the legs unobservable at most steps. For denoising, Gaussian noise with $4cm$ standard deviation is added to 3D joint position observations.

Tab. 2.2 compares to *VPoser-t* and to using *MVAE* as the motion prior during optimization rather than HuMoR. We report leg joint errors (toes, ankles, and knees), which are often occluded, separately. The right side of the table reports plausibility metrics. HuMoR gives more accurate poses, especially for occluded keypoints and leg joints. It also estimates smoother motions with fewer and less severe ground penetrations. For denoising, *VPoser-t* oversmooths which gives the lowest acceleration but least accurate motion. TestOpt with HuMoR gives inherently smooth results

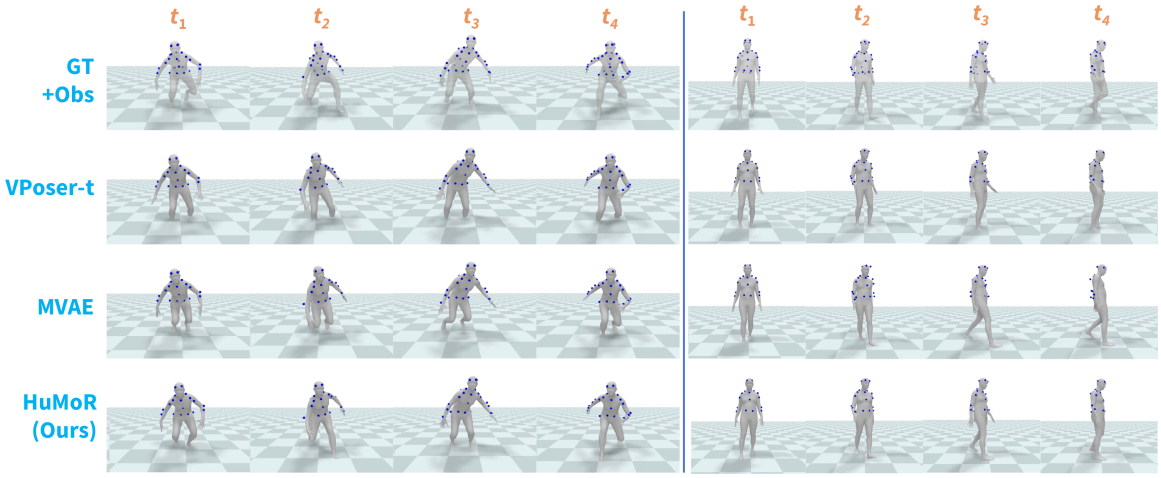

Figure 2.5: Baseline comparison when fitting to partial 3D keypoints. *GT+Obs* shows the ground truth body motion and observed keypoints in blue, while each method output shows the predicted motion with the observed keypoints for reference.

	Global Joint Error				Root-Aligned Joint Error					Ground Pen	
Method	**Vis**	**Occ**	**All**	**Legs**	**Vis**	**Occ**	**All**	**Legs**	**Accel**	**Freq**	**Dist**
VIBE [126]	90.05	192.55	116.46	121.61	12.06	23.78	15.08	21.65	243.36	7.98%	3.01
VPoser-t	28.33	40.97	31.59	35.06	12.77	26.48	16.31	25.60	4.46	9.28%	2.42
MVAE [147]	37.54	50.63	40.91	44.42	16.00	28.32	19.17	26.63	4.96	7.43%	1.55
No Delta	27.55	35.59	29.62	32.14	**11.92**	23.10	14.80	21.65	3.05	2.84%	**0.58**
No Contacts	26.65	39.21	29.89	35.73	12.24	23.36	15.11	22.25	**2.43**	5.59%	1.70
No $\mathcal{L}_{\text{SMPL}}$	31.09	43.67	34.33	36.84	12.81	25.47	16.07	23.54	3.21	4.12%	1.31
Standard Prior	77.60	146.76	95.42	99.01	18.67	39.40	24.01	34.02	5.98	8.30%	6.47
HuMoR (Ours)	**26.00**	**34.36**	**28.15**	**31.26**	12.02	**21.70**	**14.51**	**20.74**	**2.43**	**2.12%**	0.68

Table 2.3: Motion and shape from RGB video (*i.e.* 2D joints) on i3DB [172]. Joint errors are in cm and acceleration is m/s^2. Top shows results from motion estimation baselines while bottom uses ablations of HuMoR during optimization.

while still allowing for necessarily large accelerations to fit dynamic observations. Notably, HuMoR predicts person-ground contact with 97% accuracy even under severe noise.

Qualitative results for both tasks are shown in Fig. 2.1 and Fig. 2.3. Fig. 2.4 show denoising results using our method for a crawling sequence. Note that HuMoR recovers complex contact patterns involving not only the feet, but also hands and knees. A qualitative comparison to baselines for fitting to partial keypoints is shown in Fig. 2.5. *VPoser-t* fails to produce any plausible lower-body motion since it uses only a pose prior, while using *MVAE* as the motion prior often gives unnatural and implausible motions that do not align well with the observed keypoints.

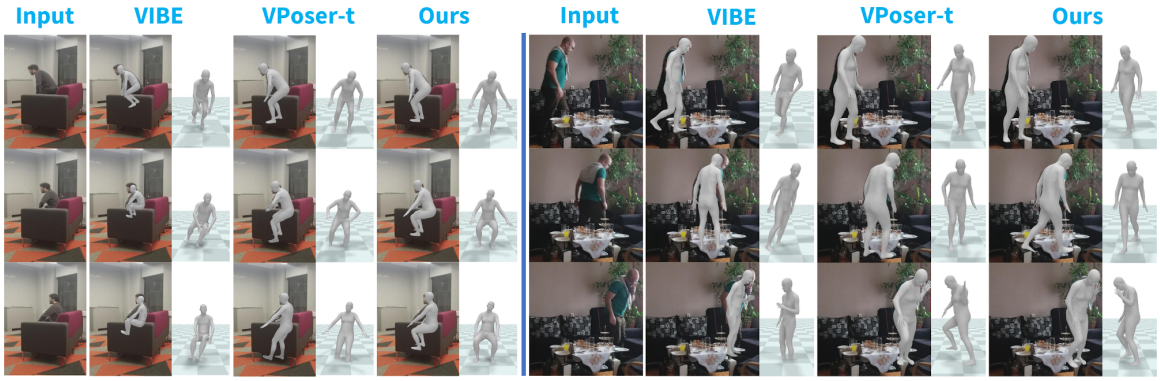

Figure 2.6: Qualitative comparison for fitting to RGB video (*i.e.* 2D joints) from i3DB [172]. Optimization using HuMoR (Ours) outputs natural and plausible sitting and walking motions under heavy occlusions compared to baseline approaches.

2.5.5 Estimation from RGB(-D) Observations

Finally, we show that TestOpt with HuMoR can be applied to real-world RGB and RGB-D observations, and outperforms baselines on positional and plausibility metrics especially from partial and noisy data. We use $3s$ (90 frame) clips from i3DB [172] and PROX [93]. Tab. 2.3 shows results on i3DB which affords quantitative 3D joint evaluation. The top half compares to baseline estimation methods; the bottom uses ablations of HuMoR in TestOpt rather than the full model. Mean per-joint position errors are reported for **global** joint positions and after root alignment.

As seen in Tab. 2.3, *VIBE* gives locally accurate predictions for visible joints, but large global errors and unrealistic accelerations due to occlusions and temporal inconsistency (see Fig. 2.6). *VPoser-t* gives reasonable global errors, but suffers frequent penetrations as shown for sitting in Fig. 2.6. Using *MVAE* or ablations of HuMoR as the motion prior in TestOpt fails to effectively generalize to real-world data and performs worse than the full model. The conditional prior and $\mathcal{L}_{\text{SMPL}}$ have the largest impact, while performance even without using contacts still outperforms the baselines.

The top half of Tab. 2.4 evaluates plausibility on additional RGB results from PROX compared to *VIBE* and *PROX-RGB*. Since *PROX-RGB* uses the scene mesh as input to enforce environment constraints, it is a very strong baseline and its performance on penetration metrics is expectedly good. HuMoR comparatively increases penetration frequency since it only gets a rough ground plane as initialization, but gives much smoother motions. The bottom half of Tab. 2.4 shows results fitting to RGB-D for the same PROX data, which uses both $\mathcal{E}_{\text{data}}^{\text{2D}}$ *and* $\mathcal{E}_{\text{data}}^{\text{PC3D}}$ in TestOpt. This improves performance using HuMoR, slightly outperforming *PROX-D* which is less robust to issues with 2D

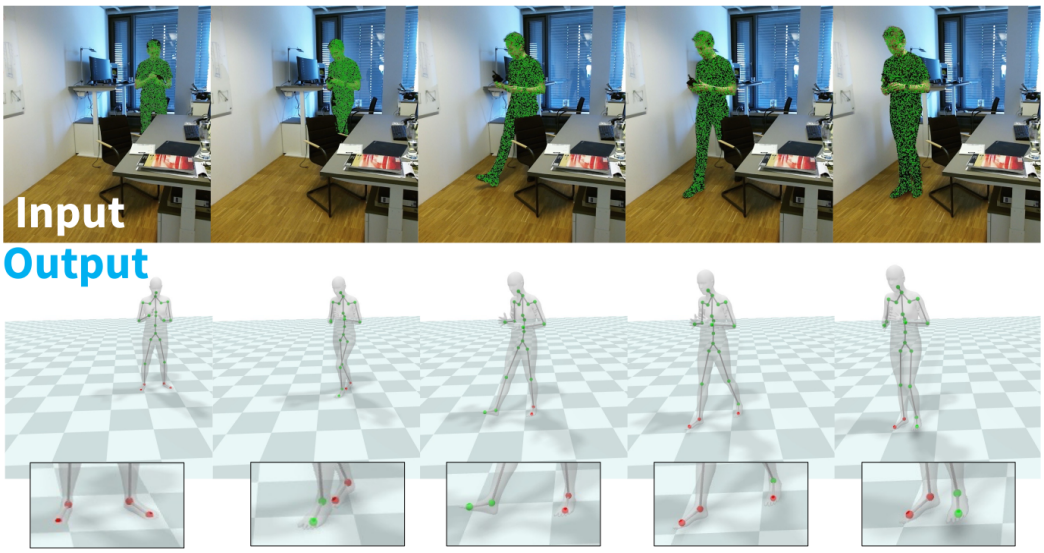

Figure 2.7: From RGB-D (top) TestOpt with HuMoR outputs 3D motion, the ground plane, and contacts (bottom).

joint detections and 3D point noise causing large errors. Qualitative examples are in Fig. 2.1 and Fig. 2.7.

Thanks to the generalizability of HuMoR, TestOpt is also effective in recovering very dynamic motions like dancing from RGB video even though HuMoR is trained on few dancing motions. Fig. 2.8 shows fitting results on videos from the AIST dance dataset [255]. Since HuMoR allows for large accelerations, it accurately generalizes to fast motions (top - note motion blur). Moreover, it is able to recover from poor 2D joint detections from OpenPose due to the cartwheel motion (bottom).

			Ground Pen	
Method	**Input**	**Accel**	**Freq**	**Dist**
VIBE [126]	RGB	86.06	23.46%	4.71
PROX-RGB [93]	RGB	196.07	**2.55%**	**0.32**
VPoser-t	RGB	3.14	13.38%	2.82
HuMoR (Ours)	RGB	**1.73**	9.99%	1.56
PROX-D [93]	RGB-D	46.59	8.95%	1.19
VPoser-t	RGB-D	3.27	10.66%	2.18
HuMoR (Ours)	RGB-D	**1.61**	**5.19%**	**0.85**

Table 2.4: Plausibility evaluation on videos in PROX [93]. Acceleration is m/s^2 and penetration distance in cm.

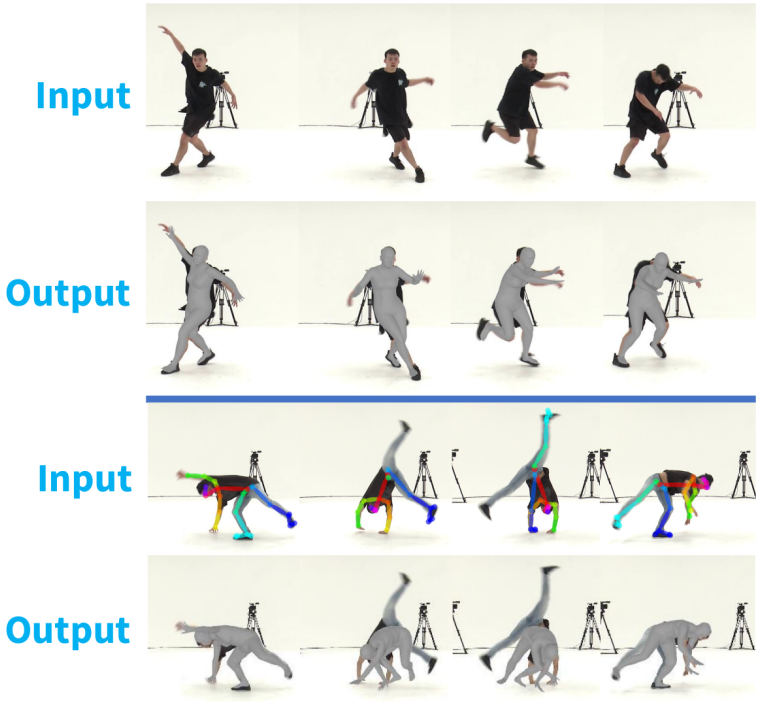

Figure 2.8: Example sequences using TestOpt with HuMoR to fit to 2D joints in AIST dance videos [255]. HuMoR generalizes to these highly dynamic motions and robustly recovers from inaccurate 2D joint detections (bottom).

2.6 Discussion

We have introduced HuMoR, a learned generative model of 3D human motion leveraged during test-time optimization to robustly recover pose and shape from 3D, RGB, and RGB-D observations. We have demonstrated that the key components of our model enable generalization to novel motions and body shapes for both generative tasks and downstream optimization. Compared to strong learning and optimization-based baselines, HuMoR excels at estimating plausible motion under heavy occlusions, and simultaneously produces consistent ground plane and contact outputs.

In developing HuMoR, it was necessary to tackle nearly all the challenges outlined in Sec. 1.1.1. To get *accuracy and diversity*, we used the generative CVAE architecture that operates in the structured human pose space. Some basic notion of *interactions* was also considered by predicting ground contacts, though capturing more detailed human-environment interactions is an important future direction (see Chapter 6). HuMoR *generalizes* to new motions and body shapes thanks to the single-step transition formulation and latent variable architecture to account for the unobservable

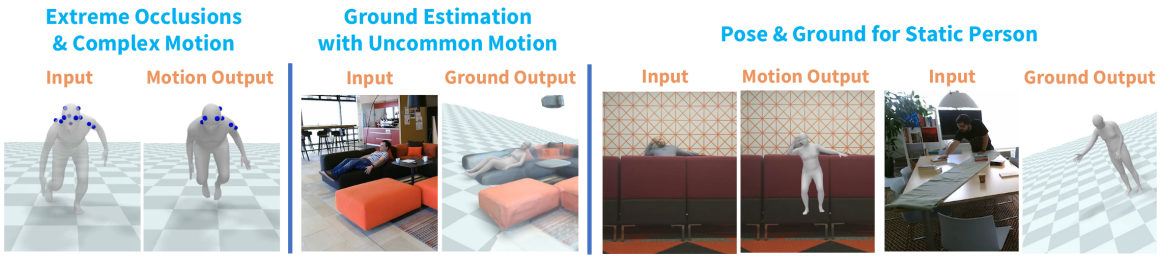

Figure 2.9: Failure cases of TestOpt using HuMoR.

factors of human motion. Lastly, we were able to *control* human motions to match video observations by using optimization in the learned latent space.

Limitations & Future Work. HuMoR leaves ample room for future studies. The static camera and ground plane assumptions are reasonable for indoor scenes but true *in-the-wild* operation demands methods handling dynamic cameras and complex terrain. Our rather simplistic contact model should be upgraded to capture scene-person interactions for improved motion and scene perception. Lastly, we plan to *learn* motion estimation directly from partial observations which will be faster than TestOpt and enable *sampling* multiple plausible motions rather than relying on a single local minimum.

Specific failure cases shown in Fig. 2.9 further highlight areas of future improvement. First, extreme occlusions (*e.g.* only a few visible points as in Fig. 2.9 left), especially at the first frame which determines $\mathbf{x}_0$, makes for a difficult optimization that often lands in local minima with implausible motions. Second, uncommon motions that are rare during CVAE training, such as laying down in Fig. 2.9 (middle), can cause spurious ground plane outputs as TestOpt attempts to make the motion more likely. Leveraging more holistic scene understanding methods and models of human-environment interaction will help in these cases. Finally, our method is dependent on motion in order to resolve ambiguity, which is usually very helpful but has corner cases as shown in Fig. 2.9 (right). For example, if the observed person is nearly static, the optimization may produce implausible poses due to ambiguous occlusions (*e.g.* standing when really the person is sitting) and/or incorrect ground plane estimations.

2.7 Additional Related Contributions

This section briefly describes additional contributions made to the area of 3D human motion modeling. In particular, Sec. 2.7.1 summarizes a physics-based approach to modeling human motion for pose

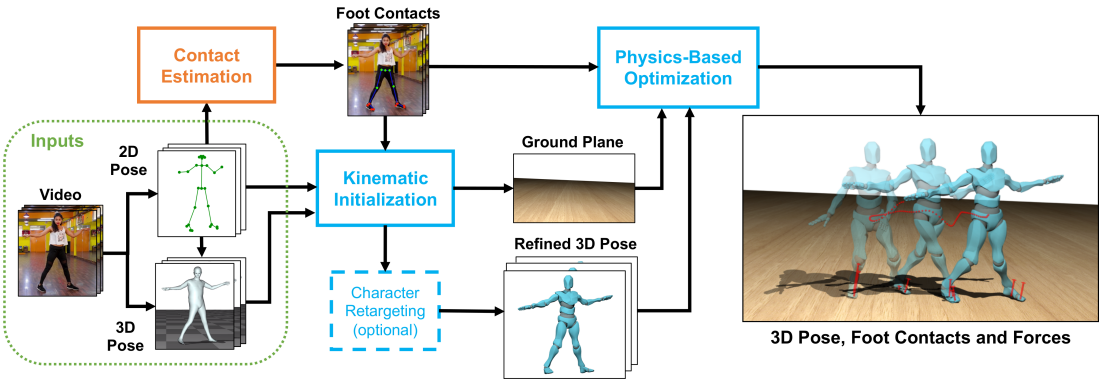

Figure 2.10: Overview of contact and human dynamics from video. Our method starts with initial estimates from existing 2D and 3D pose methods and first infers foot contacts (orange box). A reduced-dimensional physics-based trajectory optimization fits to the 3D pose input using estimated contacts. This optimization recovers physically-plausible pose and foot contact forces.

estimation [208] in contrast to the learned HuMoR model.

2.7.1 Physics-Based Human Motion Modeling

Recent methods for human pose estimation from monocular video estimate accurate overall body pose with small absolute differences from the true poses in body-frame 3D coordinates. However, the recovered motions in world-frame are visually and physically implausible in many ways, including feet that float slightly or penetrate the ground, implausible forward or backward body lean, and motion errors like jittery, vibrating poses. As summarized in Fig. 2.10, we present a physics-based method for inferring 3D human motion from video sequences that takes initial 2D and 3D pose estimates as input. We first estimate ground contact timings with a novel prediction network, which is trained without hand-labeled data. A physics-based trajectory optimization then solves for a physically-plausible motion, based on the inputs.

We show this process produces motions that are significantly more realistic than those from purely kinematic methods, substantially improving quantitative measures of both kinematic and dynamic plausibility. We demonstrate our method on character animation and pose estimation tasks on dynamic motions of dancing and sports with complex contact patterns. Please refer to the paper for full technical details and results [208].

Chapter 3

Modeling 3D Object Motion for Point Cloud Perception

In Chapter 2, we saw that 3D humans have a structured kinematic pose that offers an obvious state space for modeling motion. Therefore, HuMoR made predictions for how pose would change over time explicitly within this state space consisting of joint positions, angles, and velocities. However, 3D objects generally experience a huge variety of motions. For example, rigid motion as a result of the object or observer moving, and deformation that creates complex changes in underlying geometry. Even if objects move in an articulated manner like humans, we may not know what the kinematic structure of their skeleton is, or the skeleton may change across instances in an object class.

This variation in motion type makes modeling the change in 3D object pose over time challenging. There is no obvious low-dimensional pose space that will allow us to flexibly capture all rigid, articulated, and deformable motion. In this chapter, we look at how to model 3D object motion more generally by considering how its shape changes over time from the perspective of a 3D sensor that produces dynamic point clouds. To achieve this, we will introduce a method called CaSPR, which contains a latent model of object motion that is flexible to rigid and deformable motions alike. CaSPR was originally published in NeurIPS 2020 [207].

3.1 Introduction

The visible geometric properties of objects around us are constantly evolving over time due to object motion, articulation, deformation, or observer movement. Examples include the rigid *motion* of cars on the road, the *deformation* of clothes in the wind, and the *articulation* of moving humans. The ability

to capture and reconstruct these spatiotemporally changing geometric object properties is critical in applications like autonomous driving, robotics, and mixed reality. Recent work has made progress on learning object shape representations from static 3D observations [185, 204, 205, 237, 277] and dynamic point clouds [37, 42, 152, 153, 177, 200, 304]. Yet, important limitations remain in terms of the lack of temporal continuity, robustness, and category-level generalization.

In this chapter, we address the problem of learning object-centric representations that can aggregate and encode **spatiotemporal (ST) changes** in object shape as seen from a 3D sensor. This is challenging since dynamic point clouds captured by depth sensors or LIDAR are often incomplete and sparsely sampled over space and time. Furthermore, even point clouds corresponding to adjacent frames in a sequence will experience large sampling variation. Ideally, we would like spatiotemporal representations to satisfy several desirable properties. First, representations should allow us to capture object shape **continuously** over space *and* time. They should encode changes in shape due to varying camera pose or temporal dynamics, and support shape generation at arbitrary spatiotemporal resolutions. Second, representations should be **robust** to irregular sampling patterns in space and time, including support for full or partial point clouds. Finally, representations should support within-category **generalization** to unseen object instances and to unseen temporal dynamics. While many of these properties are individually considered in prior work [42, 114, 153, 177, 262], a unified and rigorous treatment of all these factors in space and time is largely missing.

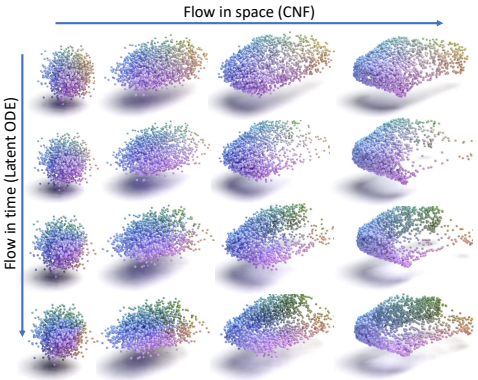

Figure 3.1: CaSPR builds a point cloud representation of (partially observed) objects continuously in both space (x-axis) and time (y-axis), while canonicalizing for extrinsic object properties like pose.

We address the limitations of previous work by learning a novel object-centric ST representation which satisfies the above properties. To this end, we introduce **CaSPR** – a method to learn **Ca**nonical **S**patiotemporal **P**oint Cloud **R**epresentations. In our approach, we split the task into two: (1) *canonicalizing* an input object point cloud sequence (partial or complete) into a shared 4D container space, and (2) learning a continuous ST latent representation on top of this canonicalized space. For the former, we build upon the Normalized Object Coordinate Space (NOCS) [241, 273] which canonicalizes intra-class 3D shape variation by normalizing for extrinsic properties like position, orientation, and scale. We extend NOCS to a 4D **Temporal-NOCS (T-NOCS)**, which additionally normalizes the duration of the input sequence to a unit interval. Given dynamic point cloud sequences,

our ST canonicalization yields spacetime-normalized point clouds. In Sec. 3.5, we show that this allows learning representations that generalize to novel shapes and dynamics.

We learn ST representations of canonicalized point clouds using *Neural Ordinary Differential Equations* (Neural ODEs) [37]. Different from previous work, we use a **Latent ODE** that operates in a lower-dimensional *learned latent space* which increases efficiency while still capturing object shape dynamics. Given an input sequence, the canonicalization network and Latent ODE together extract features that constitute an ST representation. To continuously *generate* novel spatiotemporal point clouds conditioned on an input sequence, we further leverage invertible *Continuous Normalizing Flows* (CNFs) [34, 78] which transform Gaussian noise directly to the visible part of an object's shape at a desired timestep. Besides continuity, CNFs provide direct likelihood evaluation which we use as a training loss. Together, as shown in Fig. 3.1, the Latent ODE and CNF constitute a generative model that is continuous in spacetime and robust to sparse and varied inputs. Unlike previous work [42, 153], our approach is continuous and explicitly avoids treating time as another spatial dimension by respecting its unique aspects (*e.g.*, unidirectionality).

We demonstrate that CaSPR is useful in numerous applications including (1) continuous space-time shape reconstruction from sparse, partial, or temporally non-uniform input point cloud sequences, (2) spatiotemporal 6D pose estimation, and (3) information propagation via space-time correspondences under rigid or non-rigid transformations. Our experiments show improvements to previous work while also providing insights on the emergence of intra-class shape correspondence and the learning of *time unidirectionality* [61]. In summary, our contributions are:

- The CaSPR encoder network that consumes dynamic object point cloud sequences and canonicalizes them to normalized spacetime (T-NOCS).
- The CaSPR representation of canonicalized point clouds using a Latent ODE to explicitly encode temporal dynamics, and an associated CNF for generating shapes continuously in spacetime.
- A diverse set of applications of this technique, including partial or full shape reconstruction, spatiotemporal sequence recovery, camera pose estimation, and correspondence estimation.

Supplementary video results are available on the project webpage.

3.2 Related Work

Neural Representations of Point Sets. Advances in 2D deep architectures leapt into the realm of

point clouds with PointNet [204]. The lack of locality in PointNet was later addressed by a diverse set of works [51, 52, 143, 230, 244, 252, 278, 288, 313], including PointNet++ [205] – a permutation invariant architecture capable of learning both local and global point features. We refer the reader to Guo *et al.* [85] for a thorough review. Treating time as the fourth dimension, our method heavily leverages propositions from these works. Continuous reconstruction of an object's spatial geometry has been explored by recent works in learning implicit shape representations [40, 90, 170, 185].

Spatiotemporal Networks for 3D Data. Analogous to volumetric 3D convolutions on video frames [135, 264, 308], a direct way to process spatiotemporal point cloud data is performing 4D convolutions on a voxel representation. This poses three challenges: (1) storing 4D volumes densely is inefficient and impractical, (2) direct correlation of spatial and temporal distances is undesirable, and (3) the inability to account for timestamps can hinder the final performance. These challenges have fostered further research along multiple fronts. For example, a large body of works [11, 82, 152, 279] has addressed temporal changes between a pair of scans as per-point displacements or *scene flow* [267]. While representing dynamics as fields of change is tempting, such methods lack an explicit notion of time. MeteorNet [153] was an early work to learn flow on raw point cloud sequences, however it requires explicit local ST neighborhoods which is undesirable for accuracy and generalization. Prant *et al.* [200] use temporal frames as a cue of coherence to stabilize the generation of points. CloudLSTM [304] models temporal dependencies implicitly within sequence-to-sequence learning. Making use of time in a more direct fashion, MinkowskiNet [42] proposed an efficient ST 4D CNN to exploit the sparsity of point sets. This method can efficiently perform 4D sparse convolutions, but can neither canonicalize time nor perform ST aggregation. OccupancyFlow [177] used occupancy networks [170] and Neural ODEs [37] to have an explicit notion of time.

Our method can be viewed as learning the underlying *kinematic spacetime surface* of an object motion: an idea from traditional computer vision literature for dynamic geometry registration [171].

Canonicalization. Regressing 3D points in a common global reference frame dates back to 6D camera relocalization and is known as *scene coordinates* [234]. In the context of learning the *normalized object coordinate space* (NOCS), [273] is notable for explicitly mapping the input to canonical *object coordinates*. Thanks to this normalization, NOCS enabled category-level pose estimation and has been extended to articulated objects [141], category-level rigid 3D reconstruction [43, 79, 115] via multiview aggregation [241], and non-rigid shape reconstruction either via deep implicit surfaces [299] or by disentangling viewpoint and deformation [178]. Chen *et al.* [35] proposed a latent variational NOCS to *generate* points in a canonical frame.

Normalizing Flows and Neural ODEs. The idea of transforming noise into data dates back to whitening transforms [69] and Gaussianization [38]. Tabak and Turner [249] officially defined normalizing flows (NFs) as the composition of simple maps and used it for non-parametric density estimation. NFs were immediately extended to deep networks and high dimensional data by Rippel and Adams [216]. Rezende and Mohamed used NFs in the setting of variational inference [214] and popularized them as a standalone tool for deep generative modeling *e.g.* [123, 245]. Thanks to their invertibility and exact likelihood estimation, NFs are now prevalent and have been explored in the context of graph neural networks [150], generative adversarial networks [80], bypassing topological limitations [7, 45, 60], flows on Riemannian manifolds [74, 157, 227], equivariant flows [18, 128, 215], and connections to optimal transport [67, 180, 269, 306]. The limit case where the sequence of transformations are indexed by real numbers yields continuous-time flows: the celebrated Neural ODEs [34], their latent counterparts [220], and FFJORD [78], an invertible generative model with unbiased density estimation. For a comprehensive review, we refer the reader to the concurrent surveys of [125, 184].

Our algorithm is highly connected to PointFlow [287] and C-Flow [203]. However, we tackle encoding and generating spatiotemporal point sets in addition to canonicalization while both of these works use CNFs in generative modeling of 3D point sets without canonicalizing.

3.3 Background

In this section, we lay out the notation and mathematical background required in Sec. 3.4.

Definition 1 (Flow & Trajectory) *Let us define a d-dimensional **flow** to be a parametric family of homeomorphisms $\phi : \mathcal{M} \times \mathbb{R} \mapsto \mathcal{M}$ acting on a vector $\mathbf{z} \in \mathcal{M} \subset \mathbb{R}^d$ with $\phi_0(\mathbf{z}) = \mathbf{z}$ (identity map) and $\phi_t(\mathbf{z}) = \mathbf{z}_t$. A temporal subspace of flows is said to be a **trajectory** $\mathcal{T}(\mathbf{z}) = \{\phi_t(\mathbf{z})\}_t$ if $\mathcal{T}(\mathbf{z}) \cap \mathcal{T}(\mathbf{y}) = \emptyset$ for all $\mathbf{z} \neq \mathbf{y}$, i.e., different trajectories never intersect [44, 60].*

Definition 2 (ODE-Flow, Neural ODE & Latent ODE) *For any given flow ϕ there exists a corresponding ordinary differential equation (**ODE**) constructed by attaching an optionally time-dependent vector $f(\mathbf{z}, t) \in \mathbb{R}^d$ to every point $\mathbf{z} \in \mathcal{M}$ resulting in a vector field s.t. $f(\mathbf{z}) = \phi'(\mathbf{z})|_{t=0}$. Starting from the initial state $\mathbf{z}_0$, this ODE given by $\frac{dz(t)}{dt} = f(z(t), t)$ can be integrated for time T modeling the flow $\phi_{t=T}$:*

$$\mathbf{z}_T = \phi_T(\mathbf{z}_0) = \mathbf{z}_0 + \int_0^T f_\theta(\mathbf{z}_t, t)\, dt, \tag{3.1}$$

where $\mathbf{z}_t \triangleq z(t)$ *and the field* f *is parameterized by* $\boldsymbol{\theta} = \{\theta_i\}_i$*. By the Picard–Lindelöf theorem [44], if* f *is continuously differentiable then the initial value problem in Eq.* (3.1) *has a unique solution. Instead of handcrafting,* ***Neural ODEs*** *[37] seek a function* f *that suits a given objective by modeling* f *as a neural network. We refer to a Neural ODE operating in a latent space as a* ***Latent ODE****.*

Numerous forms of Neural ODEs model $f(\cdot)$ to be *autonomous*, *i.e.*, time independent $f(\mathbf{z}_t) \equiv f(\mathbf{z}_t, t)$ [37, 60, 220], whose output fully characterizes the trajectory. While a Neural ODE advects single particles, **generative modeling** approximates the full target probability density which requires expressive models capable of exact density evaluation and sampling that avoids mode collapse.

Definition 3 (Continuous Normalizing Flow (CNF)) *Starting from a simple* $d_{\mathcal{B}}$*-dimensional base distribution* p_y *with* $\mathbf{y}_0 \in \mathbb{R}^{d_{\mathcal{B}}} \sim p_y(\mathbf{y})$*,* ***CNFs*** *[34, 78] aim to approximate the complex target distribution* $p_x(\mathbf{x})$ *by bijectively mapping empirical samples of the target to the base using an invertible function* $g_\beta : \mathbb{R}^{d_{\mathcal{B}}} \mapsto \mathbb{R}^{d_{\mathcal{B}}}$ *with parameters* $\boldsymbol{\beta} = \{\beta_i\}_i$*. Then the probability density function transforms with respect to the change of variables:* $\log p_x(\mathbf{x}) = \log p_y(\mathbf{y}) - \log\det \nabla g_\beta(\mathbf{y})$*. The warping function* g *can be replaced by an integral of continuous-time dynamics yielding a form similar to Neural ODEs except that we now consider distributions [78]:*

$$\log p_x(\mathbf{x}) \;=\; \log p_y(\mathbf{y}_0) - \int_0^T \mathrm{Tr}\left(\frac{\partial g_\beta(\mathbf{y}_t, t \,|\, \mathbf{z})}{\partial \mathbf{y}_t}\right) dt, \tag{3.2}$$

with the simplest choice that the base distribution $\mathbf{y}_0$ *is in a* d*-dimensional ball,* $p_y \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$*. Here* $\mathbf{z} \in \mathbb{R}^d$ *is an optional* ***conditioning*** *latent vector [287]. Note that this continuous system is* non-autonomous *i.e., time varying and every non-autonomous system can be converted to an autonomous one by raising the dimension to include time [49, 60].*

3.4 Method

We consider as input a sequence of potentially partial, clutter-free 3D scans (readily captured by depth sensors or LIDAR) of an object belonging to a known category. This observation is represented as a point cloud $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^3 = \{x_i, y_i, z_i\} \,|\, i = 1, \ldots, M'\}$. For a sequence of K potentially non-uniformly sampled timesteps, we denote a **spatiotemporal** (ST) point cloud as $\mathcal{P} = \{\mathcal{P}_k\}_{k=1}^K$, where $\mathcal{P}_k = \{\mathbf{p}_i \in \mathbb{R}^4 = \{x_i, y_i, z_i, s_k\} \,|\, i = 1, \ldots, M_k\}$, M_k is the number of points at frame $k \in [1, K]$ and at the *time* $s_k \in [s_1, s_K] \subset \mathbb{R}$ with $M = \sum_{k=1}^K M_k$. Our goal is to explain $\mathcal{P}$ by learning a *continuous representation* of shape that is invariant to extrinsic properties while aggregating intrinsic properties along the direction of time. **CaSPR** achieves this through:

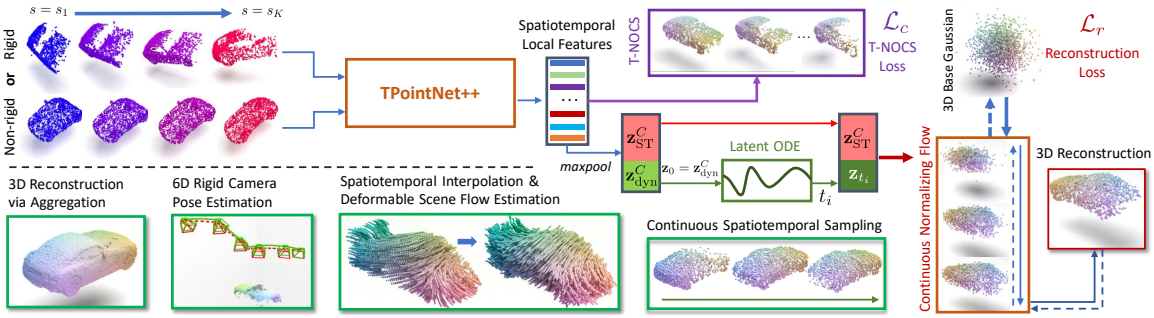

Figure 3.2: Architecture and applications of CaSPR. Our model consumes rigid or deformable point cloud sequences and maps them to a spatiotemporal canonical latent space whose coordinates are visualized by RGB colors (purple box). Using a Latent ODE, it advects a latent subspace forward in time to model temporal dynamics. A continuous normalizing flow [78] (shown in red) decodes the final latent code to 3D space by mapping Gaussian noise to the partial or full shape at desired timesteps. CaSPR enables multiple applications shown in green boxes. Training directions for the normalizing flow are indicated by dashed arrows.

1. A **canonical** spacetime container where extrinsic properties such as object pose are factored out,

2. A continuous **latent** representation which can be queried at arbitrary spacetime steps, and

3. A **generative** model capable of reconstructing partial observations conditioned on a latent code.

We first describe the method design for each of these components, depicted in Fig. 3.2, followed by implementation and architectural details in Sec. 3.4.1.

Canonicalization: The first step is *canonicalization* of a 4D ST point cloud sequence with the goal of associating observations at different time steps to a common canonical space. Unlike prior work which assumes already-canonical inputs [177, 287], this step allows CaSPR to operate on raw point cloud sequences in world space and enables multiple applications (see Fig. 3.2). Other previous work has considered canonicalization of extrinsic properties from RGB images [241, 273] or a 3D point cloud [141], but our method operates on a 4D point cloud and explicitly accounts for time labels. Our goal is to find an injective *spacetime canonicalizer* $c_\alpha(\cdot) : \mathcal{P} \mapsto \overline{\mathcal{P}} \times \overline{\mathcal{Z}}$ parameterized by $\boldsymbol{\alpha} = \{\alpha_i\}_i$, that maps a point cloud sequence $\mathcal{P}$ to a canonical *unit tesseract* $\overline{\mathcal{P}} = \{\overline{\mathcal{P}}_k\}_{k=1}^K$, where $\overline{\mathcal{P}}_k = \{\overline{\mathbf{p}}_i \in \mathbb{R}^4 = \{\bar{x}_i, \bar{y}_i, \bar{z}_i, \bar{t}_k\} \in [0,1] \,|\, i = 1, \ldots, M_k\}$ and $\mathbf{z}^C \in \overline{\mathcal{Z}} \subset \mathbb{R}^d$ is the corresponding canonical *latent* representation (embedding) of the sequence. Note that in addition to position and orientation, $\overline{\mathcal{P}}$ is normalized to have time in unit duration. We refer to $\overline{\mathcal{P}}$ as **Temporal-NOCS (T-NOCS)** as it extends NOCS [241, 273]. T-NOCS points are visualized using the spatial coordinate as the *RGB* color in Figs. 3.2 and 3.3. Given a 4D point cloud in the world frame, we can aggregate

the entire shape from K partial views by a simple union: $\overline{\mathcal{P}} = \bigcup_{i=1}^{K} c_\alpha(\mathcal{P}_i)$ [241]. Moreover, due to its injectivity, $c_\alpha(\cdot)$ preserves *correspondences*, a property useful in tasks like pose estimation or label propagation. We outline the details and challenges involved in designing a canonicalizer in Sec. 3.4.1.

Continuous Spatiotemporal Representation: While a global ST latent embedding is beneficial for canonicalization and aggregation of partial point clouds, we are interested in continuously modeling the ST input, *i.e.*, being able to compute a representation for unobserved timesteps at arbitrary spacetime resolutions. To achieve this, we split the latent representation: $\mathbf{z}^C = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}^C_{\text{dyn}}]$ where $\mathbf{z}^C_{\text{ST}}$ is the *static* ST descriptor and $\mathbf{z}^C_{\text{dyn}}$ is used to initialize an autonomous Latent ODE $\frac{d\mathbf{z}_t}{dt} = f_\theta(\mathbf{z}_t)$ as described in Dfn. 2: $\mathbf{z}_0 = \mathbf{z}^C_{\text{dyn}} \in \mathbb{R}^d$. We choose to advect the ODE in the latent space (rather than physical space [177]) to (1) enable *learning* a space best-suited to modeling the dynamics of the observed data, and (2) improve scalability due to the fixed feature size. Due to the time-independence of f_θ, $\mathbf{z}_0$ fully characterizes the latent trajectory. Advecting $\mathbf{z}_0$ forward in time by solving this ODE until any canonical timestamp $T \leq 1$ yields a continuous representation in time $\mathbf{z}_T$ that can explain changing object properties. We finally obtain a dynamic spatiotemporal representation in the product space: $\mathbf{z} \in \mathbb{R}^D = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}_T]$. Due to canonicalization to the unit interval, $T > 1$ implies *extrapolation*.

Spatiotemporal Generative Model: Numerous methods exist for point set generation [2, 79, 313], but most are not suited for sampling on the surface of a partial 4D ST point cloud. Therefore, we adapt CNFs [78, 287] as defined in Sec. 3.3. To generate a novel ST shape, *i.e.*, a sequence of 3D shapes $\mathcal{X}_1 \dots \mathcal{X}_K$, we simulate the Latent ODE for $t = 0 \dots T$ and obtain representations for each of the canonical shapes in the sequence: $\mathbf{z}_{t=0} \cdots \mathbf{z}_{t=T}$. We then sample the base distribution $\mathbf{y}_k \in \mathbb{R}^{d_{\mathcal{B}}=3} \sim p_y(\mathbf{y}) \triangleq \mathcal{N}(\mathbf{0}, \mathbf{I})$ and evaluate the conditional CNF in Eq. (3.2) by passing each sample $\mathbf{y}_k$ through the flow $g_\beta(\mathbf{y}_k \mid \mathbf{z}_t)$ conditioned on $\mathbf{z}_t$. Note that the flow is time dependent, *i.e.*, non-autonomous. To increase the temporal resolution of the output samples we pick the timesteps with higher frequency, whereas to *densify* spatially, we simply generate more samples $\mathbf{y}_k$.

3.4.1 Network Architecture

We now detail our implementations of the canonicalizer c_α, Latent ODE network f_θ, and CNF g_β.

TPointNet++ $c_\alpha(\cdot)$: The design of our canonicalizer is influenced by (1) the desire to avoid ST neighborhood queries, (2) to treat time as important as the spatial dimensions, and (3) injecting how an object appears during motion in space into its local descriptors resulting in more expressive features. While it is tempting to directly apply existing point cloud architectures such as PointNet [204] or

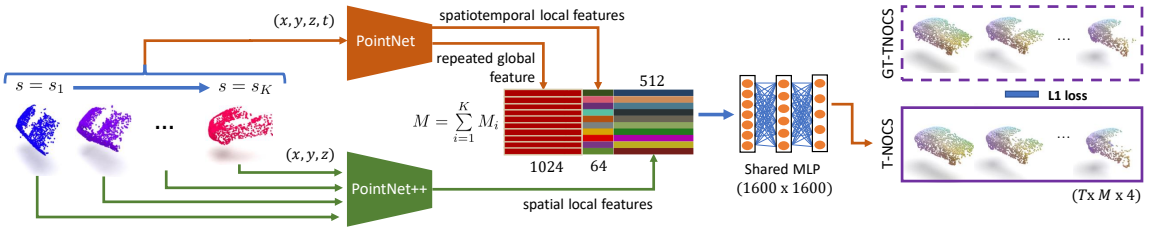

Figure 3.3: Architecture of our ST point-set canonicalization network, *TPointNet++*. It uses two branches that extract ST features using a 4D PointNet and per-view 3D local features via PointNet++. These features are combined and passed to an MLP to regress the T-NOCS points. Training is supervised via GT coordinates.

PointNet++ [205], we found experimentally that they were individually insufficient (*c.f.* Sec. 3.5). To meet our goals, we instead introduce a hybrid *TPointNet++* architecture as shown in Fig. 3.3 to implement c_α and canonicalize $\mathcal{P}$ to $\overline{\mathcal{P}}$. TPointNet++ contains a PointNet branch that consumes the entire 4D point cloud to extract both a 1024-dimensional global feature and 64-dimensional per-point ST features. This treats time explicitly and equally to each spatial dimension. We also use PointNet++ to extract a 512-dimensional local feature at each input point by applying it at each cross-section in time with no timestamp. We feed all features into a shared multi-layer perceptron (MLP) to arrive at 1600-dimensional embeddings corresponding to each input point.

We use the pointwise embeddings in two ways: (1) they are passed through a shared linear layer followed by a *sigmoid* function to estimate the T-NOCS coordinates $\widehat{\mathcal{P}}$ which approximate the ground truth $\overline{\mathcal{P}}$, and (2) we max-pool all per-point features into a single latent representation of T-NOCS $\mathbf{z}^C \in \mathbb{R}^{1600}$ which is used by the Latent ODE and CNF as described below. The full canonicalizer $c_\alpha(\cdot)$ can be trained independently for T-NOCS regression, or jointly with a downstream task.

Latent ODE $f_\theta(\cdot)$ and Reconstruction CNF $g_\beta(\cdot)$: The full CaSPR architecture is depicted in Fig. 3.2. It builds upon the embedding from TPointNet++ by first splitting it into two parts $\mathbf{z}^C = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}_0 \triangleq \mathbf{z}^C_{\text{dyn}}]$. The dynamics network of the Latent ODE f_θ is an MLP with three hidden layers of size 512. We use a Runge-Kutta 4(5) solver [133, 223] with adaptive step sizes which supports backpropagation using the adjoint method [37]. The static feature, $\mathbf{z}^C_{\text{ST}} \in \mathbb{R}^{1536}$ is skip-connected and concatenated with $\mathbf{z}_T$ to yield $\mathbf{z} \in \mathbb{R}^{1600}$ which conditions the reconstruction at $t = T$.

To sample the surface represented by $\mathbf{z}$, we use a FFJORD conditional-CNF [78, 287] as explained in Secs. 3.3 and 3.4 to map 3D Gaussian noise $\mathbf{y}_0 \in \mathbb{R}^{d_{\mathcal{B}}=3} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ onto the shape surface. The dynamics of this flow $g_\beta(\mathbf{y}_t, t \,|\, \mathbf{z})$ are learned with a modified MLP [78] which leverages a gating mechanism at each layer to inject information about the current context including $\mathbf{z}$ and current time t of the flow. This MLP contains three hidden layers of size 512, and we use the same solver as the

Figure 3.4: Canonicalization applications. Partial shape reconstruction (left section) shows pairs of GT (left) and predicted shapes (right). Pose estimation (right section) shows GT (green, solid) and predicted (red, dashed) camera pose based on regressed T-NOCS points. Points are colored by their T-NOCS location.

Latent ODE.

Training and Inference: CaSPR is trained with two objectives that use the GT canonical point cloud sequence $\overline{\mathcal{P}}$ as supervision. We primarily seek to maximize the *log-likelihood* of canonical spatial points on the surface of the object when mapped to the base Gaussian using the CNF. This reconstruction loss is $\mathcal{L}_r = -\sum_{k=1}^{K}\sum_{i=1}^{M_k} \log p_x(\bar{\mathbf{x}}_i \,|\, \mathbf{z}_{t_k})$ where $\bar{\mathbf{x}}_i$ is the spatial part of $\bar{\mathbf{p}}_i \in \overline{\mathcal{P}}_k$ and the log-likelihood is computed using Eq. (3.2). Secondly, we supervise the T-NOCS predictions from TPointNet++ with an L1 loss $\mathcal{L}_c = \sum_{i=1}^{M} |\widehat{\mathbf{p}}_i - \bar{\mathbf{p}}_i|$ with $\bar{\mathbf{p}}_i \in \overline{\mathcal{P}}$ and $\widehat{\mathbf{p}}_i \in \widehat{\mathcal{P}}$. We jointly train TPointNet++, the Latent ODE, and CNF for $\boldsymbol{\alpha}$, $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ respectively with the final loss $\mathcal{L} = \mathcal{L}_r + \mathcal{L}_c$. During inference, TPointNet++ processes a raw point cloud sequence of an unseen shape and motion to obtain the ST embedding and canonicalized T-NOCS points. The Latent ODE, initialized by this embedding, is solved forward in time to any number of canonical "query" timestamps. For each timestamp, the Latent ODE produces the feature to condition the CNF which reconstructs the object surface by the forward flow of Gaussian samples. The combined continuity of the Latent ODE and CNF enables CaSPR to reconstruct the input sequence at any desired ST resolution.

3.5 Experimental Evaluations

We now evaluate the canonicalization, representation, and reconstruction capabilities of CaSPR, demonstrate its utility in multiple downstream tasks, and justify design choices.

Dataset and Preprocessing: We introduce a new dataset containing simulated rigid motion of objects in three ShapeNet [32] categories: cars, chairs, and airplanes. The motion is produced with randomly generated camera trajectories (Fig. 3.4) and allows us to obtain the necessary inputs and supervision for CaSPR: sequences of raw partial point clouds from depth maps with corresponding canonical T-NOCS point clouds. Each sequence contains $K = 10$ frames with associated timestamps. Raw point cloud sequences are labeled with uniform timestamps from $s_1 = 0.0$ to $s_K = 5.0$ while

canonicalized timestamps range from $\bar{t}_1 = 0$ to $\bar{t}_K = 1$. For training, 5 frames with 1024 points are randomly subsampled from each sequence, giving non-uniform step sizes between observations. At test time, we use a *different spatiotemporal sampling* for sequences of held-out object instances: all 10 frames, each with 2048 points. Separate CaSPR models are trained for each shape category.

Evaluation Procedure: To measure canonicalization errors, T-NOCS coordinates are split into the spatial and temporal part with GT given by $\bar{\mathcal{X}}$ and $\bar{\mathbf{t}}$ respectively. The *spatial error* at frame k is $\frac{1}{M_k}\sum_{i=1}^{M_k} \|\widehat{\mathbf{x}}_i - \bar{\mathbf{x}}_i\|_2$ and the *temporal error* is $\frac{1}{M_k}\sum_{i=1}^{M_k} |\widehat{t_i} - \bar{t_i}|$. For reconstruction, the Chamfer Distance (CD) and Earth Mover's Distance (EMD) are measured (and reported multiplied by 10^3). For our purposes, we define the CD between two point clouds $\mathcal{X}_1, \mathcal{X}_2$ each with N points as

$$d_{CD}\left(\mathcal{X}_1, \mathcal{X}_2\right) = \frac{1}{N}\sum_{\mathbf{x}_1 \in \mathcal{X}_1} \min_{\mathbf{x}_2 \in \mathcal{X}_2} \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 + \frac{1}{N}\sum_{\mathbf{x}_2 \in \mathcal{X}_2} \min_{\mathbf{x}_1 \in \mathcal{X}_1} \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \tag{3.3}$$

and the EMD as

$$d_{EMD}\left(\mathcal{X}_1, \mathcal{X}_2\right) = \min_{\phi: \mathcal{X}_1 \to \mathcal{X}_2} \frac{1}{N}\sum_{\mathbf{x}_1 \in \mathcal{X}_1} \|\mathbf{x}_1 - \phi(\mathbf{x}_1)\|_2^2 \tag{3.4}$$

where $\phi : \mathcal{X}_1 \to \mathcal{X}_2$ is a bijection. In practice, we use a fast approximation of the EMD based on [15]. Lower is better for all metrics; we report the median over all test frames because outlier shapes cause less informative mean errors. Unless stated otherwise, qualitative point cloud results (*e.g.*, Fig. 3.4) are colored by their canonical coordinate values (so corresponding points should have the same color).

3.5.1 Evaluations and Applications

Canonicalization: We first evaluate the accuracy of canonicalizing raw partial point cloud sequences to T-NOCS using TPointNet++. Tab. 3.1 shows median errors over all frames in the test set. The bottom part evaluates TPointNet++ on each shape category while the top compares with baselines on cars. We consider the following baselines:

- *MeteorNet* [153]: A recent method that extends PointNet++ to process point cloud sequences through spatiotemporal neighborhood queries. We adapt the *MeteorNet-seg* version of the architecture with *direct grouping* for our task.
- *PointNet++ No Time*: An ablation of TPointNet++ that removes the PointNet component. This leaves PointNet++ processing each frame independently followed by the shared MLP, and therefore has no notion of time.

Method	Category	Spatial Err	Time Err
MeteorNet	Cars	0.0633	**0.0001**
PointNet++ No Time		0.0530	—
PointNet++ w/ Time		0.0510	0.0005
PointNet		0.0250	0.0012
TPointNet++ No Time		0.0122	—
TPointNet++	Cars	**0.0118**	0.0011
TPointNet++	Chairs	0.0102	0.0008
TPointNet++	Airplanes	0.0064	0.0009

Table 3.1: Canonicalization performance.

- *PointNet++ w/ Time*: This is the same ablation as above, but modified so that the PointNet++ receives the timestamp of each point as an additional input feature. This baseline represents a naive way to incorporate time, but dilutes its contributions since it is only an auxiliary feature.
- *PointNet*: An ablation of TPointNet++ that removes the PointNet++ component. This leaves only PointNet operating on the full 4D spatiotemporal point cloud. This baseline treats time equally as the spatial dimensions, but inherently lacks local geometric features.
- *TPointNet++ No Time*: An ablation of TPointNet++ that only regresses the spatial part of the T-NOCS coordinate (and not the normalized timestamp). This baseline still takes the timestamps as input, but does not regress the last time coordinate.

Notably, for spatial prediction, TPointNet++ outperforms variations of both PointNet [204] and PointNet++ [205], along with their spatiotemporal extension MeteorNet [153]. This indicates that our ST design yields more distinctive features both spatially and temporally. MeteorNet and PointNet++ (with time) achieve impressive time errors thanks to skip connections that pass the input timestamps directly towards the end of the network. Qualitative results of canonicalization are in Fig. 3.4. Fig. 3.6(c) shows canonicalization results using TPointNet++ trained jointly within the full CaSPR model rather than individually.

Representation and Reconstruction: We evaluate CaSPR's ability to represent and reconstruct observed and unobserved frames of *raw* partial point cloud sequences. The full model is trained on each category separately using both $\mathcal{L}_r$ and $\mathcal{L}_c$, and is compared to two baselines. The first is a variation of CaSPR where the CNF is replaced with an AtlasNet [79] decoder using 64 patches – an alternative approach to achieve spatial continuity. This model is trained with $\mathcal{L}_c$ and a CD loss (rather than $\mathcal{L}_r$).

		10 Observed		*3 Observed*		*7 Unobserved*	
Method	**Category**	**CD**	**EMD**	**CD**	**EMD**	**CD**	**EMD**
PointFlow	Cars	**0.454**	12.838	**0.455**	12.743	**0.525**	13.911
CaSPR-Atlas		0.492	19.528	0.540	22.099	0.530	19.635
CaSPR		0.566	**10.103**	0.590	**11.464**	0.584	**11.259**
PointFlow	Chairs	0.799	17.267	0.796	17.294	0.950	18.442
CaSPR-Atlas		**0.706**	48.665	0.723	48.912	0.749	47.322
CaSPR		0.715	**13.009**	**0.681**	**13.310**	**0.683**	**13.564**
PointFlow	Airplanes	0.251	9.500	0.252	9.534	0.281	9.814
CaSPR-Atlas		0.237	18.827	0.255	18.525	0.269	17.933
CaSPR		**0.231**	**6.026**	**0.215**	**6.144**	**0.216**	**6.175**

Table 3.2: Partial surface sequence reconstruction. Chamfer (CD) and Earth Mover's Distances (EMD) are multiplied by 10^3. On the left (*10 Observed*), 10 frames are given as input and all are reconstructed. On the right, 3 frames are used as input (*3 Observed*), but methods also reconstruct intermediate unseen steps (*7 Unobserved*).

The second baseline is the deterministic PointFlow [287] autoencoder trained to reconstruct a *single canonical* partial point cloud. This model operates on a single timestep and receives the **already canonical** point cloud as input: an easier problem. We achieve temporal continuity with PointFlow by first encoding a pair of adjacent observed point clouds to derive two shape features, and then linearly interpolating to the desired timestamp – one alternative to attain temporal continuity. The interpolated feature conditions PointFlow's CNF to sample the partial surface, similar to CaSPR.

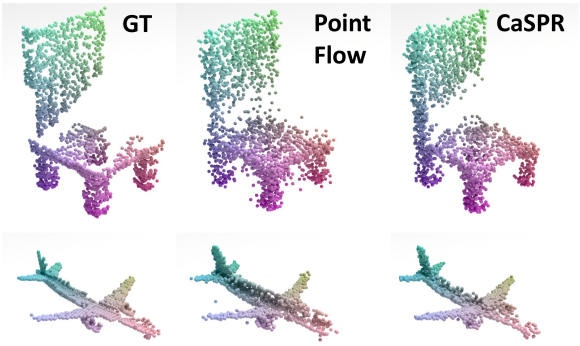

Figure 3.5: Reconstruction results. CaSPR accurately captures occlusion boundaries for camera motion at observed and unobserved timesteps, unlike linear feature interpolation with PointFlow.

Tab. 3.2 reports median CD and EMD at reconstructed test steps for each method. We evaluate two cases: (1) models receive and reconstruct all 10 observed frames (left), and (2) models get the first, middle, and last steps of a sequence and reconstruct both these 3 observed and 7 unobserved frames (right). CaSPR outperforms PointFlow in most cases, even at observed timesteps, despite operating on raw point clouds in the world frame instead of canonical. Because PointFlow reconstructs each frame independently, it lacks temporal context resulting in degraded occlusion boundaries (Fig. 3.5) and thus higher EMD. CaSPR gives consistent errors across observed and unobserved frames due to the learned motion prior of the Latent ODE, in contrast to linear feature interpolation that sees a

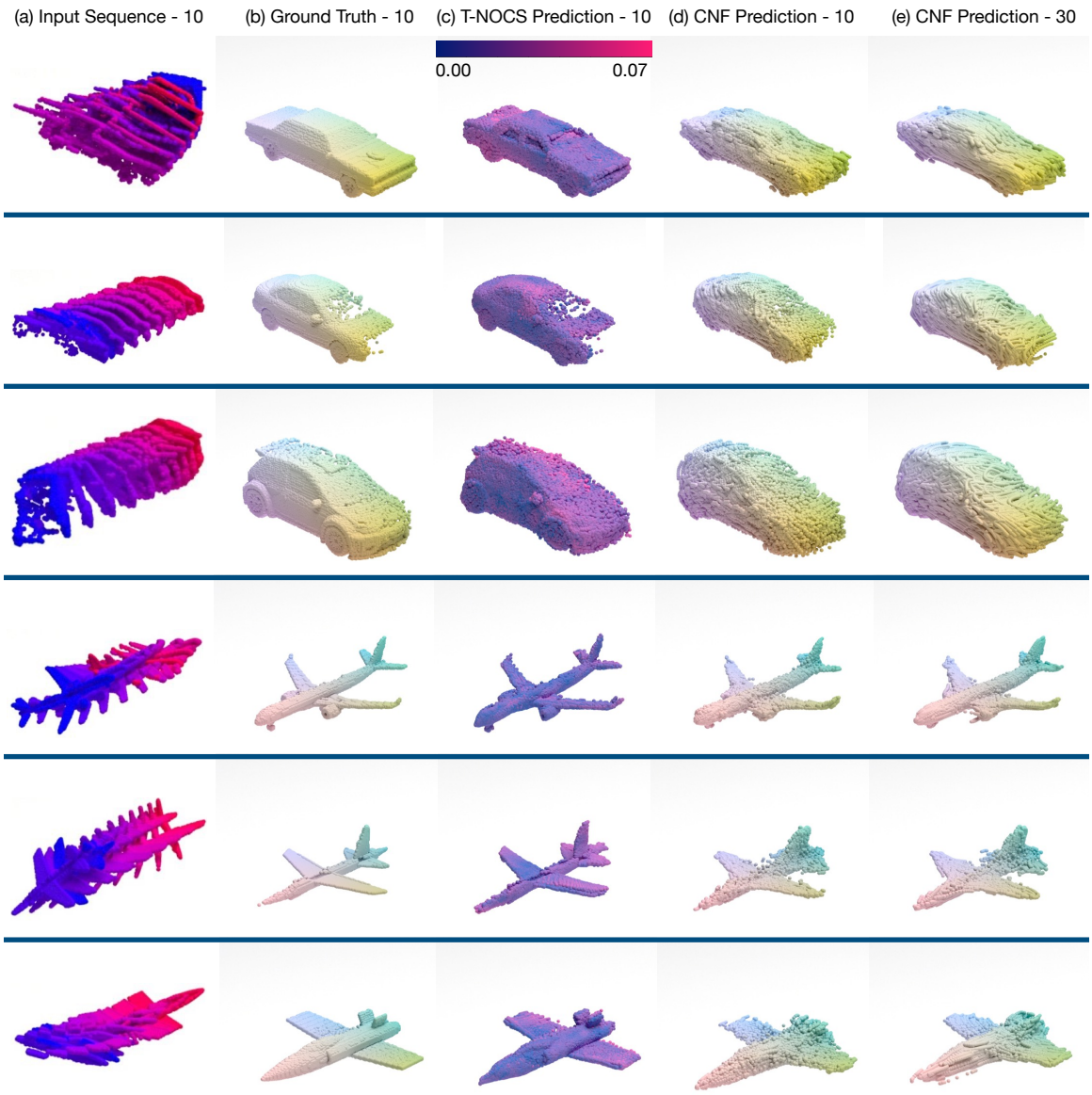

Figure 3.6: Canonicalization, aggregation, and dense reconstruction of rigid motion sequences by the full CaSPR model. Each sequence shows (a) the 10 observed raw partial point cloud frames given as input to CaSPR, (b) the GT partial reconstruction based on the observed frames, (c) the partial reconstruction achieved by aggregating T-NOCS predictions from TPointNet++ with color mapped to spatial error, (d) the aggregated prediction after reconstructing the 10 observed frames with the CNF, and (e) the aggregated prediction when interpolating 30 frames using the CNF.

marked performance drop for unobserved frames. The AtlasNet decoder achieves small CD since this is the primary training loss, but has difficulty reconstructing the correct point distribution on the

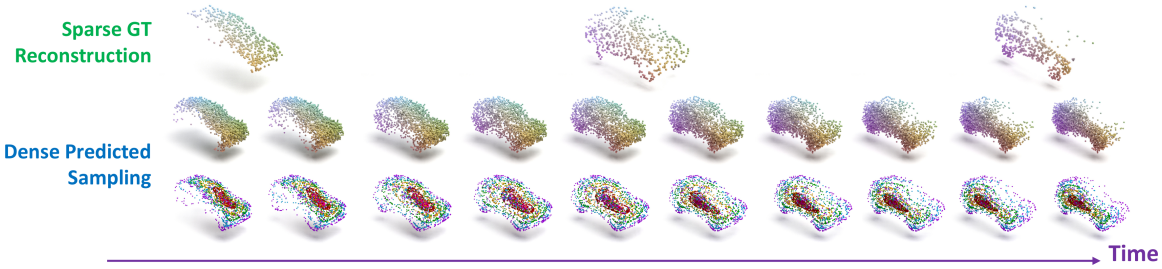

Figure 3.7: Continuous interpolation results. From three sparse frames of input with GT canonical points shown on top, CaSPR reconstructs the sequence more densely in space and time (middle). Contours of the Gaussian flowed to the car surface are shown on bottom (red is highest probability).

partial surface due to the patch-based approach, resulting in much higher EMD for all cases.

Fig. 3.6(d) shows qualitative results of aggregated shape after reconstructing the 10 observed frames with CaSPR.

Multiview Reconstruction: A direct application of TPointNet++ is partial shape reconstruction of observed geometry through a union of predicted T-NOCS spatial points. Due to the quantitative accuracy of TPointNet++ at each frame (Tab. 3.1), aggregated results closely match GT for unseen instances in all categories as shown in Fig. 3.4 (left) and Fig. 3.6(c).

Rigid Pose Estimation: The world–canonical 3D point correspondences from TPointNet++ allow fitting rigid object (or camera) pose at observed frames using RANSAC [68]. Tab. 3.3 reports median test errors showing TPointNet++ is competitive with RPM-Net [290], a recent **specialized** architecture for robust iterative rigid registration. Note here, RPM-Net takes *both* the raw depth and **GT** T-NOCS points as input. Translation and rotation errors are the distance and degree angle difference from the GT transformation. Point error measures the per frame median distance between the GT T-NOCS points transformed by the predicted pose and the input points. Qualitative results are in Fig. 3.4 (right).

Method	Category	Trans Err	Rot Err(°)	Point Err
RPM-Net	Cars	**0.0049**	**1.1135**	**0.0066**
CaSPR		0.0077	1.3639	0.0096
RPM-Net	Chairs	**0.0026**	**0.4601**	**0.0036**
CaSPR		0.0075	1.5035	0.0091
RPM-Net	Airplanes	**0.0040**	**0.5931**	**0.0048**
CaSPR		0.0051	0.9456	0.0057

Table 3.3: Pose estimation using T-NOCS.

Rigid Spatiotemporal Interpolation: The full CaSPR model can densely sample a sparse input sequence in spacetime as shown in Fig. 3.7. The model takes three input frames of 512 points (corresponding GT T-NOCS points shown on top) and reconstructs an arbitrary number of steps with 2048 points (middle). The representation can be sampled at any ST resolution but, in practice, is limited by memory. The CNF maps Gaussian noise to the visible surface (bottom). Points are

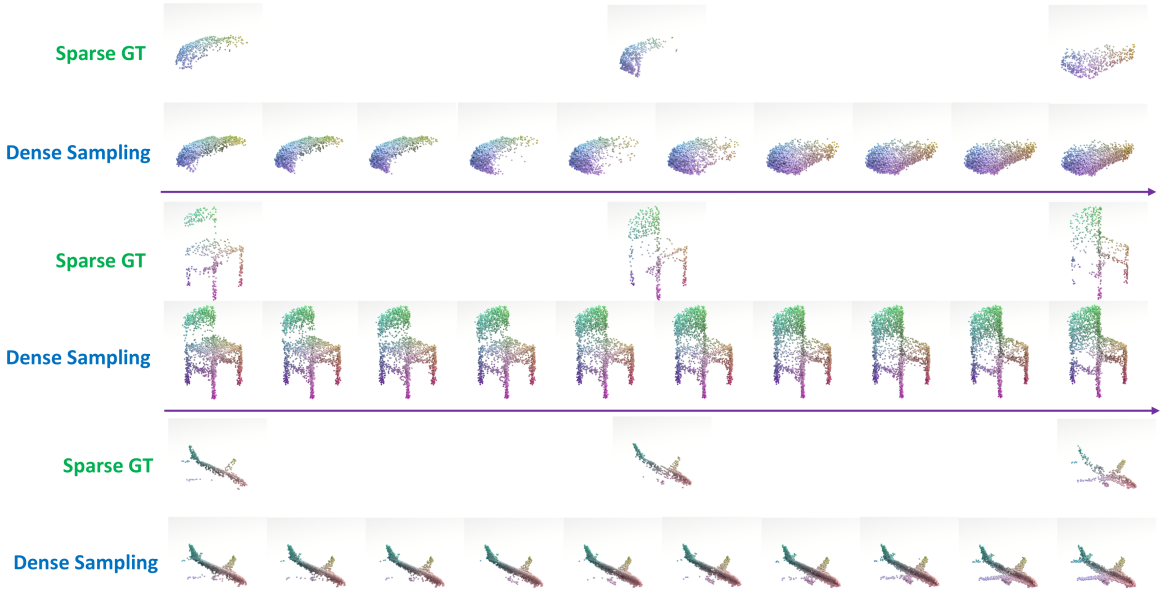

Figure 3.8: Additional examples of spatiotemporal interpolation for sparse sequences.

most dense in high probability areas (shown in red); in our data this roughly corresponds to where the camera is focused on the object surface at that timestep. Fig. 3.8 shows additional interpolation results from sparse input sequence. Fig. 3.6(d) shows qualitative results of aggregated shape after interpolating to 30 observed frames with CaSPR for more dense inputs from rigid object motion.

Non-Rigid Reconstruction and Temporal Correspondences: CaSPR can represent and reconstruct deformable objects. We evaluate on a variation of the Warping Cars dataset introduced in Occupancy Flow (OFlow) [177] which contains 10-step sequences of *full* point clouds sampled from ShapeNet [32] cars deforming over time. The sequences in this dataset are already consistently aligned and scaled, so CaSPR is trained only using $\mathcal{L}_r$.

	Reconstruction		*Correspondences*	
Method	**CD**	**EMD**	**Dist** t_1	**Dist** t_{10}
OFlow	1.512	20.401	**0.011**	**0.031**
CaSPR	**0.955**	**11.530**	0.013	0.035

Table 3.4: Reconstructing 10 observed timesteps (left) and maintaining temporal correspondences (right) on Warping Cars.

Tab. 3.4 compares CaSPR to OFlow on reconstructing deforming cars at 10 observed time steps (left) and on estimating correspondences over time (right). To measure correspondence error, we (1) sample 2048 points from the representation at $\bar{t}_1$, (2) find their closest points on the GT mesh, and (3) advect the samples to $\bar{t}_{10}$ and measure the mean distance to the corresponding GT points at both steps. Tab. 3.4 reports median errors over all $\bar{t}_1$ and $\bar{t}_{10}$. For OFlow, samples are advected using the

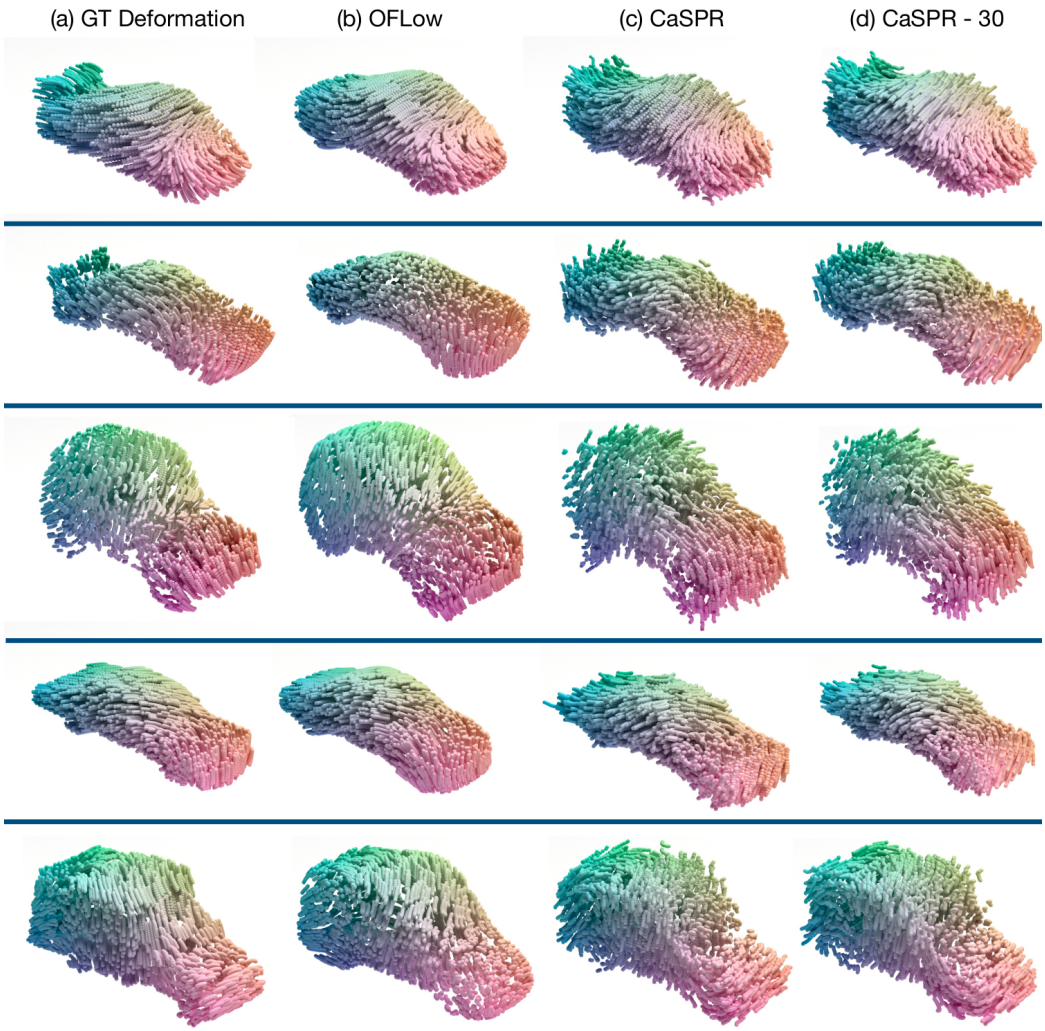

Figure 3.9: Reconstruction results on Warping Cars data. Each sequence is 10 steps in length and we show point trajectories over time for (a) the ground truth input sequence, (b) the reconstruction from Occupancy Flow, (c) the reconstruction at the 10 observed steps with CaSPR, and (d) 30 interpolated steps with CaSPR.

predicted flow field in physical space, while for CaSPR we simply use the same Gaussian samples at each step of the sequence.

CaSPR outperforms OFlow on reconstruction due to overly-smoothed outputs from the occupancy network, while both methods accurately maintain correspondences over time. Note that CaSPR advects system state in a learned latent space and temporal correspondences *naturally emerge* from the CNF when using consistent base samples across timesteps. Fig. 3.9 visualizes sampled point trajectories compared to OFlow for several sequences.

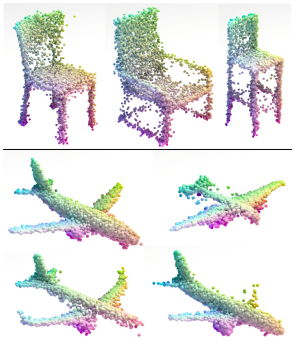

Figure 3.10: Cross-instance correspondences emerge naturally using a CNF.

Cross-Instance Correspondences: We observe consistent behavior from the CNF across objects within a shape category too. Fig. 3.10 shows reconstructed frames from various chair and airplane sequences with points colored by their corresponding location in the sampled Gaussian (before the flow). Similar colors across instances indicate the same part of the base distribution is mapped there. This could potentially be used, for instance, to propagate labels from known to novel object instances.

Learning the Arrow of Time: A desirable property of ST representations is an understanding of the *unidirectionality* of time [61]: how objects evolve forward in time. We demonstrate this property with CaSPR by training on a dataset of 1000 sequences of a single rigid car where the camera always rotates counter-clockwise at a fixed distance (but random height). CaSPR achieves a median CD of 0.298 and **EMD of 7.114** when reconstructing held-out sequences *forward in time*. However, when the same test sequences are *reversed* by flipping the timestamps, accuracy drastically drops to CD 1.225 and **EMD 88.938**. CaSPR is sensitive to the arrow of time due to the directionality of the Latent ODE and the global temporal view provided by operating on an entire sequence jointly.

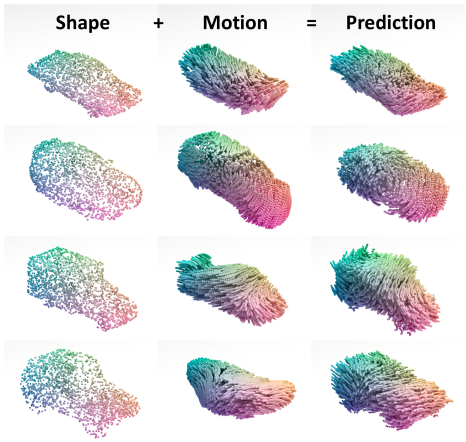

Figure 3.11: Disentanglement examples on warping cars data.

Shape & Motion Disentanglement: We evaluate how well CaSPR disentangles shape and motion as a result of the latent feature splitting $\mathbf{z}^C = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}^C_{\text{dyn}}]$. For this purpose, we transfer motion between two sequences by embedding both of them using TPointNet++, then taking the static feature $\mathbf{z}^C_{\text{ST}}$ from the first and the dynamic feature $\mathbf{z}^C_{\text{dyn}}$ from the second. Fig. 3.11 shows qualitative results where each row is a different sequence; the first frame of the shape sequence is on the left, the point trajectories of the motion sequence in the middle, and the final CaSPR-sampled trajectories using the combined feature are on the right. If these features perfectly disentangle shape and motion, we should see the shape of the first sequence with the motion of the second after reconstruction. Apparently, the explicit feature split in CaSPR does disentangle static and dynamic properties of the object to a large extent.

Label Propagation through Canonicalization: We evaluate the ability of T-NOCS canonicalization to establish correspondences by propagating point-wise labels both throughout a sequence and to

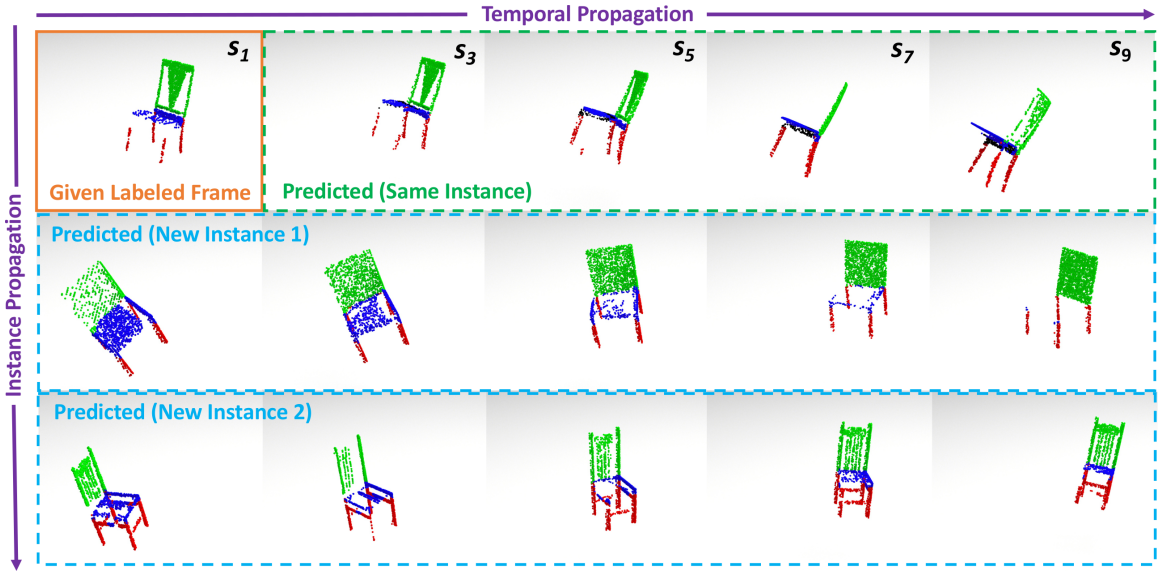

Figure 3.12: Example of semantic segmentation label propagation over time and across instances through T-NOCS canonicalization. The given labels in the first frame of the top sequence (orange box) are transferred to later frames in the same sequence (green dashed box) and to other sequences with different object instances (blue dashed boxes) by comparing to the labeled frame in the shared canonical space.

new sequences of different object instances. Given a semantic segmentation of the partial point cloud at the *first* frame of a sequence at time s_1, the first task is to label all subsequent steps in the sequence at times $s_2, \ldots, s_k$, *i.e.* propagate the segmentation forward in time. Secondly, we want to label all frames of sequences containing *different* object instances *i.e.* propagate the segmentation to different objects of the same class. We achieve both through canonicalization with TPointNet++: all frames in each sequence are mapped to T-NOCS, then unknown points are labeled by finding the closest point in the given labeled frame at s_1. If the closest point in s_1 is not within a distance of 0.05 in the canonical space, it is marked "Unknown". This may happen if part of the shape is not visible in the first frame due to self-occlusions.

Results of this label propagation for a subset of the chairs (1315 sequences) and airplanes (1215 sequences) categories of the rigid motion test set are shown in Tab. 3.5. We report median point-wise accuracy over all points (*Total Acc*) and for points successfully labeled by our approach (*Known Acc*). For the instance propagation task, we randomly use $1/3$ of test sequences as "source" sequences where the first frame is labeled, and the other $2/3$ are "target" sequences to which labels are propagated. In this case, accuracy is reported only for target sequences. Qualitative results are shown in Fig. 3.12.

Figure 3.13: Failure cases of CaSPR. The CNF has difficulty capturing local details and very thin structures (left) along with uncommon shapes (middle). TPointNet++ has trouble with symmetry or ambiguity in partial views, resulting in reflected or rotated predictions (right).

3.6 Discussion

We introduced CaSPR, a method to canonicalize and obtain object-centric representions of raw point cloud sequences, which supports spatiotemporal sampling at arbitrary resolutions. We demonstrated CaSPR's utility on rigid and deformable object motion and in applications like spatiotemporal interpolation and estimating correspondences across time and instances.

Task	Category	Total Acc	Known Acc
Temporal Propagation	Chairs	0.9419	0.9804
	Airplanes	0.9580	0.9676
Instance Propagation	Chairs	0.6553	0.8425
	Airplanes	0.7744	0.8006

Table 3.5: Segmentation label propagation performance. *Total Acc* is point-wise accuracy over all points; *Known Acc* is only for points that our method successfully labels.

Of the challenges outlined in Sec. 1.1.1, CaSPR dealt primarily with achieving *accuracy*, *generalizability*, and *robustness* in order to successfully enable point cloud perception tasks. These were addressed jointly by first canonicalizing the input point cloud to remove extrinsic factors, making it easier to model the intrinsic change in shape. This also allowed CaSPR to operate directly on partial point clouds in the frame of the sensor. The state space of the motion model was a learned latent space that demonstrated flexibility to both rigid and deformable objects. Finally, the motion was modeled using a neural ODE, which gave smooth latent dynamics that are continuous in time.

Limitations and Future Work: CaSPR leaves ample room for future exploration. We currently only support batch processing, but online processing is important for real-time applications. Additionally, CaSPR is expensive to train. Our canonicalization step requires dense supervision of T-NOCS labels which may not be available for real data. While the network is well-suited for ST interpolation, the extrapolation abilities of CaSPR need further investigation. CaSPR is object-centric, and further work is needed to generalize to object collections and scenes.

Moreover, CaSPR does fail in some specific cases. As shown in Fig. 3.13 (middle), outlier shapes

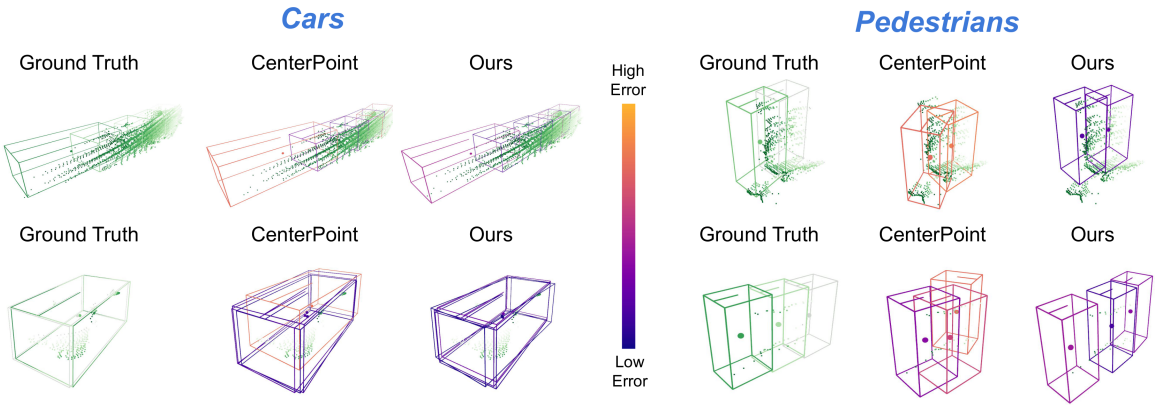

Figure 3.14: Qualitative results of spatiotemporal sequence refinement in SpOT. Refinement improves the temporal consistency of detections, especially for sparse sequences. Predicted bounding boxes are colored according to their L2 center error.

can cause noisy sampling results, and if the partial view of an object is ambiguous or the object is symmetric, TPointNet++ may predict a flipped or rotated canonical output as seen in Fig. 3.13 (right). Finally, using a single CNF for spatial sampling is fundamentally limited by an inability to model changes in topology [45, 60]. We observe this in chairs with back slats and other thin structures that are not captured by our Reconstruction CNF as shown in the left panel of Fig. 3.13. To capture fine-scale geometric details of shapes, this must be addressed.

3.7 Additional Related Contributions

This section briefly describes additional contributions made to the area of 3D object motion modeling and perception. Sec. 3.7.1 introduces SpOT [243], which tackles 3D multi-object tracking by improving TPointNet++ to refine object tracklets containing both points and bounding boxes. Then, Sec. 3.7.2 summarizes a method for predicting future 3D object states based on an initial point cloud input and velocities[211].

3.7.1 Spatiotemporal Modeling for 3D Object Tracking

3D multi-object tracking aims to uniquely and consistently identify all mobile entities through time. Despite the rich spatiotemporal information available in this setting, current 3D tracking methods primarily rely on abstracted information and limited history, *e.g.* single-frame object bounding boxes. In this work, we develop a holistic representation of traffic scenes that leverages both spatial and temporal information of the actors in the scene. Our method, SpOT (**Sp**atiotemporal **O**bject **T**racking),

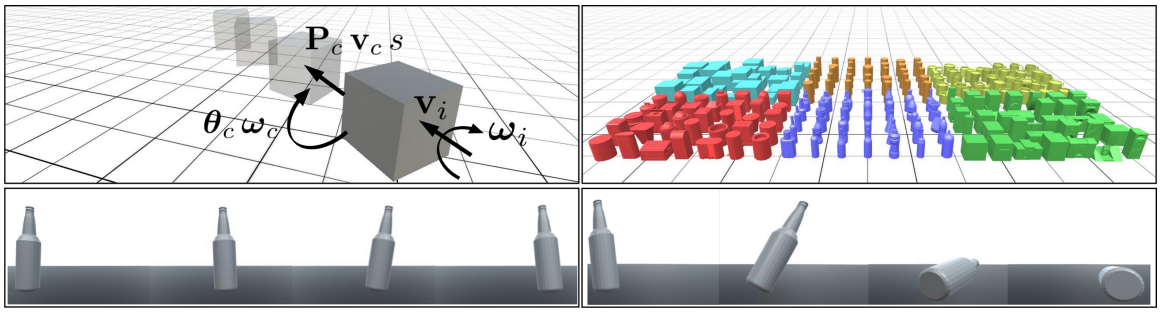

Figure 3.15: Overview of predicting future 3D object motion from a point cloud input with linear and angular velocities, $\mathbf{v}_i$ and ω_i (top left). Our goal is to predict, at each fixed time step, the change in object state: 3D position ($\mathbf{P}_c$), rotation (θ_c), linear and angular velocities ($\mathbf{v}_c, \omega_c$), and stability (s). Our method can predict the dynamics of a variety of different shapes (top right), generalizes to previously unseen object shapes and initial velocities, and tackles challenges such as *wobbling* (bottom left) and *toppling* (bottom right) of moving objects.

reformulates tracking as a spatiotemporal problem by representing tracked objects as sequences of time-stamped points and bounding boxes over a long temporal history. At each timestamp, we improve the location and motion estimates of our tracked objects through learned refinement over the full sequence of object history. By considering time and space jointly, our representation naturally encodes fundamental physical priors such as object permanence and consistency across time. Our spatiotemporal tracking framework achieves state-of-the-art performance on the Waymo and nuScenes benchmarks.

As shown in Fig. 3.14, sequence refinement in SpOT learns a prior on object motion that is key to getting temporally-consistent tracking in 3D point clouds. For full technical details and additional results, please refer to the paper [243].

3.7.2 Predicting the Future Motion of 3D Objects

Machines that can predict the effect of physical interactions on the dynamics of previously unseen object instances are important for creating better robots and interactive virtual worlds. In this work, we focus on predicting the dynamics of 3D objects on a plane that have just been subjected to an impulsive force. In particular, we predict the changes in state—3D position, rotation, velocities, and stability. Different from previous work, our approach can generalize dynamics predictions to object shapes and initial conditions that were unseen during training. Our method takes the 3D object's shape as a point cloud and its initial linear and angular velocities as input. We extract shape features and use a recurrent neural network to predict the full change in state at each time step. Our model

can support training with data from both a physics engine or the real world. Experiments show that we can accurately predict the changes in state for unseen object geometries and initial conditions.

Fig. 3.15 provides an overview of the key results of our method. Please refer to the paper for full technical details and results [211].

Chapter 4

Learned Traffic Model for Scenario Generation

Chapters 2 and 3 have shown that learned models of motion can greatly benefit perception tasks involving 3D humans and objects, respectively. For example, with HuMoR we leveraged a VAE model as a prior to recover human pose from video. As a generative model, this VAE, along with any other type of generative motion model, is also useful for synthesizing motions. However, when it comes to synthesizing motion rather than perceiving it, there are several modeling aspects that take on increased importance.

First is the need to consider *interactions* between entities and with the environment. Besides HuMoR modeling contacts with a single ground plane, interactions have been largely ignored in the work presented up to this point. For humans, it is also no longer sufficient to model motion while ignoring the high-level *intent* of a person. In HuMoR, we could randomly sample sequences of full-body poses that are physically-plausible, but they would consist of random actions that appear incoherent in terms of high-level behavior. While this formulation is generalizable and useful for motion perception, using it to synthesize human motion in simulation will result in unrealistic and erratic behavior. Finally, since generative applications are often creative, a user needs to have the ability to *control* various aspects of motion synthesized from the model.

In Chapters 4 and 5, we will explore these aspects of motion modeling in the context of simulating high-level human behavior. This involves predicting 2D top-down trajectories for both vehicles and pedestrians, which move based on interaction constraints (*e.g.*, avoiding collisions with each other and following traffic laws). This kind of behavior simulation is crucial to applications that replicate the real world with dynamic agents including vehicles, cyclists, and pedestrians. A particular focus

of this chapter is the application to autonomous vehicles (AVs), where it is necessary to simulate realistic people and traffic to develop and evaluate safe driving policies. In this context, we will see how controllability is important to (1) create scenarios to test specific aspects of an AV (this chapter), and (2) add pedestrians that follow goals and realistically travel in social groups (Chapter 5). Moreover, in Chapter 5 we will see that modeling behavior as 2D trajectories is even enough to produce full-body 3D pedestrian animations.

In this chapter, we will introduce STRIVE, a method to generate accident scenarios for AV planners that is a built on a learned model of traffic motion. This work was originally published in CVPR 2022 [210].

4.1 Introduction

The safety of contemporary autonomous vehicles is defined by their ability to safely handle complicated near-collision scenarios. However, these kinds of scenarios are rare in real-world driving, posing a data-scarcity problem that is detrimental to both the development and testing of data-driven models for perception, prediction, and planning. Moreover, the better models become, the more rare these events will be, making the models even harder to train.

A natural solution is to synthesize difficult scenarios in simulation, rather than relying on real-world data, making it easier and safer to evaluate and train AV systems. This approach is especially appealing for *planning*, where the appearance domain gap is not a concern. For example, one can manually design scenarios where the AV may fail by inserting adversarial actors or modifying trajectories, either from scratch or by perturbing a small set of real scenarios. Unfortunately, the manual nature of this approach quickly becomes prohibitively expensive when a large set of scenarios is necessary for training or comprehensive evaluation.

Recent work looks to *automatically generate* challenging scenarios [276, 1, 55, 56, 268, 179, 124]. Generally, these approaches control a single or small group of "adversaries" in a scene, define an objective (*e.g.* cause a collision with the AV), and then optimize the adversaries' behavior or trajectories to meet the objective. While most methods demonstrate generation of only 1 or 2 scenarios [1, 33, 124, 179], recent work [276] has improved scalability by starting from real-world traffic scenes and perturbing a limited set of pre-chosen adversaries. However, these approaches *lack expressive priors over plausible traffic motion*, which limits the realism and diversity of scenarios. In particular, adversarial entities in a scenario are a small set of agents heuristically chosen ahead of time; surrounding traffic will not be reactive and therefore perturbations must be careful to avoid

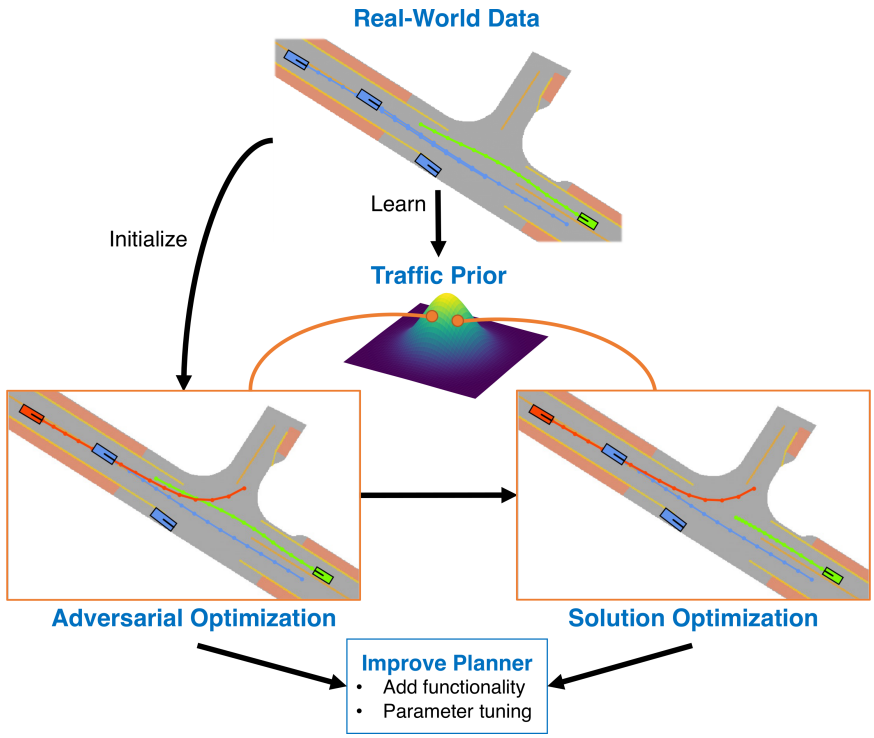

Figure 4.1: STRIVE generates challenging scenarios for a given planner. An *adversarial optimization* perturbs a real-world scene in the latent space of a learned traffic model, causing an adversary (red) to collide with the planner (green). A subsequent *solution optimization* finds a planner trajectory to avoid collisions, verifying a scenario is useful for identifying planner improvements.

implausible situations (*e.g.* collisions with auxiliary agents). Furthermore, less attention has been given to determining if a scenario is "unsolvable" [76], *i.e.*, if even an oracle AV is incapable of avoiding a collision. In this degenerate case, the scenario is not *useful* for evaluating/training a planner.

In this chapter, we introduce STRIVE – a method for generating challenging scenarios to **S**tress-**T**est d**RIVE** a given AV system. STRIVE attacks the prediction, planning, and control subset of the AV stack, which we collectively refer to as the *planner*. As shown in Fig. 4.1, our approach perturbs an initial real-world scene through an optimization procedure to cause a collision between an arbitrary adversary and a given planner. Our core idea is to measure the plausibility of a scenario during optimization by its likelihood under a learned generative model of traffic motion, which encourages scenarios to be challenging, yet realistic. As a result, STRIVE does not choose specific adversaries ahead of time, rather it jointly optimizes all scene agents, enabling a diverse set of scenarios to arise. Moreover, in order to accommodate for non-differentiable (or inaccessible) planners, which are

widely used in practice, the proposed optimization uses a differentiable proxy representation of the planner within the learned motion model, thus allowing standard gradient-based optimization to be used.

We propose to identify and characterize generated scenarios that are *useful* for improving a given planner. We first search for a "solution" to generated scenarios to determine if they are degenerate, and then cluster solvable scenarios based on collision properties. We test STRIVE on two AV planners, including a new rule-based planner, and show that it generates plausible and diverse collision scenarios in both cases. We additionally use generated scenarios to improve the rule-based planner by identifying fundamental limitations of its design and tuning hyperparameters.

In short, our contributions are: (i) a method to automatically generate plausible challenging scenarios for a given planner, (ii) a solution optimization to ensure scenario utility, and (iii) an analysis method to cluster scenarios by collision type. Supplementary videos for this work are available on the project webpage.

4.2 Related Work

Traffic Motion Modeling. Scenario replay is insufficient for testing and developing AV planners as the motion of non-ego vehicles is strongly coupled to the actions chosen by the ego planner. Advances in deep learning have allowed us to replace traditional dynamic and kinematic models [272, 130, 138] or rule-based simulators [156, 59] with neural counterparts that better capture traffic complexity [9, 182]. Efforts to predict future trajectories from a short state history and an HD map can generally be categorized according to the encoding technique, modeling of multi-modality, multi-agent interaction, and whether the trajectory is estimated in a single step or progressively. The encoding of surrounding context of each agent is often done via a bird's-eye view (BEV) raster image [47, 31, 65], though some work [146, 72] replaces the rasterization-based map encoding with a lane-graph representation. SimNet [14] increased the diversity of generated simulations by initializing the state using a generative model conditioned on the semantic map. To account for multi-modality, multiple futures have been estimated either directly [47] or through trajectory proposals [31, 65, 154, 198]. Modeling multi-actor interactions explicitly using dense graphs has proven effective for vehicles [29, 248], lanes [146], and pedestrians [109, 228, 131]. Finally, step-by-step prediction has performed favorably to one-shot prediction of the whole trajectory [57]. We follow these works and design a traffic model that uses an inter-agent graph network [112] to represent agent interaction and is variational, allowing us to sample multiple futures.

Our model builds on VAE-based approaches [29, 248] that provide a learned prior over a controllable latent space [206]. Among other design differences, we incorporate a penalty for environment collisions and structure predictions through a bicycle model to ensure physical plausibility.

Challenging Scenario Generation. Generating scenarios has the potential to exponentially increase scene coverage compared to relying exclusively on recorded drives. Advances in photo-realistic simulators like CARLA [59] and NVIDIA's DRIVE Sim, along with the availability of large-scale datasets [26, 246, 63, 118, 105], have been instrumental to methods that generate plausible scene graphs to improve perception [116, 53, 213] and planning [14, 29, 248, 120]. Our work focuses on generating challenging – or "adversarial"[1] – scenarios, which are even more crucial since they are so rare in recorded data. While most works assume perfect perception and attack the planning module [33, 124, 55, 56, 268, 76], recent efforts exploit the full stack, including image or point-cloud perception [1, 179, 144, 276, 256]. Our work focuses on attacking the planner only, though our scene parameterization as a learned traffic model could be incorporated into end-to-end methods. Unlike our approach, which uses gradient-based optimization enabled by the learned motion model, most adversarial generation works rely heavily on black-box optimization which may be slow and unreliable.

Our scenario generation approach is most similar to AdvSim [276], however instead of optimizing acceleration profiles of a simplistic bicycle model we use a more expressive data-driven motion prior. This remedies the previous difficulty of controlling many adversarial agents simultaneously in a plausible manner. Moreover, we avoid constraining the attack trajectory to not collide with the playback AV by proposing a "solution" optimization stage to filter worthwhile scenarios. Prior work [33] clusters lane-change scenarios based on trajectories of agents, while we cluster based on collision properties between the adversary and planner.

AV Planners. Despite the recent academic interest in end-to-end learning-based planners and AVs [225, 302, 224, 30, 9], rule-based planners remain the norm in practical AV systems [271]. Therefore, we evaluate our approach on a rule-based planner similar to the lane-graph-based planners used by successful teams in the 2007 DARPA Urban Challenge [174, 257] detailed in Sec. 4.4.2.

4.3 Challenging Scenario Generation

STRIVE aims to generate high-risk traffic situations for a given *planner*, which can subsequently

[1] we use "challenging" to denote generation procedures that do not explicitly attack a specific module in the perception or planning stack

be used to improve that planner (Fig. 4.1). For our purpose, the planner encapsulates prediction, planning, and control, *i.e.* we are interested in scenarios where the system misbehaves even with perfect perception. The planner takes as input past trajectories of other agents in a scene and outputs the future trajectory of the vehicle it controls (termed the *ego* vehicle). It is assumed to be black-box: STRIVE has no knowledge of the planner's internals and cannot compute gradients through it. Undesirable behavior includes collisions with other vehicles and non-drivable terrain, uncomfortable driving (*e.g.* high accelerations), and breaking traffic laws. We focus on generating **accident-prone scenarios** involving vehicle-vehicle collisions with the planner, though our formulation is general and in principle can handle alternative objectives.

Similar to prior work [276], scenario generation is formulated as an optimization problem that perturbs agent trajectories in an initial scenario from real-world data. The input is a planner f, map $\mathcal{M}$ containing semantic layers for drivable area and lanes, and a sequence from a pre-recorded real-world scene that serves as initialization for optimization. This initial scenario contains N agents with trajectories represented in 2D BEV as $Y = \{\mathbf{Y}^i\}_{i=1}^N$, where $\mathbf{Y}^i = [\mathbf{y}_1^i, \mathbf{y}_2^i, \ldots, \mathbf{y}_T^i]$ is the sequence of states for agent i. We let $\mathbf{Y}_t = [\mathbf{y}_t^1, \mathbf{y}_t^2, \ldots, \mathbf{y}_t^N]$ be the state of all agents at a single timestep. An agent state $\mathbf{y}_t^i = [x_t, y_t, \theta_t, v_t, \dot{\theta}_t]$ at time t contains the 2D position (x_t, y_t), heading θ_t, speed v_t, and yaw rate $\dot{\theta}_t$. When rolled out within a scenario, at each timestep the planner outputs the next **ego** state $\mathbf{y}_t^{\text{plan}} = f(\mathbf{y}_{<t}^{\text{plan}}, \mathbf{Y}_{<t}, \mathcal{M})$ based on the *past* motion of itself and other agents. For simplicity, we will write the rolled out planner trajectory as $\mathbf{Y}^{\text{plan}} = f(Y, \mathcal{M})$ where $\mathbf{Y}^{\text{plan}} = [\mathbf{y}_1^{\text{plan}}, \mathbf{y}_2^{\text{plan}}, \ldots, \mathbf{y}_T^{\text{plan}}]$ for the remainder of this chapter. Scenario generation perturbs trajectories for all non-ego agents to best meet an adversarial objective $\mathcal{L}_{\text{adv}}$ (*e.g.* cause a collision with the planner):

$$\min_{Y} \mathcal{L}_{\text{adv}}(Y, \mathbf{Y}^{\text{plan}}), \quad \mathbf{Y}^{\text{plan}} = f(Y, \mathcal{M}). \tag{4.1}$$

One may optimize a single or small set of "adversaries" in Y explicitly, *e.g.* through the kinematic bicycle model parameterization [276, 130, 199]. While this enforces plausible single-agent dynamics, **interactions** must be constrained to avoid collisions between non-ego agents and, even then, the resulting traffic patterns may be unrealistic. We propose to instead *learn* to model traffic motion using a neural network and then use it at optimization time **(i) to parameterize** all trajectories in a scenario as vectors in the latent space, and **(ii) as a prior** over scenario plausibility. Next, we describe this traffic model, followed by the "adversarial" optimization that produces collision scenarios.

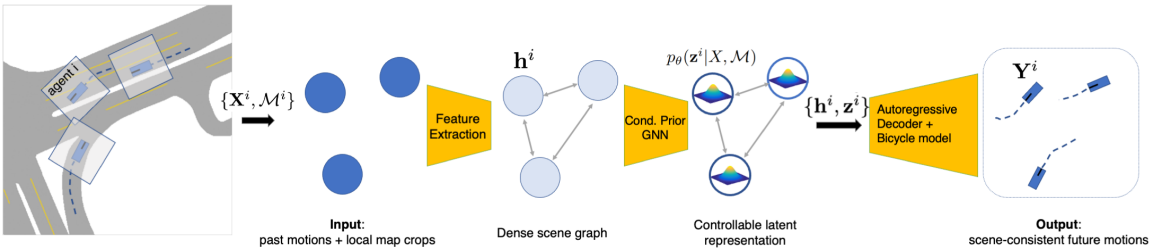

Figure 4.2: Test-time architecture of the learned traffic model. To jointly sample future trajectories for all agents in a scene, past motion and local map context is first processed individually for each agent. The *conditional prior*, then outputs a latent distribution at each node that can be sampled and fed through the *autoregressive decoder* to predict future agent trajectories.

4.3.1 Modeling "Realism": Learned Traffic Model

We wish to generate accident-prone scenarios that are assumed to develop over short time periods ($<$10 sec) [175]. Therefore, traffic modeling is formulated as *future forecasting*, which predicts future trajectories for all agents in a scene based on their past motion. We learn $p_\theta(Y|X, \mathcal{M})$ to enable sampling a future scenario Y conditioned on the fixed past $X = \{\mathbf{X}^i\}_{i=1}^N$ (defined similar to Y) and the map $\mathcal{M}$. Two properties of the traffic model make it particularly amenable to downstream optimization: a low-dimensional latent space for efficient optimization, and a prior distribution over this latent space to determine the plausibility of a given scenario. Inspired by recent work [29, 248], we design a conditional variational autoencoder (CVAE), shown in Fig. 4.2, that meets these criteria while learning accurate and scene-consistent joint future predictions. To sample future motions at test time, a *conditional prior* and *decoder* network are used. As detailed next, both are graph neural networks operating on a fully-connected scene graph of all agents.

Feature Extraction. Context features for each agent in the scene $\mathbf{h}^i = [\mathbf{p}^i, \mathbf{m}^i, \mathbf{s}^i]$ are first extracted based on: the past trajectory $\mathbf{X}^i$, the map $\mathcal{M}$, a one-hot encoding of the semantic class $\mathbf{s}^i$, and the agent's bounding box length/width $\mathbf{b}^i = (l, w)$. The past trajectory feature for each agent $\mathbf{p}^i \in \mathbb{R}^{64}$ is encoded from $\mathbf{X}^i$, $\mathbf{s}^i$, and $\mathbf{b}^i$ using a multi-layer perceptron (MLP). The map feature $\mathbf{m}^i \in 64$ is extracted using a convolutional network from a local rasterized map crop around the agent at the last step of $\mathbf{X}^i$. The map input contains a binary channel for each semantic layer (*e.g. drivable area*, *lane divider*, *etc.*).

Conditional Prior. The *prior* models $p_\theta(Z|X, \mathcal{M})$ where $Z = \{\mathbf{z}^i\}_{i=1}^N$ is a set of agent latent vectors. It operates on a fully-connected scene graph with a context feature $\mathbf{h}_i$ placed at each node. The prior (along with posterior and decoder explained below) is a graph neural network (GNN) similar

to a scene interaction module [29, 248]. It performs one round of message passing, which involves an *edge network*, *aggregation function*, and *update network*. Consider a single node i in the scene graph. First, interaction features are computed for every incoming edge. For an edge from node $j \to i$, the edge feature is computed using the *edge network* $\mathcal{E}$ as $\mathbf{e}^{ij} = \mathcal{E}(\mathbf{h}^i, \mathbf{h}^j, \mathcal{T}^{ij})$ where $\mathcal{T}^{ij}$ is the relative position and heading of agent j in the local frame of agent i. After computing all edge features, they are aggregated into a single interaction feature $\mathbf{e}^i$ using maxpooling $\mathbf{e}^i = \max(\mathbf{e}^{i1}, \mathbf{e}^{i2}, \dots)$. The update network then gives the output at each node $\mathbf{o}^i = \mathcal{U}(\mathbf{h}^i, \mathbf{e}^i)$.

After message passing, the *prior* outputs parameters of a Gaussian for each agent in the scene, forming a "distributed" latent representation that captures the variation in possible futures:

$$p_\theta(\mathbf{z}^i | X, \mathcal{M}) = \mathcal{N}(\mu^i_\theta(X, \mathcal{M}), \sigma^i_\theta(X, \mathcal{M})). \tag{4.2}$$

Decoder. The deterministic *decoder* $Y = d_\theta(Z, X, \mathcal{M})$ operates on the scene graph with both a sampled latent $\mathbf{z}^i$ and past context $\mathbf{h}^i$ at each node. Decoding is performed autoregressively: at timestep t, one round of message passing resolves interactions before predicting accelerations $\dot{v}_t, \ddot{\theta}_t$ for each agent. Accelerations immediately go through the kinematic bicycle model [130, 199] to obtain the next state $\mathbf{y}^i_{t+1}$, which updates $\mathbf{h}^i$ before continuing rollout. The determinism and graph structure of the decoder encourages scene-consistent future predictions even when agent $\mathbf{z}$'s are independently sampled. Importantly for latent optimization, the decoder ensures plausible vehicle dynamics by using the kinematic bicycle model, even if the input Z is unlikely.

Training. Training is performed on pairs of (X, Y_{gt}) using a modified CVAE objective:

$$\mathcal{L}_{\text{cvae}} = \mathcal{L}_{\text{recon}} + w_{\text{KL}}\mathcal{L}_{\text{KL}} + w_{\text{coll}}\mathcal{L}_{\text{coll}}. \tag{4.3}$$

To optimize this loss, a *posterior* network $q_\phi(Z | Y_{\text{gt}}, X, \mathcal{M})$ is introduced similar to the prior, but operating jointly on past and future motion. Future trajectory features are extracted separately, while past features are the same as used in the *prior*. The full training loss uses trajectory samples from both the posterior Y_{post} and prior Y_{prior}:

$$\mathcal{L}_{\text{recon}} = \sum_{i=1}^{N} ||\mathbf{Y}^i_{\text{post}} - \mathbf{Y}^i_{\text{gt}}||^2 \tag{4.4}$$

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(q_\phi(Z | Y_{\text{gt}}, X, \mathcal{M}) || p_\theta(Z | X, \mathcal{M})) \tag{4.5}$$

$$\mathcal{L}_{\text{coll}} = \mathcal{L}_{\text{agent}} + \mathcal{L}_{\text{env}} \tag{4.6}$$

where $\mathbf{Y}^i_{\text{post}} \in Y_{\text{post}}$, $\mathbf{Y}^i_{\text{gt}} \in Y_{\text{gt}}$, and D_{KL} is the KL divergence. Collision penalties $\mathcal{L}_{\text{agent}}$ and $\mathcal{L}_{\text{env}}$ use a differentiable approximation of collision detection to penalize Y_{prior} for collisions between agents or with the non-drivable map area. $\mathcal{L}_{\text{agent}}$ was introduced in TrafficSim [248] and penalizes vehicle-vehicle collisions by representing all N vehicles by disks. We estimate each agent vehicle i by 5 disks with radius r_i and compute the loss by summing over all pairs of agents (i, j) over time as:

$$\mathcal{L}_{\text{agent}} = \frac{1}{N^2} \sum_{(i,j), i \neq j} \sum_{t=1}^{T} \mathcal{L}_{\text{pair}}(\mathbf{y}^i_t, \mathbf{y}^j_t) \tag{4.7}$$

$$\mathcal{L}_{\text{pair}}(\mathbf{y}^i_t, \mathbf{y}^j_t) = \begin{cases} 1 - \dfrac{d}{r_i + r_j}, & d \leq r_i + r_j \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

where d is the minimum distance over all pairs of disks representing agents i and j.

$\mathcal{L}_{\text{env}}$ uses a similar idea to penalize collisions with the non-drivable area. At each step of rollout, collisions are detected between the ego vehicle and the non-drivable area, and a collision point $\mathbf{c}$ is determined as the mean of all vehicle pixels that overlap with non-drivable area. The loss is then calculated as:

$$\mathcal{L}_{\text{env}} = \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}_{\text{drivable}}(\mathbf{y}_t, \mathcal{M}) \tag{4.9}$$

$$\mathcal{L}_{\text{drivable}}(\mathbf{y}_t, \mathcal{M}) = \begin{cases} 1 - \dfrac{d}{d_{\text{max}}}, & \text{if partial collision} \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

where d is the distance between the vehicle position (center of bounding box) and collision point $\mathbf{c}$, and d_{max} is half the ego bounding box diagonal. Note the loss is only applied if there is a partial collision, *i.e.* only part of the bounding box overlaps with the non-drivable area – this is because if the vehicle is completely embedded in the non-drivable area, the loss will not give a useful gradient.

4.3.2 Adversarial Optimization

To leverage the learned traffic model, the real-world scenario used to initialize optimization is split into the past X and future Y_{init}. Throughout optimization, past trajectories in X (including that of the planner) are *fixed* while the future is perturbed to cause a collision with the given planner f. This perturbation is done in the learned latent space of the traffic model – as described below, we optimize

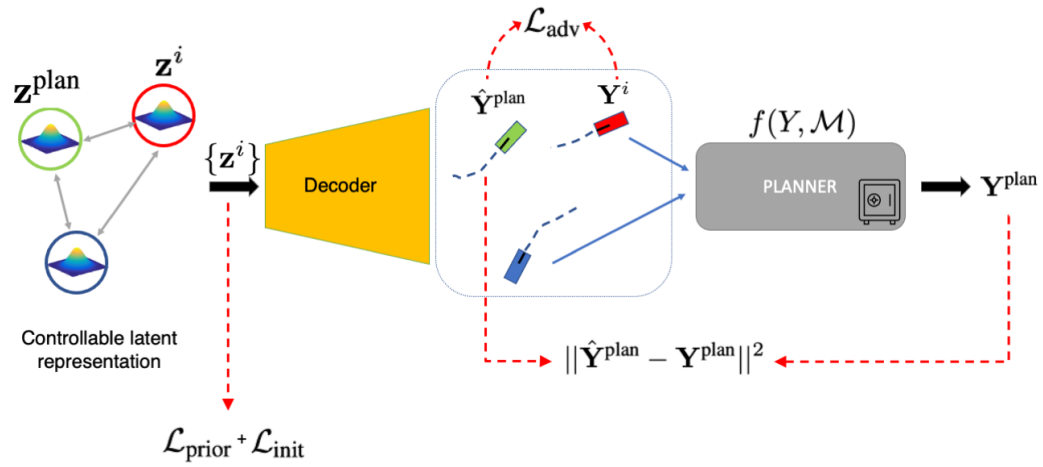

Figure 4.3: At each step of adversarial optimization, latent representations of both the planner and non-ego agents are decoded with the learned decoder and non-ego trajectories are given to the planner for rollout within the scenario. Finally, losses are computed.

the set of latents for all N *non-ego* agents $Z = \{\mathbf{z}^i\}_{i=1}^N$ along with a latent representation of the planner $\mathbf{z}^{\text{plan}}$.

Latent scenario parameterization encourages plausibility in two ways. First, since the decoder is trained on real-world data, it will output realistic traffic patterns if Z stays near the learned manifold. Second, the learned prior network gives a distribution over latents, which is used to penalize unlikely Z. This strong prior on behavioral plausibility enables jointly optimizing *all agents* in the scene rather than choosing a small set of specific adversaries in advance.

At each step of optimization (Fig. 4.3), the perturbed scenario is decoded with $d_\theta(Z, \mathbf{z}^{\text{plan}}, X, \mathcal{M})$ and non-ego trajectories Y are passed to the (black-box) planner, which rolls out the ego motion before losses can be computed. Adversarial optimization seeks two simultaneous objectives:

1. Match Planner. Although optimization has no direct control over the planner's behavior – an external function that is queried only when required – it is still necessary to represent the planner within the traffic model (*i.e.* include it in the scene graph with an associated latent $\mathbf{z}^{\text{plan}}$) so that interactions with other agents are realistic. In doing this, future predictions from the decoder include an estimate of the planner trajectory $\hat{\mathbf{Y}}^{\text{plan}}$ that, ideally, is close to the true planner trajectory $\mathbf{Y}^{\text{plan}} = f(Y, \mathcal{M})$. Note that this gives a *differentiable approximation* of the planner, enabling typical gradient-based optimization to be used for the second objective described below. To encourage

matching the real planner output with this "internal" approximation, we use

$$\min_{\mathbf{z}^{\text{plan}}} ||\hat{\mathbf{Y}}^{\text{plan}} - \mathbf{Y}^{\text{plan}}||^2 - \alpha \log p_\theta(\mathbf{z}^{\text{plan}}|X, \mathcal{M}) \tag{4.11}$$

where the right term lightly regularizes $\mathbf{z}^{\text{plan}}$ to stay likely under the learned prior and α balances the two terms.

2. Collide with Planner. The goal for non-ego agents is to cause the planner to collide with another vehicle:

$$\min_{Z} \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{init}} + \mathcal{L}_{\text{coll}}. \tag{4.12}$$

The *adversarial* term encourages a collision by minimizing the positional distance between controlled agents and the current traffic model approximation of the planner:

$$\mathcal{L}_{\text{adv}} = \sum_{i=1}^{N} \sum_{t=1}^{T} \delta_t^i \cdot ||\mathbf{y}_t^i - \hat{\mathbf{y}}_t^{\text{plan}}||^2 \tag{4.13}$$

$$\delta_t^i = \frac{\exp(-||\mathbf{y}_t^i - \hat{\mathbf{y}}_t^{\text{plan}}||)}{\sum_j \sum_t \exp(-||\mathbf{y}_t^j - \hat{\mathbf{y}}_t^{\text{plan}}||)} \tag{4.14}$$

where $\mathbf{y}_t$ here only includes the 2D position. Intuitively, the δ_t^i coefficients defined by the *softmin* in Eq. (4.14) are finding a candidate agent and timestep to collide with the planner. The agent with the largest δ_t^i is the most likely "adversary" based on distance, and Eq. (4.13) prioritizes causing a collision between this adversary and the planner while still allowing gradients to reach other agents. This weighting helps $\mathcal{L}_{\text{prior}}$ to avoid all agents unrealistically colliding with the planner.

The *prior* term encourages latents to stay likely under the learned prior network:

$$\mathcal{L}_{\text{prior}} = -\frac{1}{N} \sum_{i=1}^{N} \gamma^i \cdot \log p_\theta(\mathbf{z}^i|X, \mathcal{M}) \tag{4.15}$$

$$\gamma^i = 1 - \sum_t \delta_t^i. \tag{4.16}$$

The γ^i coefficient will weight likely adversaries near zero, *i.e.* agents close to colliding with the planner are allowed to deviate from the learned traffic manifold to exhibit rare and challenging behavior. Because the traffic model training data does not contain collisions, we found it difficult for an agent to collide with the planner using a large prior loss, thus motivating the weighting in Eq. (4.16). Note that even when γ^i is small, agents will maintain physical plausibility since the

decoder uses the kinematic bicycle model.

$\mathcal{L}_{\text{init}}$ encourages staying close to the initialization in latent space, since it is already known to be realistic:

$$\mathcal{L}_{\text{init}} = \frac{1}{N} \sum_{i=1}^{N} \gamma^i \cdot ||\mathbf{z}^i - \mathbf{z}^i_{\text{init}}||^2 \tag{4.17}$$

where $\mathbf{z}^i_{\text{init}} \in Z_{\text{init}}$ are the latents that initialize optimization. Finally, similar to CVAE training, $\mathcal{L}_{\text{coll}}$ discourages non-ego agents from colliding with each other and the non-drivable area. In practice, all loss terms are balanced by manual inspection of a small set of generated scenarios.

Initialization and Optimization. Given a real-world scene, Z_{init} is obtained through the posterior network q_ϕ, then further refined with an initialization optimization that fits to the input future trajectories of all agents (similar to Eq. (4.11)), including the initial planner rollout. Optimization is implemented in PyTorch[186] using ADAM[121] with a learning rate of 0.05. Runtime depends on the planner and number of agents; for our rule-based planner (see Sec. 4.4.2), a 10-agent scenario takes 6-7 minutes.

4.4 Analyzing and Using Generated Scenarios

4.4.1 Filtering and Collision Classification

Solution Optimization. Adversarial optimization produces plausible scenarios, but it cannot guarantee they are *solvable* and *useful*: *e.g.* a scenario in which the ego is squeezed by multiple cars produces an unavoidable collision and is therefore uninformative for evaluating or improving a planner. Therefore, we perform an additional optimization to identify an ego trajectory that avoids collision; if this optimization fails, the scenario is discarded for downstream tasks. This solution optimization is initialized from the output of the adversarial optimization and essentially inverts the objectives described in Sec. 4.3.2: non-ego latent Z are tuned to maintain the adversarial trajectories while $\mathbf{z}^{\text{plan}}$ is optimized to avoid collisions and stay likely under the prior.

Clustering and Labeling. To gain insight into the distribution of collision scenarios and inform their downstream use, we propose a simple approach to cluster and label them. Specifically, scenarios are characterized by the explicit relationship between the planner and adversary at the time of collision: the relative direction and heading of the adversary are computed in the frame of the planner and concatenated to form a collision feature for each scenario. These features are clustered with

k-means to form semantically similar groups of accidents that are labeled by visual inspection. Their distribution can then be visualized as in Fig. 4.6.

4.4.2 Improving the Planner

With a large set of labeled collision scenarios, the planner can be improved in two main ways. First, discrete improvements to functionality may be needed if many scenarios of the same type are generated. For example, a planner that strictly follows lanes is subject to collisions from head on or behind as it fails to swerve, indicating necessary new functionality to leave the lane graph. Second, scenarios provide data for tuning hyperparameters or learned parameters.

Rule-based Planner. To demonstrate how STRIVE scenarios are used for these kinds of improvements, we introduce a simple, yet competent, rule-based planner that we use as a proxy for a real-world planner. Our planner is ideal for evaluating STRIVE as it is easily interpretable, uses a small set of hyperparameters, and has known failure modes. It relies entirely on the lane graph to both predict future trajectories of non-ego vehicles and generate candidate trajectories for the ego vehicle. Among these candidates, it chooses that which covers the most distance with a low "probability of collision." In more detail, the planner has the following structure:

1. Extract from the lane graph a finite set of splines that each vehicle might follow.
2. Generate predictions for the future motion of non-ego vehicles along each of the splines from (1).
3. Generate candidate trajectories for the ego vehicle and use the predictions from (2) to estimate the "probability of collision" $p_{\text{col}}(\tau)$ for each candidate τ.
4. Among trajectories that are unlikely to collide $\{\tau \mid p_{\text{col}}(\tau) < p_{\text{max}}\}$, choose the trajectory that covers the most distance. If no trajectories are unlikely to collide, choose the trajectory that is least likely to collide.
5. Repeat every Δt seconds.

Note the "intent" of the planner is deterministic, *i.e.* it will always follow the same lane graph path (*e.g.* choosing whether to turn left or right) when rollout starts from the same initialization. Planner behavior is affected by hyperparameters such as how $p_{\text{col}}(\tau)$ is computed, p_{max}, and the maximum speed and forward acceleration. This planner has the additional limitation that it cannot change lanes, which scenarios generated by STRIVE exposes in Sec. 4.5.3.

4.5 Experiments

We next highlight the new capabilities that STRIVE enables. Sec. 4.5.1 demonstrates the ability to generate challenging and useful scenarios on two different planners; these scenarios contain a diverse set of collisions, as shown through analysis in Sec. 4.5.2. Generated scenarios are used to improve our rule-based planner in Sec. 4.5.3.

Dataset. The nuScenes dataset [26] is used both to train the traffic model and to initialize adversarial optimization. It contains $20s$ traffic clips annotated at 2 Hz, which we split into $8s$ scenarios. Only *car* and *truck* vehicles are used and the traffic model operates on the rasterized *drivable area*, *carpark area*, *road divider*, and *lane divider* map layers. We use the splits and settings of the nuScenes prediction challenge which is $2s$ (4 steps) of past motion to predict $6s$ (12 steps) of future, meaning collision scenarios are $8s$ long, but only the future $6s$ trajectories are optimized.

Planners. Scenario generation is evaluated on two different planners. The *Replay* planner simply plays back the ground truth ego trajectory from nuScenes data. This is an open-loop setting where the planner's $6s$ future is fully rolled out without re-planning. The *Rule-based* planner, described in Sec. 4.4.2, allows a more realistic closed-loop setting where the planner reacts to the surrounding agents during future rollout by re-planning at 5 Hz.

Metrics. The **collision rate** is the fraction of optimized initial scenarios from nuScenes that succeed in causing a planner collision, which indicates the sample efficiency of scenario generation. **Solution rate** is the fraction of these colliding scenarios for which a solution was found, which measures how often scenarios are *useful*. **Acceleration** indicates how comfortable a driven trajectory is; challenging scenarios should generally increase acceleration for the planner, while the adversary's acceleration should be reasonably low to maintain plausibility. If a scenario contains a collision, acceleration (and other trajectory metrics) is only calculated up to the time of collision. **Collision velocity** is the relative speed between the planner and adversary at the time of collision; it points to the severity of a collision.

4.5.1 Scenario Generation Evaluation

First, we demonstrate that STRIVE generates challenging, yet solvable, scenarios causing planners to collide and drive uncomfortably. Moreover, compared to an alternative generation approach that *does not* leverage the learned traffic prior, STRIVE scenarios are more plausible. Scenario generation is initialized from 1200 $8s$ sequences from nuScenes. Before adversarial optimization, scenes are pre-filtered heuristically by how likely they are to produce a useful collision, leaving $<$500 scenarios

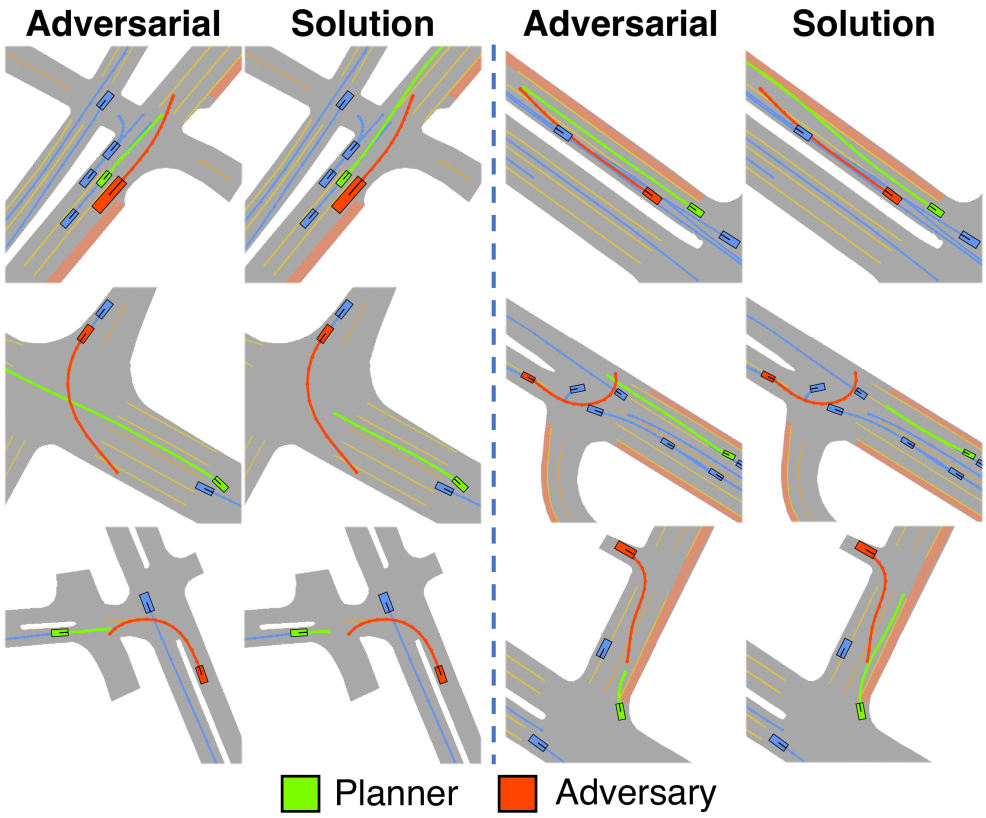

Figure 4.4: Qualitative results on the *Rule-based* planner. Adversarial and solution optimization results are shown. STRIVE produces diverse collision scenarios including lane changes, (u-)turning in front of the planner, and pulling into oncoming traffic.

to optimize.

Planner-Specific Scenarios. Tab. 4.1 shows that compared to rolling out a given planner on "regular" (unmodified) nuScenes scenarios, challenging scenarios from STRIVE produce more collisions and less comfortable driving. For the *Rule-based* planner, metrics on challenging scenarios are compared to the corresponding set of regular scenarios from which they originated (regular scenarios for *Replay* are omitted since nuScenes data contains no collisions and, by definition, planner behavior does not change).

Collision and solution rates indicate that generated scenarios are accident-prone and *useful* (solvable). For the *Rule-based* planner, adversarial optimization causes collisions in 27.4% of scenarios compared to only 1.2% in the regular scenarios. Generated scenarios also contain more severe collisions in terms of velocity, and elicit larger accelerations, *i.e.* less comfortable driving. The position and angle errors between approximate ($\hat{\mathbf{Y}}^{\text{plan}}$) and true ($\mathbf{Y}^{\text{plan}}$) planner trajectories at

				Planner Trajectory		Match Planner Err	
Planner	**Scenarios**	Collision (%)	Solution (%)	Accel (m/s^2)	Coll Vel (m/s)	Pos (m)	Ang (deg)
Replay	Challenging	43.7 (**+43.7**)	82.4	0.85	7.82	0.28	1.32
Rule-based	Regular	1.2	–	1.63	8.48	–	–
Rule-based	Challenging	27.4 (**+26.2**)	86.8	1.91 (**+0.28**)	9.65 (**+1.17**)	1.23	3.79

Table 4.1: Evaluation of generated *challenging* scenarios. Generated scenarios contain far more collisions compared to the corresponding *regular* (unmodified) scenes, as well as higher acceleration and collision speeds. Acceleration is measured in the forward direction (*i.e.* change in speed), since the *Rule-based* planner cannot change lanes. Rightmost columns show small errors between $\hat{\mathbf{Y}}^{\text{plan}}$ and $\mathbf{Y}^{\text{plan}}$.

	Plausibility of Adversary Trajectory ↓				Usefulness ↑
Scenarios	Accel (m/s^2)	Env Coll (%)	NN Dist (m)	NLL	Solution (%)
Bicycle	2.00	16.5	0.97	962.9	73.4
STRIVE	**0.98**	**10.8**	**0.72**	**323.4**	**83.5**

Table 4.2: Scenario generation for *Replay* planner compared to the *Bicycle* baseline, which does not leverage a learned traffic model.

the end of adversarial optimization are shown on the right (see Sec. 4.3.2). The largest position error of $1.23m$ is reasonable relative to the $4.084m$ length of the planner vehicle. Qualitative results for the *Rule-based* planner visualize 2D waypoint trajectories (Fig. 4.4); though not shown, STRIVE also generates speed and heading.

Baseline Comparison. STRIVE is next compared to a baseline approach to demonstrate that leveraging a learned traffic model is key to realistic and useful scenarios. Previous works are not directly comparable as they focus on small-scale scenario generation (*e.g.* [1, 33]) and/or attack the full AV stack rather than just the planner [276, 179]. Therefore, in the spirit of AdvSim [276] we implement the *Bicycle* baseline, which explicitly optimizes the kinematic bicycle model parameters (acceleration profile) of a single pre-chosen adversary in the scenario to cause a collision. Rather than using the learned traffic model, it relies on the bicycle model, collision penalties, and acceleration regularization to maintain plausibility. This precludes using the differentiable planner approximation from the traffic model, thus requiring gradient estimation (*e.g.* finite differences) for the closed-loop setting, which we found is $\approx 40\times$ slower and requires several hours to generate a single scenario. Therefore, comparison is done only on the *Replay* planner.

Tab. 4.2 shows that scenarios generated by *Bicycle* exhibit more unrealistic adversarial driving, and are more difficult to find a solution for. All metrics are reported only for scenarios where both

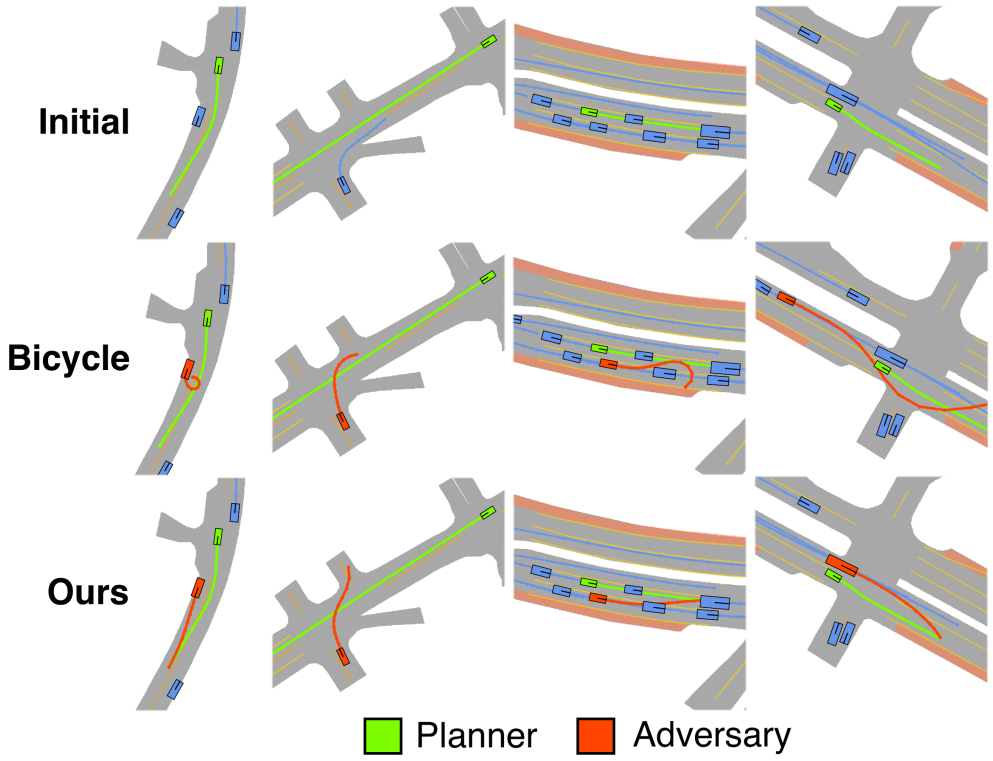

Figure 4.5: Qualitative comparison of generated scenarios for the *Replay* planner. *Bicycle* often produces semantically unrealistic trajectories as no learned traffic model is leveraged.

methods successfully caused a collision. In addition to higher accelerations, the *Bicycle* adversary collides with the non-drivable area more often (*Env Coll*), and exhibits less typical trajectories as measured by the distance to the nearest-neighbor ego trajectory in the nuScenes training split (*NN Dist*). After fitting the *Bicycle*-generated scenarios with our traffic model, we see the adversary's behavior is also less realistic as measured by the negative log-likelihood (NLL) of its latent $\mathbf{z}$ under the learned prior. These observations are supported qualitatively in Fig. 4.5.

4.5.2 Analyzing Generated Scenarios

Before improving a given planner, the analysis from Sec. 4.4.1 is used to identify useful scenarios by filtering out unsolvable scenarios and classifying collision types. For classification, collision features are clustered with $k = 10$ and clusters are visualized to manually assign the semantic labels shown in Fig. 4.6. The distribution of generated collision scenarios for both planners in Sec. 4.5.1 is shown in Fig. 4.6(a). STRIVE generates a diverse set of scenarios with solvable scenes found in all clusters.

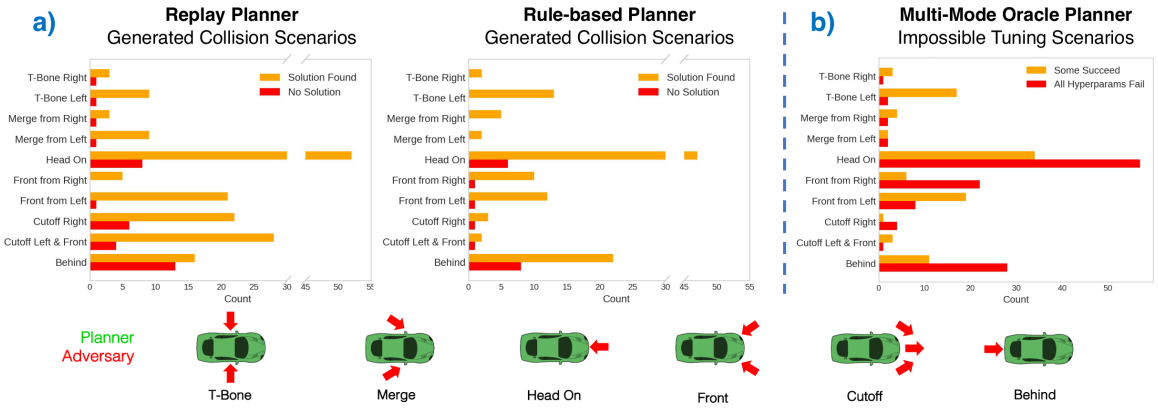

Figure 4.6: (Bottom) Collision types are depicted; the arrow indicates the position/direction of the adversary. **(a)** Distribution of generated scenarios for both planners. **(b)** Scenarios used to tune the multi-mode *oracle* planner. Scenarios where all parameter settings cause a collision are in red. A majority of *Head On*, *Front from Right*, and *Behind* scenarios always fail due to the inability to change lanes.

"Head On" is the most frequently generated scenario type, likely because *Replay* is non-reactive and *Rule-based* cannot change lanes. "Behind" exhibits the highest rate of unsolvable scenarios since being hit from behind is often the result of a negligent following vehicle, rather than undesirable planner behavior. *Replay* is much more susceptible to being cut off since it is open-loop, while the closed-loop *Rule-based* can successfully react to avoid such collisions.

4.5.3 Improving Rule-Based Planner

Now that we have a large set of labeled collision scenarios, in addition to the original nuScenes data containing "regular" scenarios (with few collisions), we can improve the *Rule-based* planner to be better prepared for challenging situations. Besides uncovering fundamental flaws that lead us to add new functionality, improvement is based on hyperparameter tuning via a grid search over possible settings. For each set of hyperparameters, the planner is rolled out within all scenarios of a dataset, and the optimal tuning is chosen based on the minimum collision rate. Tuning searches over p_{max} in the range of $[0.05, 0.2]$, max speeds in the range $[12.5, 20.0]$ m/s, max accelerations in the range $[3.0, 4.5]$ m/s^2, and parameters related to computing $p_{\text{col}}(\tau)$. In total, tuning sweeps over 432 hyperparameter combinations.

The planner is first tuned on regular scenarios before adversarial optimization is performed to create a set of challenging scenarios to guide further improvements. Performance of this initial regular-tuned planner on held out regular (*Reg*) and collision (*Coll*) scenarios is shown in the top

Improvement	**Collision** (%) Reg / Coll	**Coll Vel** (m/s) Reg / Coll	**Accel** (m/s^2) Reg / Coll
None (regular-tuned)	4.6 / 68.6	4.59 / 10.48	1.96 / 2.26
+ Challenging data	6.0 / 51.4	5.48 / 13.88	2.29 / 2.50
+ Extra *learned* mode	4.6 / 54.3	4.60 / 10.86	2.02 / 2.55
+ Extra *oracle* mode	4.6 / 54.3	4.59 / 10.40	1.96 / 2.39

Table 4.3: Improving *Rule-Based* planner. Including challenging tuning data and adding an extra "mode" improves performance on collision scenarios (*Coll*) while maintaining performance in regular scenarios (*Reg*). Acceleration is in the forward direction.

row of Tab. 4.3. Before any improvements, the planner collides in 68.6% of challenging and 4.6% of regular scenarios. Note that avoiding collisions altogether on regular scenarios is not possible: even if we choose optimal hyperparameters for each scenario separately, the collision rate is still 3.2%.

Tuning on Challenging Scenes. The first improvement, shown in the second row of Tab. 4.3, is to naïvely combine regular and challenging scenarios for tuning. Combined tuning greatly reduces the collision rate on challenging scenarios, but negatively impacts performance on regular driving. This points to a first *fundamental issue*: the planner uses a single set of hyperparameters for all driving situations, causing it to drive too aggressively in regular scenarios when tuned on challenging ones.

Multi-Mode Operation. To address this, we add a second set of parameters such that the planner has one for regular and one for accident-prone situations. Using this second "accident mode" of operation requires a binary classification of the current scene during rollout. For this, we augment the planner with a learned component that decides which parameter set to use based on a moving window of the past $2s$ of traffic; it is trained on scenarios generated by STRIVE. The extra parameter set is tuned on collision scenarios only. As shown in the third row of Tab. 4.3, this learned extra mode reduces the collision rate on challenging scenarios by 14.3% compared to the vanilla planner without hindering performance on regular scenes. We compare it to an *oracle* version (bottom row) that automatically switches into accident mode $2s$ before a collision is supposed to happen on generated scenarios, showing the learned version is achieving near-optimal performance.

Lane Change Limitation. The inability of the planner to switch lanes is another fundamental issue exposed by collision scenarios. Fig. 4.6(b) shows the distribution of tuning scenarios for the *oracle* multi-mode version; red bars indicate "impossible" scenarios where all sets of evaluated hyperparameters collide. A majority of "Head On" and "Behind" scenarios are impossible, pointing out the lane change limitation. Adversarial optimization has indeed exploited the flaw and the

proposed analysis made it visible.

4.5.4 Traffic Model Prediction Evaluation

Though our learned traffic model was designed primarily for controllability and plausibility to create scenarios, for completeness we evaluate its prediction abilities compared to state-of-the-art baselines and when key components are ablated. We evaluate on the test split of the nuScenes [26] prediction challenge using $2s$ (4 steps) of past motion to predict $6s$ (12 steps) of future. This data contains vehicles from the *bus*, *car*, *truck*, *construction*, and *emergency* categories.

Evaluation is done with standard future prediction metrics including minimum average displacement error (**ADE**) and minimum final displacement error (**FDE**), which are measured over K samples from the traffic model. For a single agent being evaluated, these metrics are

$$\text{ADE} = \min_k \frac{1}{T} \sum_{t=1}^{T} ||\hat{\mathbf{y}}_t^{(k)} - \mathbf{y}_t||_2 \tag{4.18}$$

$$\text{FDE} = \min_k ||\hat{\mathbf{y}}_T^{(k)} - \mathbf{y}_T||_2 \tag{4.19}$$

where $\hat{\mathbf{y}}_t^{(k)}$ is the predicted *position* of the agent in the kth sample at time t and $\mathbf{y}_t$ is the ground truth. In our experiments, we use $K = 10$ samples.

For the ablation study, we also measure the **environment and vehicle collision rates**. Environment collision rate is the fraction of predicted future trajectories where more than 5% of the vehicle bounding box overlaps with the non-drivable area. This is measured over all K samples. The vehicle collision rate is measured over all agents in each scene (rather than only the single one specified at each data point in the prediction challenge test split) and all K samples. It is the same as used in TrafficSim [248], which counts the number of agents in collision (*i.e.* have a bounding box overlap more than IoU 0.02 with another agent).

Baseline Comparison. Prediction performance is compared to reported results for recent state-of-the art models AgentFormer [298], Trajectron++ [228], DLow-AF [295], and LDS-AF [161]. Results are shown in Tab. 4.4. Our full model is trained only on car and $truck$ vehicles to be used for scenario generation, so to evaluate on the prediction challenge test split, we modify the category of input vehicles to our model to be one of these (*e.g. bus* $\rightarrow$ *truck*). Our learned traffic model makes accurate predictions and is competitive with current SOTA methods as shown in Tab. 4.4. We also train an ablation of our model that does not use the kinematic bicycle model, instead the decoder directly predicts output position and headings. This version is trained on all categories in the

Model	ADE $(m)\downarrow$	FDE $(m)\downarrow$
LDS-AF [161]	1.66	3.58
DLow-AF [295]	1.78	3.77
Trajectron++ [228]	1.51	-
AgentFormer [298]	**1.45**	**2.86**
Ours, Full	1.75	3.57
Ours, No Bicycle	1.60	3.17

Table 4.4: Learned traffic model future prediction accuracy on all nuScenes prediction categories compared to current state of art. ADE/FDE is reported using 10 samples.

Model	ADE (m)	FDE (m)	Env Coll $(\%)$	Veh Coll $(\%)$
Full	1.74	3.54	10.6	5.6
No Bicycle	**1.72**	**3.45**	**7.2**	**3.7**
No $\mathcal{L}_{\text{env}}$	1.91	3.92	13.2	5.0
No Autoregress	3.68	8.00	16.2	5.4

Table 4.5: Traffic model ablation study on *cars* and *truck* nuScenes categories only (same as used for scenario generation). Though not using the bicycle model gives better performance, it gives less realistic single-agent vehicle dynamics which is very undesirable for scenario generation.

challenge dataset, and makes more accurate predictions according to ADE/FDE. Note, however, that using the bicycle model is very important for adversarial and solution optimization to ensure output trajectories have reasonable dynamics even when the optimized Z is off-manifold.

Ablation Study. To evaluate key design differences from TrafficSim [248] and ILVM [29], which our model is based on, we ablate various components of our traffic model design. Results are shown in Tab. 4.5, where all models are trained and evaluated only on the car and $truck$ categories, since this is what is used for scenario generation. *No Bicycle* directly predicts the position and heading from the decoder rather than acceleration profiles that go through the kinematic bicycle model; again, this gives slightly improved performance but less realistic per-agent dynamics. *No* $\mathcal{L}_{env}$ only uses vehicle collision penalties while training, similar to prior work [248]. Removing the environment collision penalty results in a higher collision rate and lower predictive accuracy. *No Autoregress* uses a GNN decoder that predicts the entire future trajectory in one shot rather than as an autoregressive rollout. This makes the future prediction task more difficult, substantially reducing accuracy.

4.6 Discussion

STRIVE enables the automatic and scalable generation of plausible, accident-prone scenarios to

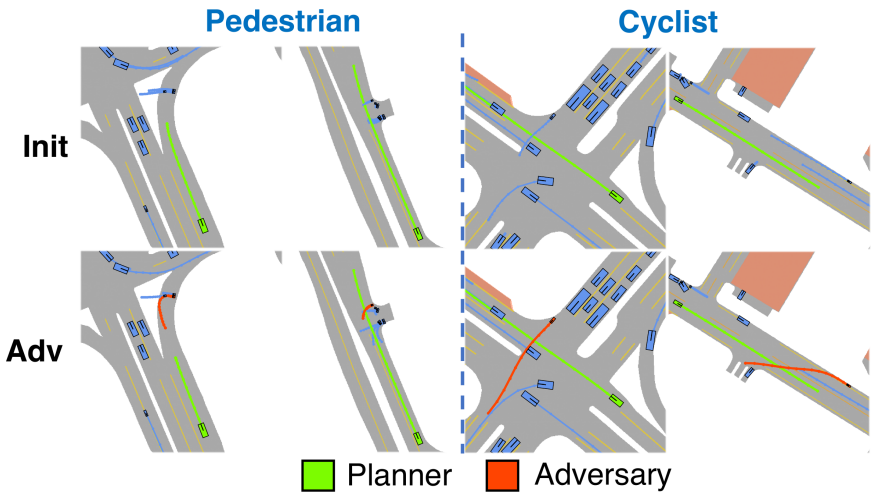

Figure 4.7: Generated scenarios for the *Replay* planner using a traffic model trained on *all* categories. Top row shows the initial scene and bottom is the output of adversarial optimization. When choosing the adversary ahead of time, STRIVE can cause collisions with both pedestrians (left) and cyclists (right).

improve a given planner. It does this based on a generative motion model of 2D vehicle trajectories that had to address several of the challenges outlined in Sec. 1.1.1. *Accuracy and diversity* was achieved by adapting a state-of-the-art CVAE model for trajectory synthesis. Moreover, this model was explicitly aware of *interactions* between agents by leveraging a graph structure and message passing layers. Finally, the latent space of this CVAE allowed for *controllability* through optimization to create both accident and solution scenarios.

Limitations and Future Work. Our method assumes perfect perception and only attacks the planner, but using our traffic model to additionally attack detection and tracking is of great interest. Other kinds of adversaries like adding/removing assets and changing map topology will also uncover additional AV weaknesses. Finally, STRIVE generates scenarios from existing data and only considers collisions between vehicles, but other incidents involving pedestrians and cyclists are also important. As a proof-of-concept for this, we train the learned traffic model on all categories in the nuScenes dataset and generate scenarios for the *Replay* planner where the adversary is a pedestrian or cyclist. As shown in Fig. 4.7, STRIVE gives promising results towards this direction.

Fig. 4.8 shows examples of other STRIVE limitations. First, our proposed solution optimization is iterative and operates on the full temporal planner trajectory, therefore it has access to future information that sometimes allows performing evasive maneuvers even before an "attack" is apparent. An example is in Fig. 4.8(a) where the optimized solution simply does not not pull into the roundabout

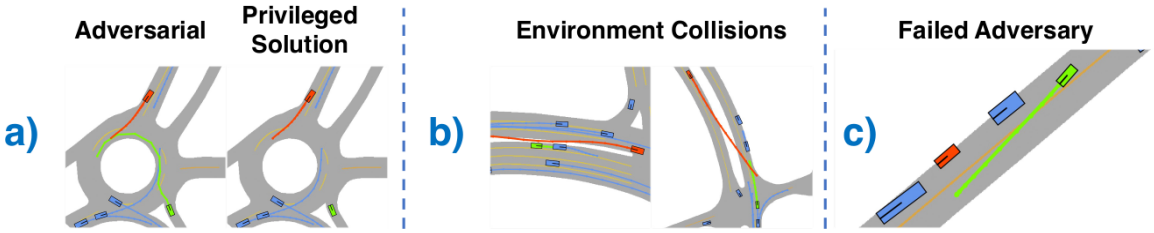

Figure 4.8: Example failure cases of STRIVE. (a) The solution optimization has access to privileged information, sometimes resulting in unrealistic "solutions". (b) Adversaries sometimes drive on non-drivable area to cause a collisions. (c) Attacks that require unlikely motion under the learned prior (*e.g.* a parked car pulling out) can be difficult to produce.

where the collision occurs. Fig. 4.8(b) shows that the adversary sometimes crosses non-drivable areas in order to collide with the planner. Though this scenario is technically possible, it is extreme behavior that may not be desired. However, these situations can be easily detected and discarded, and usually occur only when there is no other feasible adversary near the planner. Finally, adversarial optimization can have difficulty exhibiting behavior that is very unlikely under the prior even when it is realistic, *e.g.* a parked car pulling out as shown in Fig. 4.8(c), since these motions are rare in the traffic model training data.

Our method is intended to make AVs safer by exposing them to challenging and rare scenarios similar to the real world. However, our experiments expose the difficulty of properly balancing regular and challenging data when tuning a planner. Care must be taken to integrate generated scenarios into AV testing and to design unified planners that robustly address highly variable driving conditions.

Chapter 5

Controllable Trajectory Generation

In Chapter 4, we saw that generative CVAE motion models allow for detailed control of vehicle trajectories using optimization in the learned latent space. This controllability enabled creating specific traffic scenarios, such as causing accidents. In principle, though, the idea is applicable to any kind of desired controllability on vehicle or pedestrian trajectories. Since the optimization is done in a learned latent space using a data-driven prior as regularization, controlled trajectories are also encouraged to remain plausible. Though promising, this test-time optimization approach can be costly, requiring several minutes to modify trajectories for complex objectives. Moreover, the optimization process finds a single local minimum corresponding to a single output scenario; it is not able to generate a variety of different scenarios when starting from the same initialization.

In this chapter, we will introduce TRACE, which looks to address these shortcomings using recent advancements in diffusion modeling. By adapting diffusion models to conditionally generate pedestrian trajectories, we can guide trajectories as part of the usual denoising process to meet user objectives, while getting a diversity of samples. This work was originally published in CVPR 2023 [209]; although TRACE is connected to a character controller to enable full-body character animation, my primary contribution to this project was the guided trajectory diffusion model itself.

5.1 Introduction

Synthesizing high-level human behavior, in the form of 2D positional trajectories, is at the core of modeling pedestrians for applications like autonomous vehicles and architectural and environmental design. An important feature of such synthesis is *controllability* – generating trajectories that meet user-defined objectives, edits, or constraints. For example, a user may place specific waypoints

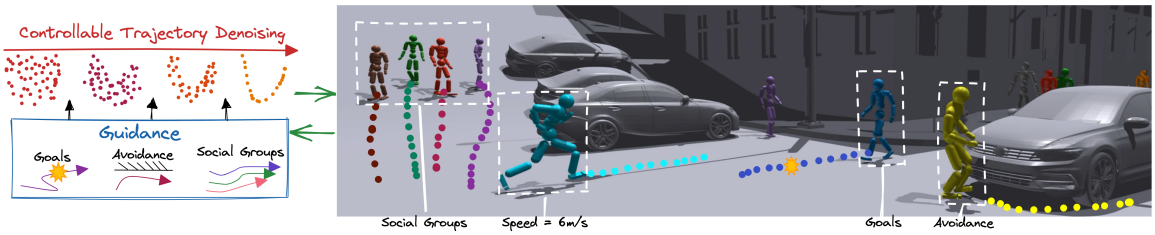

Figure 5.1: (Left) We propose TRACE, a trajectory diffusion model that enables user control through test-time guidance. (Right) Generated trajectories are passed to a novel physics-based humanoid controller (PACER), forming a closed-loop pedestrian animation system.

for characters to walk through, specify social groups for pedestrians to travel in, or define a social distance to maintain.

Attaining controllability is straightforward for algorithmic or rule-based models of human behavior. They have built-in objectives by construction. In the simplest case, human trajectories can be determined by the shortest paths between control points [92], but more sophisticated heuristics have also been developed for pedestrians [97, 13], crowds [212, 119], and traffic [253, 156]. Unfortunately, algorithmically generated trajectories often appear unnatural. Learning-based approaches, on the other hand, can improve naturalness by mimicking real-world data. These methods often focus on short-term trajectory prediction using a single forward pass of a neural network [86, 298, 228, 4]. However, the ability to *control* these models is limited to sampling from an output trajectory distribution [166, 284] or using an expensive latent space traversal [210] procedure. As a result, learning-based methods often predict motions that are implausible, containing collisions with obstacles or between pedestrians. This motivates another notion of *controllability* – ensuring physical plausibility of trajectories during agent-agent and agent-environment interactions.

In this work, we are particularly interested in using controllable pedestrian trajectory models for character animation. We envision a simple interface where a user provides high-level objectives, such as waypoints and social groups, and a system converts them to *physics-based* full-body human motion. Compared to existing kinematic motion models [201, 103, 147], physics-based methods have the potential to produce high-quality motion with realistic subtle behaviors during transitions, obstacle avoidance, traversing uneven terrains, *etc*.. Although there exist physics-based animation models [193, 95, 196, 194, 147, 281], controlling their behavior requires using task-specific planners that need to be re-trained for new tasks, terrains, and character body shapes.

We develop a generative model of trajectories that is data driven, controllable, and tightly integrated with a physics-based animation system for full-body pedestrian simulation (Fig. 5.1). Our

method enables us to generate pedestrian trajectories that are realistic and amenable to meeting user-defined objectives at test time. We integrate this trajectory generator as a high-level planner for a physics-based pedestrian controller, resulting in a closed-loop controllable pedestrian animation system.

For trajectory generation, we introduce a **TRA**jectory Diffusion Model for **C**ontrollable P**E**destrians (TRACE). Inspired by recent successes in trajectory generation through denoising [81, 110, 314], TRACE generates the future trajectory for each pedestrian in a scene and accounts for the surrounding context through a spatial grid of learned map features that is queried locally during denoising. We leverage classifier-free sampling [101] to allow training on mixed annotations (*e.g.*, with and without a semantic map), which improves controllability at test time by trading off sample diversity and compliance to conditioning. User-controlled sampling from TRACE is achieved through test-time *guidance* [54, 101, 102], which perturbs the output at each step of denoising toward the desired objective. We extend prior work [110] by introducing several analytical loss functions for pedestrians and re-formulating trajectory guidance to operate on clean trajectory outputs from the model [102], improving sample quality and adherence to user objectives.

For character animation, we develop a general-purpose **P**edestrian **A**nimation **C**ontroll**ER** (PACER) capable of driving physics-simulated humanoids with diverse body types to follow trajectories from a high-level planner. We focus on (1) motion quality: PACER learns from a small motion database to create natural and realistic locomotion through adversarial motion learning [196, 194]; (2) terrain and social awareness: by learning in diverse terrains with other humanoids, PACER learns to move through stairs, slopes, uneven surfaces, and to avoid obstacles and other pedestrians; (3) diverse body shapes: by training on different body types, PACER draws on years of simulation experience to control a wide range of characters; (4) compatibility with high-level planners: PACER accepts 2D waypoints and can be a plug-in model for any 2D trajectory planner.

We demonstrate a controllable pedestrian animation system using TRACE as a high-level planner for PACER, the low-level animator. The planner and controller operate in a closed loop through frequent re-planning according to simulation results. We deepen their connection by guiding TRACE with the value function learned during RL training of PACER to improve animation quality in varying tasks. We evaluate TRACE on synthetic [13] and real-world pedestrian data [192, 140, 26], demonstrating its test-time flexibility to both user-specified and plausibility objectives while synthesizing realistic motion. Furthermore, we show our animation system is capable and robust with a variety of tasks, terrains, and characters. In summary, we contribute:

- A diffusion model for pedestrian trajectories that is readily controlled at test time through

guidance;

- A general-purpose pedestrian animation controller for diverse body types and terrains;
- A pedestrian animation system that tightly integrates the two to drive simulated characters in a controllable way.

5.2 Related Work

Pedestrian Trajectory Prediction. Modeling high-level pedestrian behavior has been extensively studied in the context of motion prediction (forecasting). Approaches range from physics and planning-based [97, 261, 96] to recent learned methods [298, 228, 4, 31, 137]. We refer the reader to the thorough survey by Rudenko *et al.* [221] for an overview and focus this discussion on controllability. Most forecasting works are motivated by planning for autonomous vehicles (AVs) or social robots [86] rather than *controllability* or longer-term synthesis. Rule-based models for pedestrians [13, 212, 119] and vehicle traffic [253, 156] can easily incorporate user constraints [139] making them amenable to control. However, trajectories from these approaches are not always human-like; methods have even been developed to choose the best simulation method and tune parameters to make crowd scenarios more realistic [117].

Data-driven methods produce human-like motions, but neural network-based approaches are difficult to explicitly *control*. Some works decompose forecasting into goal prediction followed by trajectory prediction based on goals [166, 50]. These models offer limited control by selecting goal locations near a target or that minimizes an objective (*e.g.* collisions) [284]. Synthesized pedestrian behavior can also be controlled by strategically choosing a starting location [202]. STRIVE [210] showed that a VAE trajectory model can be controlled through test-time optimization in the learned latent space. Reinforcement learning (RL) agents can be controlled in crowd simulations by incorporating tasks into reward functions for training [136]. By varying the weights of different rewards, characters can be controlled to exhibit one of several behaviors at test time [183]. Our method, TRACE, trains to mimic trajectories from data and is agnostic to any task: all controls are defined at test time, allowing flexibility to new controls after training. Instead of lengthy test-time optimization, we use guidance for control.

Controllable Character Animation. Full-body pedestrian animation typically involves a high-level task (*e.g.* trajectory following, obstacle avoidance) and low-level body control. Some methods solve both with a single network that implicitly does high-level planning and low-level animation.

GAMMA [312] trains a kinematic model to go to waypoints, while PFNN [103] follows gamepad inputs. Physics-based humanoid controllers such as AMP [196] train different models for each task, limiting their general applicability.

Two-stage methods split the task into separate high-level planning and low-level character control, where task information is only used by the planner. Planning can be done with traditional A* [92], using learned trajectory prediction [28], searching in a pre-trained latent space [147, 281, 194], or using hierarchical RL [95, 193, 194, 201, 281]. DeepLoco [193], Haworth *et al.* [95], and ASE [194] utilize hierarchical RL to achieve impressive dynamic control for various tasks. They require lengthy training for both low-level and high-level controllers and often jointly train as a final step. They must also train different planners for different tasks.

Our approach follows the two-stage paradigm, with the distinction that both our high-level (TRACE) and low-level (PACER) models consume task information for pedestrian navigation: through test-time guidance and map-conditioned path following, respectively. TRACE and PACER are unaware of each other at training time, yet can be tightly integrated in a closed loop: trace-pace-retrace.

Diffusion Models and Guidance. Diffusion models have shown success in generating images [100, 176, 260], videos [99], and point clouds [303]. Guidance has been used for test-time control in several ways: classifier [54] and classifier-free [101] guidance re-inforce input conditioning, while reconstruction guidance [102] has been used for coherent video generation. Gu *et al.* [81] adapt the diffusion framework for short-term pedestrian trajectory forecasting conditioned on past trajectories. Diffuser [110] generates trajectories for planning and control in robotics applications with test-time guidance. Closest to ours is the concurrent work of CTG [314], which builds on Diffuser to develop a controllable vehicle traffic model, focusing on following formalized traffic rules like speed limits. Our method TRACE contains several key differences: we encode map conditioning into an expressive feature grid queried in denoising, we use classifier-free sampling to enable multi-dataset training and test-time flexibility, we re-formulate guidance to operate on clean model outputs, and we link with a low-level animation model using value function guidance.

5.3 Method

To model high-level pedestrian behavior, we first introduce the controllable trajectory diffusion model (TRACE). In Sec. 5.3.2, we detail our low-level physics-based pedestrian controller, PACER, and in Sec. 5.3.3 how they can be combined into an end-to-end animation system.

5.3.1 Controllable Trajectory Diffusion

Problem Setting. Our goal is to learn high-level pedestrian behavior in a way that can be *controlled* at test time. For pedestrian animation, we focus on two types of control: (1) user specification, *e.g.*, goal waypoints, social distance, and social groups, and (2) physical plausibility, *e.g.*, avoiding collisions with obstacles or between pedestrians.

We formulate synthesizing pedestrian behavior as an agent-centric trajectory forecasting problem. At each time step, the model outputs a future trajectory plan for a target *ego* agent conditioned on that agent's past, the past trajectories of all neighboring agents, and the semantic map context. Formally, at timestep t we want the future state trajectory $\boldsymbol{\tau}_s = [\mathbf{s}_{t+1} \quad \mathbf{s}_{t+2} \quad \dots \quad \mathbf{s}_{t+T_f}]$ over the next T_f steps where the state $\mathbf{s} = [x \quad y \quad \theta \quad v]^T$ includes 2D position (x, y), heading angle θ, and speed v. We assume this state trajectory is actually the result of a sequence of actions [314] defined as $\boldsymbol{\tau}_a = [\mathbf{a}_{t+1} \quad \mathbf{a}_{t+2} \quad \dots \quad \mathbf{a}_{t+T_f}]$ where each action $\mathbf{a} = [\dot{v} \quad \dot{\theta}]^T$ contains the acceleration $\dot{v}$ and yaw rate $\dot{\theta}$. The state trajectory can be recovered from the initial state and action trajectory using a given dynamics model $\boldsymbol{\tau}_s = f(\mathbf{s}_t, \boldsymbol{\tau}_a)$. The full state-action trajectory is then denoted as $\boldsymbol{\tau} = [\boldsymbol{\tau}_s; \boldsymbol{\tau}_a]$. To predict the future trajectory, the model receives as input the past state trajectory of the ego pedestrian $\mathbf{x}^{\text{ego}} = [\mathbf{s}_{t-T_p} \quad \dots \quad \mathbf{s}_t]$ along with the past trajectories of N neighboring pedestrians $X^{\text{neigh}} = \{\mathbf{x}^{\text{i}}\}_{i=1}^{N}$. It also gets a crop of the rasterized semantic map $\mathcal{M} \in \mathbb{R}^{H \times W \times C}$ in the local frame of the ego pedestrian at time t. These inputs are summarized as the *conditioning* context $C = \{\mathbf{x}^{\text{ego}}, X^{\text{neigh}}, \mathcal{M}\}$.

Our key idea is to train a diffusion model to do this conditional trajectory generation, which can be *guided* at test time to enable controllability. For simplicity, the following formulation uses the full trajectory notation $\boldsymbol{\tau}$, but in practice, the state trajectory is always a result of actions, *i.e.*, diffusion/denoising are on $\boldsymbol{\tau}_a$ which determines the state.

Trajectory Diffusion Model

We build on Diffuser [110] and generate trajectories through iterative denoising, which is learned as the reverse of a pre-defined *diffusion* process [100, 238]. Starting from a clean future trajectory $\boldsymbol{\tau}^0 \sim q(\boldsymbol{\tau}^0)$ sampled from the data distribution, the forward noising process produces a sequence of progressively noisier trajectories $(\boldsymbol{\tau}^1, \dots, \boldsymbol{\tau}^k, \dots, \boldsymbol{\tau}^K)$ by adding Gaussian noise at each process step k:

$$
\begin{aligned}
q(\boldsymbol{\tau}^{1:K} \mid \boldsymbol{\tau}^0) &:= \prod_{k=1}^{K} q(\boldsymbol{\tau}^k \mid \boldsymbol{\tau}^{k-1}) \\
q(\boldsymbol{\tau}^k \mid \boldsymbol{\tau}^{k-1}) &:= \mathcal{N}(\boldsymbol{\tau}^k; \sqrt{1-\beta_k}\boldsymbol{\tau}^{k-1}, \beta_k \mathbf{I})
\end{aligned}
\tag{5.1}
$$

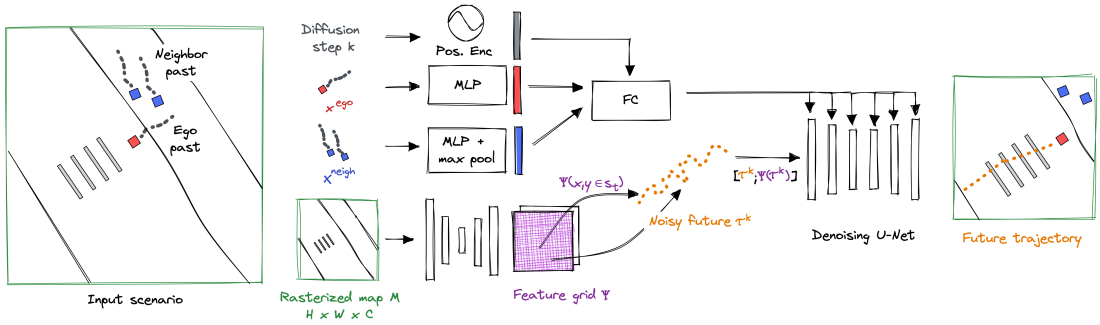

Figure 5.2: Trajectory diffusion model (TRACE). Future trajectory denoising is conditioned on past and neighbor motion by adding processed features to intermediate U-Net features. Map conditioning is provided through a feature grid queried along the noisy input trajectory.

where β_k is the variance at each step from a fixed schedule, and with large enough K we get $q(\boldsymbol{\tau}^K) \approx \mathcal{N}(\boldsymbol{\tau}^K; \mathbf{0}, \mathbf{I})$. TRACE learns the reverse of this process so that sampled noise can be *denoised* into plausible trajectories. Each step of this reverse process is conditioned on C:

$$p_\phi(\boldsymbol{\tau}^{k-1} \mid \boldsymbol{\tau}^k, C) := \mathcal{N}(\boldsymbol{\tau}^{k-1}; \boldsymbol{\mu}_\phi(\boldsymbol{\tau}^k, k, C), \boldsymbol{\Sigma}_k) \tag{5.2}$$

where ϕ are model parameters and $\boldsymbol{\Sigma}_k$ is from a fixed schedule. TRACE learns to parameterize the mean of the Gaussian distribution at each step of the denoising process.

Training and Classifier-Free Sampling. Importantly for guidance, the network does *not* directly output $\boldsymbol{\mu}$. Instead, at every step it learns to predict the final clean trajectory $\boldsymbol{\tau}^0$, which is then used to compute $\boldsymbol{\mu}$ [176]. In particular, we can compute $\boldsymbol{\mu}$ from $\boldsymbol{\tau}^k$ and $\boldsymbol{\tau}^0$ using

$$\boldsymbol{\mu}(\boldsymbol{\tau}^0, \boldsymbol{\tau}^k) := \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\boldsymbol{\tau}^0 + \frac{\sqrt{\alpha_k}\,(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\boldsymbol{\tau}^k \tag{5.3}$$

where β_k is the variance from the schedule, $\alpha_k := 1 - \beta_k$, and $\bar{\alpha}_k := \prod_{j=0}^{k} \alpha_j$.

Training supervises the network output $\hat{\boldsymbol{\tau}}^0$ with ground truth future trajectories (*i.e.* denoising score matching [270, 240, 100]):

$$L = \mathbb{E}_{\boldsymbol{\epsilon}, k, \boldsymbol{\tau}^0, C}\left[||\boldsymbol{\tau}^0 - \hat{\boldsymbol{\tau}}^0||^2\right] \tag{5.4}$$

where $\boldsymbol{\tau}^0$ and C are sampled from the training dataset, $k \sim \mathcal{U}\{1, 2, \ldots, K\}$ is the step index, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is used to corrupt $\boldsymbol{\tau}^0$ to give the noisy input trajectory $\boldsymbol{\tau}^k$.

Our training procedure allows the use of *classifier-free sampling*[1] at test time, which has been shown to improve compliance to conditioning in diffusion models [101]. We simultaneously train both a conditional model $\boldsymbol{\mu}_\phi(\boldsymbol{\tau}^k, k, C)$ and unconditional model $\boldsymbol{\mu}_\phi(\boldsymbol{\tau}^k, k)$ by randomly dropping out conditioning during training. At test time, predictions from both models are combined with weight w as:

$$\tilde{\boldsymbol{\epsilon}}_\phi = \boldsymbol{\epsilon}_\phi(\boldsymbol{\tau}^k, k, C) + w\left(\boldsymbol{\epsilon}_\phi(\boldsymbol{\tau}^k, k, C) - \boldsymbol{\epsilon}_\phi(\boldsymbol{\tau}^k, k)\right) \tag{5.5}$$

where $\boldsymbol{\epsilon}_\phi$ is the model's prediction of how much noise was added to the clean trajectory to produce the input $\boldsymbol{\tau}^k$. This is straightforward to compute using [176]:

$$\boldsymbol{\epsilon} = \frac{\boldsymbol{\tau}^k - \sqrt{\bar{\alpha}_k}\boldsymbol{\tau}^0}{\sqrt{1-\bar{\alpha}_k}}. \tag{5.6}$$

Note that $w{>}0$ and $w{<}0$ increase and decrease the effect of conditioning, respectively, while $w{=}0$ and $w{=}{-}1$ result in the purely conditional or unconditional model, respectively. This flexibility allows a user to trade off respecting conditioning with trajectory diversity, which benefits controllability (see Sec. 5.4.2). This approach also enables training on multiple distinct datasets with varying annotations: conditioning is already being dropped out randomly, so it is easy to use mixed data with subsets of the full conditioning. Since there are pedestrian datasets with diverse motions but no semantic maps [192, 140] and others with limited motions but detailed maps [26], we find mixed training is beneficial to boost diversity and controllability (see Sec. 5.4.2).

Architecture. As shown in Fig. 5.2, TRACE uses a U-Net similar to [110] which has proven effective for trajectories. The input trajectory $\boldsymbol{\tau}_k$ at step k is processed by a sequence of 1D temporal convolutional blocks that progressively down and upsample the sequence in time, leveraging skip connections. A key challenge is how to condition the U-Net on C to predict trajectories compliant with the map and other pedestrians. To incorporate the step k, ego past $\mathbf{x}^{\text{ego}}$, and neighbor past X^{neigh}, we use a common approach [110, 102] that extracts a single conditioning feature and adds it to the intermediate trajectory features within each convolutional block. We propose encoding $\mathcal{M}$ with a 2D convolutional network into a feature grid where each pixel contains a high-dimensional feature. At step k of denoising, the 2D position at each timestep t of the current noisy input trajectory $\boldsymbol{\tau}^k$ is queried in the map feature grid to obtain a feature $\mathbf{g}_t = \boldsymbol{\Psi}(x_t, y_t) \in \mathbb{R}^{32}$. This query is done through bilinear interpolation of map features at the corresponding point. Over all timesteps, these form a feature trajectory $\mathbf{G} = [\mathbf{g}_{t+1} \dots \mathbf{g}_{t+T_f}]$ that is concatenated along the channel dimension

[1] we refer to it as "sampling" instead of the common term "guidance" to avoid confusion with the guidance introduced in Sec. 5.3.1

with $\boldsymbol{\tau}^k \in \mathbb{R}^{T_f \times 6}$ (containing both actions and states) to get the full trajectory input to the denoising U-Net $[\boldsymbol{\tau}^k; \mathbf{G}] \in \mathbb{R}^{T_f \times 38}$. Intuitively, this allows learning a localized representation that can benefit subtle map interactions such as obstacle avoidance.

Controllability through Clean Guidance

After training TRACE to generate realistic trajectories, controllability is implemented through test-time *guidance*. Intuitively, guidance nudges the sampled trajectory at each step of denoising towards a desired outcome. Let $\mathcal{J}(\boldsymbol{\tau})$ be a guidance loss function measuring how much a trajectory $\boldsymbol{\tau}$ violates a user objective. This may be learned [110] or an analytical differentiable function [314]. Guidance uses the gradient of $\mathcal{J}$ to perturb the predicted mean from the model at each denoising step such that the right side of Eq. (5.2) becomes $\mathcal{N}(\boldsymbol{\tau}^{k-1}; \tilde{\boldsymbol{\mu}}_\phi(\boldsymbol{\tau}^k, k, C), \boldsymbol{\Sigma}_k)$ where $\tilde{\boldsymbol{\mu}}$ is the perturbed (guided) mean. Prior work [110, 314] directly perturbs the *noisy* network-predicted mean with

$$\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu} - \alpha \boldsymbol{\Sigma}_k \nabla_{\boldsymbol{\mu}} \mathcal{J}(\boldsymbol{\mu}) \tag{5.7}$$

where α determines the guidance strength. Note that Eq. (5.7) evaluates $\mathcal{J}$ at the noisy mean, so learned loss functions must be trained at varying noise levels and analytic loss functions may hit numerical issues.

To avoid this, we build upon "reconstruction guidance", which operates on the *clean* model prediction $\hat{\boldsymbol{\tau}}^0$ [102]. We extend the guidance formulation introduced in [102] for temporal video upsampling to work with arbitrary loss functions. At each denoising step with input $\boldsymbol{\tau}^k$, we first perturb the predicted clean trajectory from the network $\hat{\boldsymbol{\tau}}^0$ with

$$\tilde{\boldsymbol{\tau}}^0 = \hat{\boldsymbol{\tau}}^0 - \alpha \boldsymbol{\Sigma}_k \nabla_{\boldsymbol{\tau}^k} \mathcal{J}(\hat{\boldsymbol{\tau}}^0), \tag{5.8}$$

then compute $\tilde{\boldsymbol{\mu}}$ in the same way as we would in Eq. (5.2), *i.e.* as if $\tilde{\boldsymbol{\tau}}^0$ were the output of the network. Note that the gradient is evaluated wrt the noisy input trajectory $\boldsymbol{\tau}^k$ rather than the clean $\hat{\boldsymbol{\tau}}^0$, requiring backpropagation through the denoising model. We formulate several analytical guidance objectives like waypoint reaching, obstacle avoidance, collision avoidance, and social groups (see Sec. 5.4.1, 5.4.2), and show a learned RL value function can also be used in Sec. 5.4.3.

Scene-Level Guidance. Some guidance objectives are based on multi-agent interactions, *e.g.*, agent collision avoidance and social groups. In this case, we assume that all pedestrians in a scene can be denoised simultaneously in a batched fashion. At each denoising step, the loss function is evaluated at the current trajectory prediction of all pedestrians and gradients propagated back to each one for

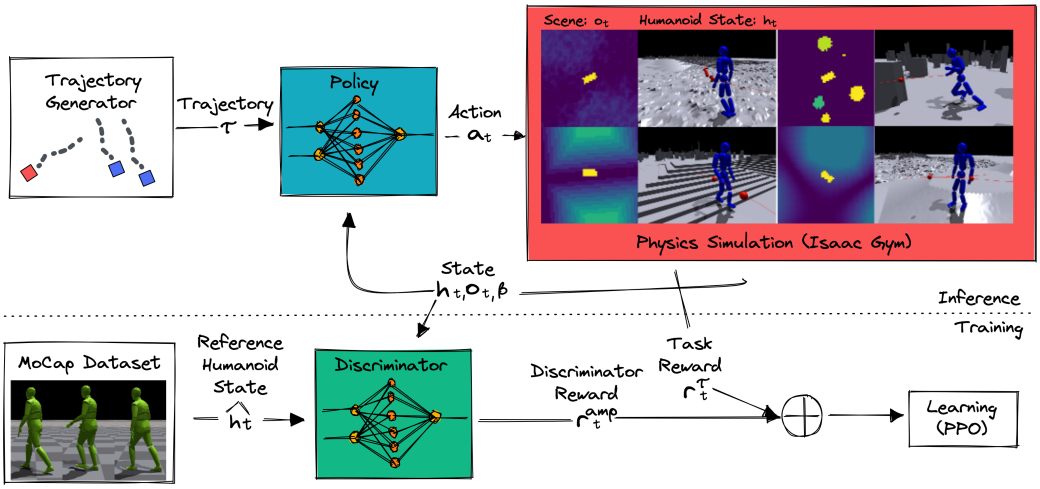

Figure 5.3: Pipeline: Pedestrian Animation Controller (PACER).

guidance. This can be seen as sampling a scene-level future rather than a single agent future.

5.3.2 Physics-Based Pedestrian Animation

To enable full-body pedestrian simulation, we design the Pedestrian Animation ControllER (PACER) to execute the 2D trajectories generated by TRACE in a physics simulator. In our experiments, we use a similar procedure as [297, 160], to automatically generate humanoid models that conform to the kinematic structure of SMPL [155].

Background: Goal-Conditioned RL. Our framework (Fig. 5.3) follows the general goal-conditioned reinforcement learning framework, where a goal-conditioned policy π_{PACER} is trained to follow 2D target trajectories specified by $\boldsymbol{\tau}_s$. The task is formulated as a Markov Decision Process (MDP) defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma \rangle$ of states, actions, transition dynamics, reward function, and discount factor. The state $\mathcal{S}$, transition dynamics $\mathcal{T}$, and reward R are calculated by the environment based on the current simulation and goal $\boldsymbol{\tau}_s$, while the action $\mathcal{A}$ is computed by the policy π_{PACER}. The policy's objective is to maximize the discounted return $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$ where r_t is the per-timestep reward. We utilize Proximal Policy Optimization (PPO) [229] to find the optimal control policy π_{PACER}.

Terrain, Social, and Body Awareness. To create a controller that can simulate crowds in realistic 3D scenes (*e.g.* scans, neural reconstructions, or artist-created meshes (Fig. 5.1)), our humanoid must be terrain aware, socially aware of other agents, and support diverse body types. Our control policy is conditioned on the state of the simulated character $\boldsymbol{h}_t$, environmental features $\boldsymbol{o}_t$, body

type $\boldsymbol{\beta}$, and the goal states $\boldsymbol{\tau}_s$: $\pi_{\text{PACER}}(\boldsymbol{a}_t|\boldsymbol{h}_t, \boldsymbol{o}_t, \boldsymbol{\beta}, \boldsymbol{\tau}_s)$. The policy inputs a rasterized local height and velocity map of size $\boldsymbol{o}_t \in R^{64\times64\times3}$ to give agents crucial information about their surroundings. To allow for social awareness, nearby humanoids are represented as a cuboid and rendered on the global height map at runtime. In this way, each humanoid views other people as dynamic obstacles to avoid. Obstacle and interpersonal avoidance are learned by using obstacle collision as a termination condition. By conditioning and training with different body parameters $\boldsymbol{\beta}$ our policy learns to adapt to characters with diverse morphologies.

Realistic Motion through Adversarial Learning. To learn the optimal control policy π_{PACER} that (1) follows a 2D trajectory closely and (2) creates realistic pedestrian motions, we follow Adversarial Motion Prior (AMP) [196], which uses a motion discriminator to encourage the policy to generate motions that are similar to the movement patterns contained in a dataset of motion clips recorded from human actors. The discriminator $D(\boldsymbol{h}_t, \boldsymbol{a}_t)$ is then used to specify a motion style reward r_t^{amp} for training the policy. The style reward is combined with a trajectory following reward r_t^{τ} and an energy penalty r_t^{energy} [70] to produce the total reward $r_t = r_t^{\text{amp}} + r_t^{\tau} + r_t^{\text{energy}}$. To mitigate artifacts arising from asymmetric gaits, such as limping, we utilize the motion-symmetry loss proposed by [294]:

$$L_{\text{sym}}(\theta) = \|\pi_{\text{PACER}}(\boldsymbol{h}_t, \boldsymbol{o}_t, \boldsymbol{\beta}, \boldsymbol{\tau}_s) - \Phi_a(\pi_{\text{PACER}}(\Phi_s(\boldsymbol{h}_t, \boldsymbol{o}_t, \boldsymbol{\beta}, \boldsymbol{\tau}_s)))\|^2, \tag{5.9}$$

where Φ_s and Φ_a mirrors the state and action along the character's sagittal plane. This loss encourages the policy to produce more symmetric motions, leading to more naturalistic gaits. During training, random terrains are generated following the procedure used in [222]. We create stairs, slopes, uneven terrains, and obstacles consisting of random polygons. The morphology of the character is also randomized by sampling a gender and body type from the AMASS dataset [162]. The policy and discriminator are then conditioned on the SMPL gender and body shape $\boldsymbol{\beta}$ parameters.

5.3.3 Controllable Pedestrian Animation System

The high-level trajectory planning from TRACE is combined with the low-level character control from PACER to create an end-to-end pedestrian animation system. The two components are trained independently, but at run-time, they operate in a closed feedback loop: PACER follows planned trajectories for $2s$ before TRACE re-planning, taking past character motion from PACER as input. By combining terrain and social awareness of PACER with collision avoidance guidance, both high and low level systems are task-aware and work in tandem to prevent collisions and falls.

Value Function as Guidance. To enable tighter two-way coupling between TRACE and PACER, in Sec. 5.4.3 we explore using the value function learned during RL training of PACER to guide trajectory diffusion. The value function predicts expected future rewards and is aware of body pose and surrounding terrain and agents. Using the value function to guide denoising encourages TRACE to produce trajectories that are easier to follow and better suited to the current terrain (which TRACE is unaware of otherwise). Unlike Diffuser [110], which requires training a reward function with samples from the diffusion model at varying noise levels, our guidance (Eq. (5.8)) operates on clean trajectories so we can use the value function directly from RL training.

5.4 Experiments

We first demonstrate the controllability of TRACE when trained on synthetic (Sec. 5.4.1) and real-world (Sec. 5.4.2) pedestrian data. Sec. 5.4.3 evaluates our full animation system on several tasks and terrains.

Implementation Details. TRACE is trained to predict T_f=$5s$ of future motion at 10 Hz from T_p=$3s$ of past motion, and uses K=100 diffusion steps. During training, map and neighbor conditioning inputs are independently dropped with 10% probability. At test time, we sample (and guide) multiple future trajectories for each pedestrian in a scene and choose one with the lowest guidance loss, which we refer to as *filtering*. For PACER, we randomly sample terrain, body type, and procedural 2D trajectories during training and used a motion dataset consisting of locomotion sequences from AMASS [162]. All physics simulations are performed using NVIDIA's Isaac Gym simulator [165].

Datasets. The *ORCA* dataset (Sec. 5.4.1) contains synthetic data of $10s$ scenes generated using the ORCA crowd simulator [13]. Up to 20 agents are placed in a $15m$×$15m$ environment with up to 20 static primitive obstacles. Agent placement and goal velocity along with obstacle location, scale, and orientation are randomized per scene. The dataset contains two distinct subsets: *ORCA-Maps* has many obstacles but few agents, while *ORCA-Interact* has no obstacles (*i.e.* no map annotations) but many agents.

For real-world data (Sec. 5.4.2), we use ETH/UCY and nuScenes. ETH/UCY [192, 140] is a common trajectory forecasting benchmark containing scenes with dense crowds and interesting pedestrian dynamics but does not have semantic maps. nuScenes [26] contains $20s$ driving scenes in common street settings. We convert the pedestrian bounding box annotations to 2D trajectories and use them for training and evaluation. Detailed semantic maps are also annotated with layers for roads, crosswalks, and sidewalks.

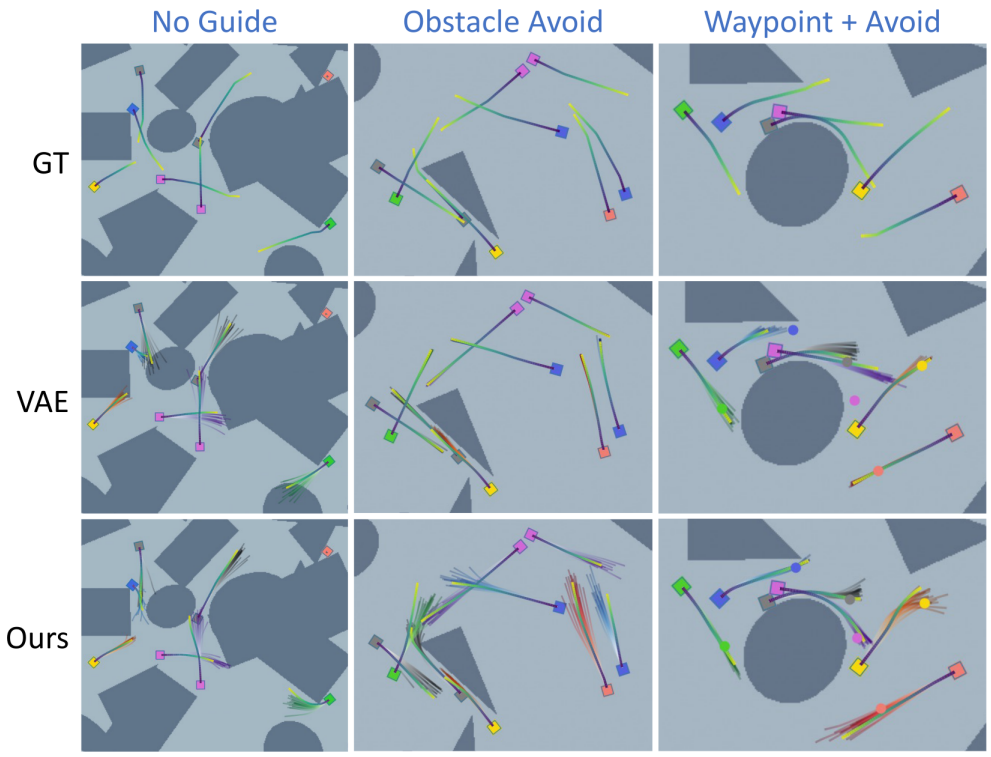

Figure 5.4: Guidance results on *ORCA-Maps*. For VAE and TRACE, 20 samples are visualized for each pedestrian (the boxes) along with the final trajectory chosen via filtering which is bolded.

Metrics. We care about trajectory plausibility and meeting user controls. Controllability is evaluated with a *Guidance Error* that depends on the task: *e.g.*, for avoidance objectives this is collision rate, while the waypoint error measures the minimum distance from the trajectory. *Obstacle and Agent Collision Rates* measure the frequency of collisions. *Realism* is measured at the dataset or trajectory level by (1) computing the Earth Mover's Distance (EMD) between the generated and ground truth test-set histograms of trajectory statistics (*e.g.* velocity, longitudinal/lateral acceleration) [284], or (2) measuring the mean accelerations of each trajectory assuming pedestrians generally move smoothly.

5.4.1 Augmenting Crowd Simulation

We first evaluate TRACE trained on *ORCA-Maps* and *ORCA-Interact*. These provide a clean test bed for comparisons since there is a clear definition of correct pedestrian behavior – no obstacle or agent collisions are present in the data. All methods operate in an open loop by predicting a single $5s$ future for each pedestrian. This way, compounding errors inherent to closed-loop operation are

Guide	Method	Guidance Error	Collision Rate Obstacle	Collision Rate Agent	Realism (EMD) Vel	Realism (EMD) Lon Acc	Realism (EMD) Lat Acc
None	VAE [210]	–	0.076	**0.118**	0.038	0.039	0.040
	TRACE	–	**0.050**	0.132	**0.029**	**0.008**	**0.009**
Obstacle Avoid	VAE [210]	0.018	0.018	**0.116**	0.040	0.036	0.039
	TRACE-Filter	0.018	0.018	0.123	**0.019**	**0.011**	**0.015**
	TRACE-Noisy	0.015	0.015	0.125	0.021	0.012	0.017
	TRACE	**0.014**	**0.014**	0.124	0.020	**0.011**	0.017
Agent Avoid	VAE [210]	0.010	0.075	0.010	0.041	0.038	0.039
	TRACE-Filter	0.049	**0.050**	0.049	0.031	0.012	0.013
	TRACE-Noisy	**0.000**	0.056	**0.000**	0.035	0.013	**0.012**
	TRACE	**0.000**	0.058	**0.000**	**0.025**	**0.010**	**0.012**
Waypoint	VAE [210]	**0.078**	0.051	**0.092**	0.070	0.031	0.033
	TRACE-Filter	0.333	**0.046**	0.112	**0.044**	**0.013**	**0.013**
	TRACE-Noisy	0.129	0.052	0.110	0.067	0.038	0.033
	TRACE	0.105	0.048	0.093	0.057	**0.013**	0.014
Waypoint & Obs Avoid & Agt Avoid	VAE [210]	**0.207**	**0.021**	0.015	0.053	0.032	0.032
	TRACE-Filter	0.527	0.023	0.096	**0.025**	0.014	0.016
	TRACE-Noisy	0.236	0.022	0.017	0.057	0.025	0.022
	TRACE	0.211	**0.021**	**0.009**	0.036	**0.007**	**0.009**

Table 5.1: Guidance evaluation on *ORCA-Maps* dataset. TRACE using full diffusion guidance improves upon VAE latent optimization and selective sampling (*TRACE-Filter*) in terms of meeting objectives, while maintaining strong realism.

not a factor.

Results for single and multi-objective guidance on the *ORCA-Maps* test set are shown in Tab. 5.1. TRACE is compared to a *VAE* baseline [210] adapted to our setup, which achieves controllability through test-time latent optimization. This is a very strong baseline that generally works well but requires potentially lengthy optimization. We also compare to two ablations: *TRACE-Filter* samples from the diffusion model *without guidance* and chooses the best sample according to the guidance loss (similar to [284]), while *TRACE-Noisy* uses the guidance formulated in Eq. (5.7) from prior works [110, 314]. Models are trained on the combined dataset of *ORCA-Maps* (with map annotations) and *ORCA-Interact* (no map annotations). The guidance losses are: **None** samples randomly with no guidance; **Obstacle avoid** discourages collisions between map obstacles and pedestrian bounding boxes; **Agent avoid** discourages collisions between pedestrians by denoising all their futures in a scene jointly; **Waypoint** encourages a trajectory to pass through a goal at any point in the planning horizon. For this experiment, the waypoint is set as the position of each pedestrian at $4s$ into the

				Guidance	**Realism** (Mean)	
Guide	**Method**	**Train Data**	w	Error	Lon Acc	Lat Acc
Waypoint	VAE [210]	Mixed	–	**0.340**	0.193	0.172
	TRACE	nuScenes	-0.5	0.421	0.177	0.168
		Mixed	0.0	0.551	0.159	0.145
		Mixed	-0.5	0.366	**0.140**	**0.132**
Waypoint	VAE [210]	Mixed	–	0.962	0.443	0.441
perturbed	TRACE	nuScenes	-0.5	0.977	0.239	0.238
		Mixed	0.0	1.129	0.233	0.218
		Mixed	-0.5	**0.802**	**0.212**	**0.204**
Social	VAE [210]	Mixed	–	0.297	0.109	0.104
groups	TRACE	nuScenes	-0.5	0.287	0.155	0.158
		Mixed	0.0	**0.244**	0.110	0.101
		Mixed	-0.5	0.245	**0.094**	**0.087**

Table 5.2: Guidance evaluation on nuScenes. Training on mixed data and using $w<0$ for classifier-free sampling are important to achieve controllability for out-of-distribution objectives.

future in the ground truth data. These are *in-distribution* objectives, since they reinforce behavior already observed in the ground truth data.

In Tab. 5.1, TRACE successfully achieves all objectives through the proposed guidance. It is competitive or better than the VAE optimization in terms of guidance, while maintaining velocity and acceleration distributions closer to ground truth as indicated by *Realism*. Fig. 5.4 shows that random samples from the VAE contain collisions, and using latent optimization for guidance gives similar local minima across samples thereby limiting diversity compared to TRACE. Finally, using our proposed clean guidance (Eq. (5.8)) instead of the noisy version produces consistently better results in guidance and realism.

5.4.2 Real-world Data Evaluation

We next evaluate controllability when trained on real-world data, and focus on *out-of-distribution* (OOD) guidance objectives to emphasize the flexibility of our approach. In this experiment, methods operate in a *closed loop*: pedestrians are rolled out for $10s$ and re-plan at 1 Hz. Results on a held out nuScenes split are shown in Tab. 5.2. We compare TRACE trained on mixed data (ETH/UCY+nuScenes), after training on nuScenes only, and using two different classifier-free sampling weights w. Along with in-distribution **Waypoint** (now at $9s$), two additional objectives are evaluated: **Waypoint perturbed** uses a noisily perturbed ground truth future position (at $9s$),

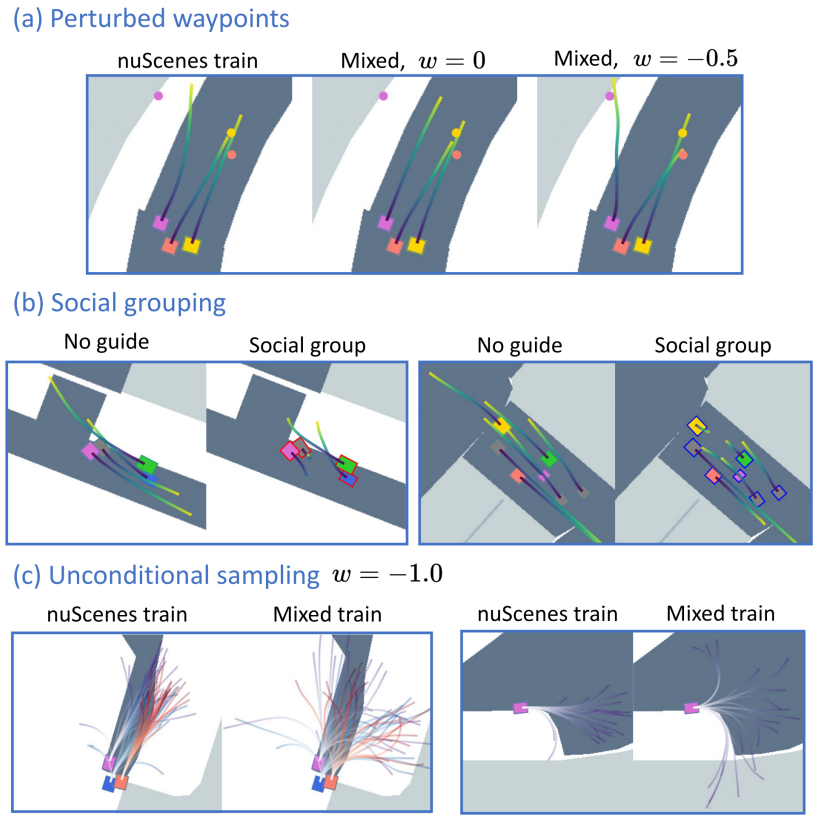

Figure 5.5: nuScenes results demonstrating flexibility of TRACE. (a) Using mixed training and $w{=}{-}0.5$ is best for noisy waypoints. (b) Social group guidance encourages sets of pedestrians to stay close. (c) Mixed training (ETH/UCY+nuScenes) learns a more diverse distribution as demonstrated by unconditional sampling.

requiring pedestrians to go off sidewalks or into streets to reach the goal; **Social groups** specifies groups of agents to stay close and travel together. Groups are set heuristically based on spatial proximity and velocity at initialization.

In Tab. 5.2, we observe that OOD flexibility requires (1) training on mixed data, and (2) classifier-free sampling. Since nuScenes data is less diverse (people tend to follow the sidewalk), TRACE trained on just nuScenes struggles to hit perturbed waypoints. Though the VAE is trained on mixed data, it seemingly cannot leverage diverse pedestrian dynamics from ETH/UCY when rolling out on nuScenes maps, limiting OOD success. TRACE reaches OOD objectives using classifier-free sampling with $w{=}{-}0.5$ to downweight the conditioning of the semantic map and leverage diverse trajectories learned from ETH/UCY. The flexibility of TRACE is further highlighted in Fig. 5.5.

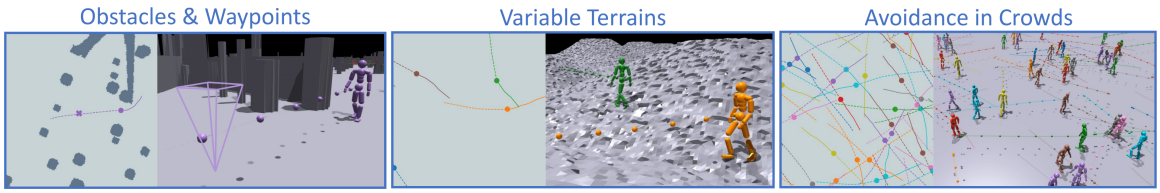

Figure 5.6: Our animation system enables traversing variable terrains, avoiding obstacles, meeting goals, and large crowds.

Terrain	**Guide**	**Fail** Rate	**Traj Follow** Error	**Discrim** Reward
Random	Procedural	0.133	0.680	1.950
	None	**0.093**	**0.104**	1.887
	Waypoint	0.107	0.111	**2.113**
Obstacles	Procedural	0.307	0.948	2.278
	None	0.125	0.093	2.512
	Obs Avoid	**0.063**	**0.089**	**2.521**
Flat (Crowd)	Procedural	0.127	0.371	2.320
	None	0.087	0.082	2.374
	Agt Avoid	**0.013**	**0.071**	**2.402**

Table 5.3: Closed-loop animation results. Our system successfully follows waypoints and avoids collisions in a variety of terrains, and additional guidance improves performance.

5.4.3 Controllable Pedestrian Animation

Finally, we demonstrate our full controllable pedestrian animation system. TRACE is trained on *ORCA* and used as a planner for the pre-trained PACER without any fine-tuning. We evaluate the animations by: *Fail Rate* measures the fraction of agents that fall down or collide with an obstacle or other agent, *Trajectory Following Error* measures the average deviation of the character from TRACE's plan, and *Discriminator Reward* is the mean reward returned by the adversarial motion prior used to train PACER, which measures how human-like a generated motion appears.

Tab. 5.3 evaluates the animations from our system using TRACE with and without guidance in various settings: *Random* is an assortment of smooth and rough slopes and stairs with varying difficulties, *Obstacles* is a flat terrain with large obstacles, and *Flat* is a flat terrain with pedestrians spawned in a crowd of 30. For each setting, 600 rollouts of $10s$ are simulated across 30 different characters. As a baseline to determine the difficulty of environments and reasonable discriminator rewards, we also include metrics when using the (terrain and obstacle unaware) *Procedural* trajectory generation method used to train PACER.

Terrain	Guide: Waypoint	Guide: Value	Waypoint Error	Fail Rate	Traj Follow Error	Discrim Reward
Random	√		0.541	0.107	**0.111**	2.113
	√	√	**0.481**	**0.100**	0.112	**2.162**
Obstacles	√		1.065	0.220	0.138	2.552
	√	√	**0.929**	**0.178**	**0.113**	**2.609**
Flat	√		0.248	0.063	**0.084**	2.555
(Crowd)	√	√	**0.175**	**0.053**	**0.084**	**2.607**

Table 5.4: Using the value function learned in RL training as guidance improves quality of trajectory following and robustness to varying terrain, obstacles, and other agents.

Our combined system performs well in the physically-simulated environment with TRACE providing easy-to-follow trajectories resulting in high-quality animations from PACER as evalutated by the discriminator. Diffusion guidance can further improve failure rates, especially for avoiding agent collisions in dense crowds. Fig. 5.6 shows some qualitative applications of our animation system. Tab. 5.4 shows the effect of using the learned value function from training PACER as a guidance loss for TRACE. In each setting, adding value guidance in addition to waypoint guidance makes trajectories easier to follow, reduces failures, and improves the discriminator reward. As a result, waypoint guidance error also improves.

5.5 Discussion

We have introduced a controllable trajectory diffusion model, a robust physics-based humanoid controller, and an end-to-end animation system that combines the two. Developing TRACE required addressing several of the key features discussed in Sec. 1.1.1. To get *accuracy and diversity*, it was necessary to adapt recent diffusion models to handle conditioning on context and generate plausible action sequences for pedestrian control. Part of this conditioning was a learned map feature grid that was queried at each denoising step to account for *interactions* between pedestrians and their environment. Lastly, *controllability* was enabled through guidance of the denoising process.

Limitations and Future Work. TRACE represents an exciting step in being able to control the high-level behavior of learned pedestrian models, and opens several directions for future work. First is improving the efficiency of sampling from trajectory diffusion models to make them feasible for real-time planning: currently TRACE takes 1-5s (depending on the guidance used) to generate a motion plan. Recent work in distilling diffusion models to work with only a handful of steps [169] offers a potential solution. In addition to high-level motion controllability, exploring how diffusion

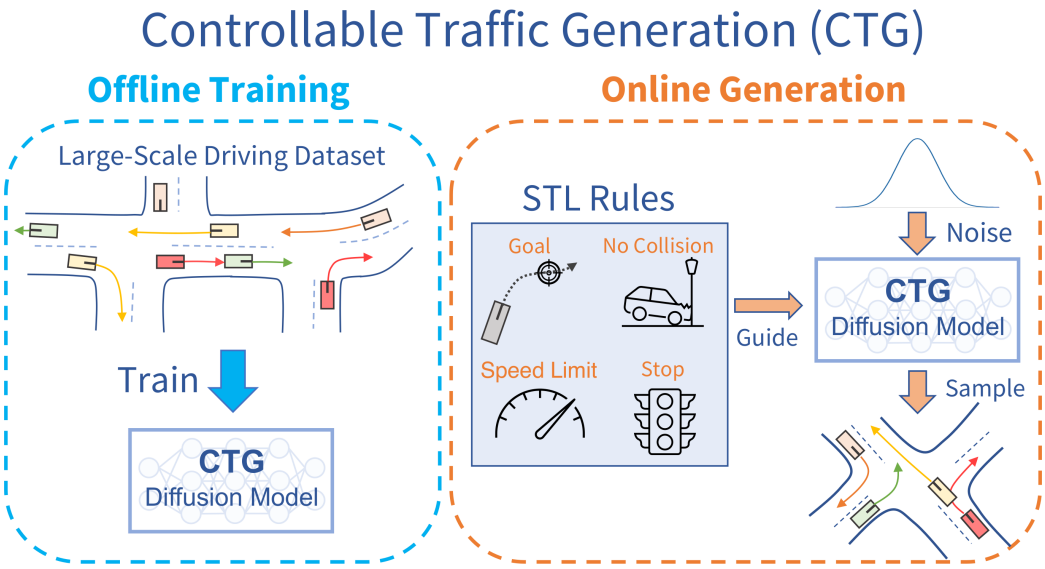

Figure 5.7: Controllable Traffic Generation (CTG) uses two key stages to enable controllable simulation. (Left) a conditional diffusion model is trained to generate realistic trajectories. (Right) Guided model sampling uses signal temporal logic (STL) rules to meet desired objectives.

models can be extended to low-level control can also be an interesting followup.

5.6 Additional Related Contributions

This section briefly describes additional contributions made to the area of modeling human behavior with 2D trajectories. In particular, Sec. 5.6.1 introduces CTG [314], which leverages a guided diffusion framework similar to TRACE for simulating vehicle traffic and ensuring road rules are met.

5.6.1 Controllable Traffic Generation

Controllable and realistic traffic simulation is critical for developing and verifying autonomous vehicles. Typical heuristic-based traffic models offer flexible control to make vehicles follow specific trajectories and traffic rules. On the other hand, data-driven approaches generate realistic and human-like behaviors, improving transfer from simulated to real-world traffic. However, to the best of our knowledge, no traffic model offers both controllability and realism. In this work, we develop a conditional diffusion model for controllable traffic generation (CTG) that allows users to control desired properties of trajectories at test time (e.g., reach a goal or follow a speed limit) while maintaining realism and physical feasibility through enforced dynamics. The key technical idea

is to leverage recent advances from diffusion modeling and differentiable logic to guide generated trajectories to meet rules defined using signal temporal logic (STL). We further extend guidance to multi-agent settings and enable interaction-based rules like collision avoidance. CTG is extensively evaluated on the nuScenes dataset for diverse and composite rules, demonstrating improvement over strong baselines in terms of the controllability-realism tradeoff.

Fig. 5.7 provides an overview of the key ideas of CTG. Please refer to the paper for full technical details and results [314].

Chapter 6

Conclusion and Future Vision

In this thesis, we have seen how learned models of motion can be employed to solve fundamental perception and generation problems for 3D humans, 3D objects, and 2D human behavior. To make these learned motion models effective, we developed architectures and state spaces that allowed the models to be accurate and diverse, aware of interactions, robust, generalizable, and controllable.

In Chapter 2, we saw with HuMoR that conditional VAEs can effectively capture the unobservable factors of human motion, and model the change in 3D pose at a single time step in a generalizable way. This allowed performing optimization in the learned latent space to robustly recover human motion from video, even under noise and occlusions. Chapter 3 explored modeling more general 3D object motion as the change in shape from the perspective of a sensor. For this purpose we introduced CaSPR, which used a flexible learned latent space to model the change in 3D object shape, making it agnostic to the type of observed motion. This enabled continuously reconstructing point cloud sequences for both rigid and deformable objects and, more generally, helped solve several perception problems from dynamic point cloud inputs.

In Chapters 4 and 5, we leveraged powerful generative models to capture realistic vehicle and pedestrian trajectories. STRIVE (Chapter 4) used a scene-centric conditional VAE that captured interactions between vehicles using message passing layers within a graph neural network. The learned latent space of the VAE allowed test-time optimization to create traffic scenarios that meet user specifications, such as causing or avoiding an accident. TRACE (Chapter 5) introduced a denoising-diffusion model that queried a learned map feature grid throughout the denoising process to reason about local interactions between pedestrians and their environment. By guiding the denoising process, it also enabled users to specify objectives for trajectories, such as goal waypoints, collision avoidance, or social groups.

Each of these works offers promising progress towards enabling intelligent systems to successfully understand and leverage motion. However, there are still many problems to solve to fully achieve real-world embodied systems and simulations teeming with realistic dynamic agents. This leaves exciting future directions in generating and perceiving motion:

3D Human Motion Synthesis. On the heels of generative approaches like HuMoR (Chapter 2), it is clear that data-driven models for 3D human motion have the potential to greatly impact the field of character animation and simulation. We envision a character animation system that is intuitive, yet flexible to be controlled by a user: digital characters can be dropped in a 3D scene, given direction from the user (like a task to perform specified by natural language), and then they will realistically interact with the world to carry out their tasks.

Achieving such a system requires developing three levels of motion understanding: (1) high-level reasoning, (2) navigation, and (3) low-level pose control. For *high-level reasoning*, characters must ingest commands from a user and make a plan to carry them out. This requires breaking a high level language command (*e.g.*, "make some pasta") into a series of sub-commands that can be used to successfully prompt a language-conditioned motion model [197, 251, 307] (*e.g.*, "walk forward to the kitchen" $\rightarrow$ "open the cupboard" $\rightarrow$ "grab the box of pasta" $\rightarrow \cdots$). To carry out a plan, the character must *navigate* in their environment to avoid collisions with static and dynamic obstacles [28]. In contrast to traditional path planning algorithms that produce the shortest possible non-colliding path, characters should carry out realistic and semantically-plausible paths that adhere to social norms or rules of the environment.

The *full 3D pose* of the character must be synthesized to carry out the navigation and desired actions. An open challenge here is ensuring generated poses realistically interact with the surrounding environment [92, 310]. From sitting in chairs, to laying in beds, to picking up and manipulating objects, motions must exhibit physically-realistic contacts and avoid unrealistic penetrations. One interesting direction towards this end is mixing physics-based and data-driven models to get the best of both worlds [296]. Motions should also be semantically plausible as to interact with objects realistically based on affordances, *e.g.*, picking up a mug by the handle. These kinds of interactions can be seen as one way to *control* the motions generated by a model. In general, we would like to develop generative models that can produce motions at test time to meet both soft objectives and hard constraints. While TRACE (Chapter 5) makes progress in this direction for 2D trajectories using a guided diffusion model, similar capabilities must be developed for full 3D pose so that users can generate motion to meet language or action commands [251, 307], keyframe poses [48], specific motion styles [254], or environment geometry [107].

Modeling Human Behavior. The ability to generate realistic 2D trajectories for humans to follow (*i.e.*, human behavior) is useful beyond character animation. For example, populating simulations of indoor and outdoor spaces, streets, and cities requires dynamic pedestrians, vehicles, and cyclists. However, trajectories must adhere to constraints imposed by the environment, such as avoiding collisions between agents and following rules of the road. While recent progress can avoid collision and stay on-road for short-term rollouts (*e.g.*, STRIVE and TRACE in Chapters 4 and 5), it becomes more challenging for long-term simulations that must remain stable and robust for several minutes [284]. The work in this thesis shows that iterative test time procedures, either through optimization or denoising-diffusion, are promising for enforcing constraints. However, efficiency must be improved to make them feasible in real-time settings. Another potential solution is through reinforcement learning (RL), which can produce behavior policies that successfully adhere to rules through rewards, but do not always give human-like motions. Therefore, some combination of supervised learning and RL may give both realistic motion and rule-following [16].

STRIVE demonstrates that generative traffic models can be useful for automatically creating accident scenarios; an interesting direction is extending this to creating scenarios from higher-level user directions. For example, a user may use natural language to specify the personality of a certain driver (*e.g.*, "agressive"), the maneuver to make (*e.g.*, "turn left then go straight"), or a description at the scene level (*e.g.*, "traffic jam at a 4-way intersection").

Learned Motion from Limited Supervision. The work in this thesis relies heavily on having a large-scale, high-quality, and diverse dataset of motion to learn from. This includes human motion capture (mocap) data, object point cloud observations, or vehicle and pedestrian trajectories. However, these data sources are not always available: for animals, or humans interacting with objects and scenes, clean mocap data is scarce and it is not scalable to collect more [291]. It is expensive to capture every single type of motion we wish to model and in some cases, like simultaneously capturing human and object pose, capture can be challenging due to heavy occlusions. One potential avenue is using weakly supervised methods to learn 3D motion models directly from widely available videos, or other data with limited 2D annotations or noisy 3D annotations. Recent advancements in generative video models [102] could even be harnessed to create free data for some types of motion.

Improving Dynamic Perception. Finally, we must harness improved generation capabilities for human and object motion for perception. While HuMoR follows analysis-by-synthesis by optimizing for human motion in a learned latent space, this test-time procedure can be slow and prone to local minima. Therefore, it is important to explore other ways these models can inform perception. Moreover, it is not clear how other families of generative models can be used as a prior for perception:

while VAEs are amenable to optimization in a low-dimensional latent space, diffusion models do not afford such a formulation. The motion models in this thesis are also class specific, and assume ahead of time what type of entity is being modeled. For general purpose dynamic perception, it will be interesting to explore what kind of class-agnostic motion priors can be developed. For example, learning point/pixel-level [91] motion priors or part-based representations [286].

The work in this thesis focuses on perceiving humans and objects in relative isolation. However, in-the-wild observations have complex interactions with the environment, making it important to consider recovering motion and scene geometry together [292]. Incorporating physics-based priors or differentiable physics simulation could be particularly useful in this case [232, 87]. Real-world observations may also be from a moving sensor, so it will be necessary to have approaches that disambiguate between observer motion and the motion of the observed entity [289].

Overall, we anticipate that the efforts in this thesis are just the beginning of the exciting and challenging area of learned motion modeling, and we expect future breakthroughs will help to solve fundamental perception and generation problems in computer vision.

Bibliography

[1] Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277. IEEE, 2019.

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 40–49. PMLR, 2018.

[3] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. Structured prediction helps 3d human motion modelling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7144–7153, 2019.

[4] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[5] Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Lars Petersson, and Stephen Gould. A stochastic conditioning scheme for diverse human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5223–5232, 2020.

[6] Anurag Arnab, Carl Doersch, and Andrew Zisserman. Exploiting temporal context for 3d human pose estimation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3395–3404, 2019.

[7] Andrei Atanov, Alexandra Volokhova, Arsenii Ashukha, Ivan Sosnovik, and Dmitry Vetrov. Semi-conditional normalizing flows for semi-supervised learning. *arXiv preprint arXiv:1905.00505*, 2019.

[8] Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Consumer Depth Cameras for Computer Vision*, pages 71–98. Springer, 2013.

[9] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics: Science and Systems (RSS)*, 2019.

[10] Albert E Beaton and John W Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974.

[11] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7962–7971, 2019.

[12] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1171–1179, Cambridge, MA, USA, 2015. MIT Press.

[13] Jur van den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium (ISRR)*, pages 3–19. Springer, 2011.

[14] Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Błażej Osiński, Hugo Grimmet, and Peter Ondruska. Simnet: Learning reactive self-driving simulations from real-world observations. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[15] Dimitri P Bertsekas. A distributed asynchronous relaxation algorithm for the assignment problem. In *IEEE Conference on Decision and Control*, pages 1703–1704, 1985.

[16] Raunak Bhattacharyya, Blake Wulfe, Derek J Phillips, Alex Kuefler, Jeremy Morton, Ransalu Senanayake, and Mykel J Kochenderfer. Modeling human driving behavior through generative adversarial imitation learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[17] Benjamin Biggs, David Novotny, Sebastien Ehrhardt, Hanbyul Joo, Ben Graham, and Andrea Vedaldi. 3d multi-bodies: Fitting sets of plausible 3d human models to ambiguous image data. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[18] Marin Biloš and Stephan Günnemann. Equivariant normalizing flows for point processes and sets. *arXiv preprint arXiv:2010.03242*, 2020.

[19] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science. Springer International Publishing, October 2016.

[20] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, pages 10–21. Association for Computational Linguistics (ACL), 2016.

[21] Matthew Brand and Aaron Hertzmann. Style machines. In *ACM SIGGRAPH*, pages 183–192, July 2000.

[22] Robert Bridson. *Fluid simulation for computer graphics*. AK Peters/CRC Press, 2015.

[23] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH 2005 Courses*, pages 3–es. 2005.

[24] Marcus A. Brubaker, David J. Fleet, and Aaron Hertzmann. Physics-based person tracking using the anthropomorphic walker. *IJCV*, (1), 2010.

[25] Marcus A. Brubaker, Leonid Sigal, and David J. Fleet. Estimating contact dynamics. In *Proc. ICCV*, 2009.

[26] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[27] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[28] Zhe Cao, Hang Gao, Karttikeya Mangalam, Qi-Zhi Cai, Minh Vo, and Jitendra Malik. Long-term human motion prediction with scene context. In *European Conference on Computer Vision*, pages 387–404. Springer, 2020.

[29] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 624–641. Springer, 2020.

[30] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021.

[31] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning (CoRL)*, pages 86–99. PMLR, 2020.

[32] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[33] Baiming Chen, Xiang Chen, Qiong Wu, and Liang Li. Adversarial evaluation of autonomous vehicles in lane-change scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[34] Changyou Chen, Chunyuan Li, Liqun Chen, Wenlin Wang, Yunchen Pu, and Lawrence Carin. Continuous-time flows for efficient inference and density estimation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[35] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11973–11982, 2020.

[36] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[37] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6572–6583, 2018.

[38] Scott Saobing Chen and Ramesh A Gopinath. Gaussianization. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 423–429, 2001.

[39] Yixin Chen, Siyuan Huang, Tao Yuan, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. Holistic++ scene understanding: Single-view 3d holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8648–8657, 2019.

[40] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5948, 2019.

[41] Vasileios Choutas, Georgios Pavlakos, Timo Bolkart, Dimitrios Tzionas, and Michael J Black. Monocular expressive body regression through body-driven attention. In *European Conference on Computer Vision*, pages 20–40. Springer, 2020.

[42] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019.

[43] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 628–644. Springer, 2016.

[44] Earl A Coddington and Norman Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

[45] Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. *arXiv preprint arXiv:1909.13833*, 2019.

[46] Enric Corona, Albert Pumarola, Guillem Alenya, and Francesc Moreno-Noguer. Context-aware human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.

[47] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019.

[48] Rishabh Dabral, Muhammad Hamza Mughal, Vladislav Golyanik, and Christian Theobalt. Mofusion: A framework for denoising-diffusion-based motion synthesis. In *Computer Vision and Pattern Recognition (CVPR)*, 2023.

[49] Jared Quincy Davis, Krzysztof Choromanski, Vikas Sindhwani, Jake Varley, Honglak Lee, Jean-Jacques Slotine, Valerii Likhosterov, Adrian Weller, and Ameesh Makadia. Time dependence in non-autonomous neural odes. In *ICLR Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[50] Patrick Dendorfer, Aljosa Osep, and Laura Leal-Taixé. Goal-gan: Multimodal trajectory prediction based on goal position estimation. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[51] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[52] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 195–205, 2018.

[53] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *European Conference on Computer Vision*, pages 715–733. Springer, 2020.

[54] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[55] Wenhao Ding, Baiming Chen, Bo Li, Kim Ji Eun, and Ding Zhao. Multimodal safety-critical scenarios generation for decision-making algorithms evaluation. *IEEE Robotics and Automation Letters*, 6(2):1551–1558, 2021.

[56] Wenhao Ding, Baiming Chen, Minjun Xu, and Ding Zhao. Learning to collide: An adaptive safety-critical scenarios generating method. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2243–2250. IEEE, 2020.

[57] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2095–2104, 2020.

[58] Carl Doersch and Andrew Zisserman. Sim2real transfer learning for 3d human pose estimation: motion to the rescue. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:12949–12961, 2019.

[59] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[60] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 3134–3144, 2019.

[61] Arthur Eddington. *The nature of the physical world: The Giffor Lectures 1927*, volume 23. BoD–Books on Demand, 2019.

[62] Ahmed Elgammal and Chan-Su Lee. Separating style and content on a nonlinear manifold. In *IEEE Conf. Comp. Vis. and Pattern Recognition*, pages 478–485, 2004. Vol. 1.

[63] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. *ICCV*, 2021.

[64] Anthony C Fang and Nancy S Pollard. Efficient synthesis of physically valid human motion. *Acm transactions on graphics (tog)*, 22(3):417–426, 2003.

[65] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020.

[66] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[67] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. *arXiv preprint arXiv:2002.02798*, 2020.

[68] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[69] Jerome H Friedman. Exploratory projection pursuit. *Journal of the American statistical association*, 82(397):249–266, 1987.

[70] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. *ArXiv*, abs/2210.10044, 2022.

[71] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real time motion capture using a single time-of-flight camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 755–762. IEEE, 2010.

[72] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

[73] S. Geman and D. McClure. Statistical methods for tomographic image reconstruction. *Bulletin of the International Statistical Institute*, 52(4):5–21, 1987.

[74] Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

[75] Georgios Georgakis, Ren Li, Srikrishna Karanam, Terrence Chen, Jana Košecká, and Ziyan Wu. Hierarchical kinematic human mesh recovery. In *European Conference on Computer Vision*, pages 768–784. Springer, 2020.

[76] Zahra Ghodsi, Siva Kumar Sastry Hari, Iuri Frosio, Timothy Tsai, Alejandro Troccoli, Stephen W Keckler, Siddharth Garg, and Anima Anandkumar. Generating and characterizing scenarios for safety testing of autonomous vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 157–164. IEEE, 2021.

[77] Saeed Ghorbani, Calden Wloka, Ali Etemad, Marcus A Brubaker, and Nikolaus F Troje. Probabilistic character motion synthesis using a hierarchical deep latent variable model. In *Computer Graphics Forum*, volume 39. Wiley Online Library, 2020.

[78] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

[79] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018.

[80] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[81] Tianpei Gu, Guangyi Chen, Junlong Li, Chunze Lin, Yongming Rao, Jie Zhou, and Jiwen Lu. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17113–17122, 2022.

[82] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3263, 2019.

[83] Riza Alp Guler and Iasonas Kokkinos. Holopose: Holistic 3d human reconstruction in-the-wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10884–10894, 2019.

[84] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306, 2018.

[85] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2020.

[86] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264, 2018.

[87] Erik Gärtner, Mykhaylo Andriluka, Erwin Coumans, and Cristian Sminchisescu. Differentiable dynamics for articulated 3d human motion reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[88] Marc Habermann, Weipeng Xu, Michael Zollhofer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5052–5063, 2020.

[89] Ikhsanul Habibie, Daniel Holden, Jonathan Schwarz, Joe Yearsley, and Taku Komura. A recurrent variational autoencoder for human motion synthesis. In *28th British Machine Vision Conference*, 2017.

[90] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[91] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 59–75. Springer, 2022.

[92] Mohamed Hassan, Duygu Ceylan, Ruben Villegas, Jun Saito, Jimei Yang, Yi Zhou, and Michael J Black. Stochastic scene-aware motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11374–11384, 2021.

[93] Mohamed Hassan, Vasileios Choutas, Dimitrios Tzionas, and Michael J. Black. Resolving 3D human pose ambiguities with 3D scene constraints. In *International Conference on Computer Vision*, pages 2282–2292, 2019.

[94] Mohamed Hassan, Partha Ghosh, Joachim Tesch, Dimitrios Tzionas, and Michael J Black. Populating 3d scenes by learning human-scene interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14708–14718, 2021.

[95] M. Brandon Haworth, Glen Berseth, Seonghyeon Moon, Petros Faloutsos, and Mubbasir Kapadia. Deep integration of physical humanoid control and crowd navigation. *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, 2020.

[96] Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[97] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[98] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[99] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[100] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[101] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[102] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

[103] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[104] Mir Rayat Imtiaz Hossain and James J Little. Exploiting temporal information for 3d human pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 68–84, 2018.

[105] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. `https://level-5.global/level5/data/`, 2020.

[106] Nicholas R. Howe, Michael E. Leventon, and William T. Freeman. Bayesian reconstruction of 3D human motion from single-camera video. In *Advances in Neural Information Processing Systems 12*, pages 820–826, 2000.

[107] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. *arXiv preprint arXiv:2301.06015*, 2023.

[108] Yinghao Huang, Federica Bogo, Christoph Lassner, Angjoo Kanazawa, Peter V Gehler, Javier Romero, Ijaz Akhter, and Michael J Black. Towards accurate marker-less human shape and pose estimation over time. In *2017 International Conference on 3D Vision*, pages 421–430. IEEE, 2017.

[109] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019.

[110] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *International Conference on Machine Learning (ICML)*, 2022.

[111] Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)*, 38(4):1–12, 2019.

[112] Daniel Johnson, Hugo Larochelle, and Daniel Tarlow. Learning graph structure with a finite-state automaton layer. *Advances in Neural Information Processing Systems*, 33:3082–3093, 2020.

[113] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Regognition*, 2018.

[114] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[115] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1966–1974, 2015.

[116] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4551–4560, 2019.

[117] Ioannis Karamouzas, Nick Sohre, Ran Hu, and Stephen J Guy. Crowd space: a predictive crowd analysis technique. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018.

[118] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Level 5 perception dataset 2020. `https://level-5.global/level5/data/`, 2019.

[119] Jongmin Kim, Yeongho Seol, Taesoo Kwon, and Jehee Lee. Interactive manipulation of large-scale crowd animation. *ACM Transactions on Graphics (TOG)*, 33(4):1–10, 2014.

[120] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. DriveGAN: Towards a Controllable High-Quality Neural Simulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021.

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

[122] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[123] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[124] Moritz Klischat and Matthias Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2352–2358. IEEE, 2019.

[125] Ivan Kobyzev, Simon Prince, and Marcus A Brubaker. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257*, 2019.

[126] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2020.

[127] Muhammed Kocabas, Chun-Hao P Huang, Otmar Hilliges, and Michael J Black. Pare: Part attention regressor for 3d human body estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11127–11137, 2021.

[128] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

[129] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *Proceedings International Conference on Computer Vision (ICCV)*, pages 2252–2261. IEEE, October 2019. ISSN: 2380-7504.

[130] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1094–1099. IEEE, 2015.

[131] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[132] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM Transactions on Graphics 21(3), Proc. SIGGRAPH*, pages 473–482, July 2002.

[133] Wilhelm Kutta. Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys.*, 46:435–453, 1901.

[134] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6050–6059, 2017.

[135] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 156–165, 2017.

[136] Jaedong Lee, Jungdam Won, and Jehee Lee. Crowd simulation by deep reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, pages 1–7, 2018.

[137] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 336–345, 2017.

[138] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14, 2014.

[139] Marilena Lemonari, Rafael Blanco, Panayiotis Charalambous, Nuria Pelechano, Marios Avraamides, Julien Pettré, and Yiorgos Chrysanthou. Authoring virtual crowds: A survey. In *Computer Graphics Forum*, volume 41, pages 677–701. Wiley Online Library, 2022.

[140] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007.

[141] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3706–3715, 2020.

[142] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: A two-level statistical model for character motion synthesis. In *ACM Transactions on Graphics 21(3), Proc. SIGGRAPH*, pages 465–472, July 2002.

[143] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 820–830, 2018.

[144] Yiming Li, Congcong Wen, Felix Juefei-Xu, and Chen Feng. Fooling lidar perception via adversarial trajectory perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7898–7907, 2021.

[145] Zongmian Li, Jiri Sedlar, Justin Carpentier, Ivan Laptev, Nicolas Mansard, and Josef Sivic. Estimating 3d motion and forces of person-object interactions from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[146] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020.

[147] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. Character controllers using motion vaes. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, volume 39. ACM, 2020.

[148] C. Karen Liu, Aaron Hertzmann, and Zoran Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph*, 2005.

[149] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2019.

[150] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. In *Advances in Neural Information Processing Systems*, pages 13556–13566, 2019.

[151] Miao Liu, Dexin Yang, Yan Zhang, Zhaopeng Cui, James M Rehg, and Siyu Tang. 4d human body capture from egocentric video via 3d scene grounding. In *2021 international conference on 3D vision (3DV)*, pages 930–939. IEEE, 2021.

[152] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[153] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9246–9255, 2019.

[154] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7577–7586, 2021.

[155] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[156] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE, 2018.

[157] Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Neural manifold ordinary differential equations. *arXiv preprint arXiv:2006.10254*, 2020.

[158] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[159] Zhengyi Luo, S Alireza Golestaneh, and Kris M Kitani. 3d human motion estimation via motion compression and refinement. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[160] Zhengyi Luo, Ryo Hachiuma, Ye Yuan, and Kris Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. In *Advances in Neural Information Processing Systems*, 2021.

[161] Yecheng Jason Ma, Jeevana Priya Inala, Dinesh Jayaraman, and Osbert Bastani. Likelihood-based diverse sampling for trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13279–13288, 2021.

[162] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, 2019.

[163] Osama Makansi, Özgün Cicek, Yassine Marrakchi, and Thomas Brox. On exposing the challenging long tail in future prediction of traffic actors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13147–13157, 2021.

[164] Osama Makansi, Julius Von Kügelgen, Francesco Locatello, Peter Vincent Gehler, Dominik Janzing, Thomas Brox, and Bernhard Schölkopf. You mostly walk alone: Analyzing feature attribution in trajectory prediction. In *International Conference on Learning Representations*, 2022.

[165] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[166] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15233–15242, 2021.

[167] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. Xnect: Real-time multi-person 3d motion capture with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 39(4):82–1, 2020.

[168] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[169] Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. *arXiv preprint arXiv:2210.03142*, 2022.

[170] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019.

[171] Niloy J Mitra, Simon Flöry, Maks Ovsjanikov, Natasha Gelfand, Leonidas J Guibas, and Helmut Pottmann. Dynamic geometry registration. In *Symposium on geometry processing*, pages 173–182, 2007.

[172] Aron Monszpart, Paul Guerrero, Duygu Ceylan, Ersin Yumer, and Niloy J. Mitra. iMapper: Interaction-guided scene mapping from monocular videos. *ACM SIGGRAPH*, 2019.

[173] Aron Monszpart, Nils Thuerey, and Niloy J Mitra. Smash: physics-guided reconstruction of collisions from videos. *ACM Transactions on Graphics (TOG)*, 35(6):1–14, 2016.

[174] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[175] Wassim G Najm, John D Smith, and Mikio Yanagisawa. *Pre-Crash Scenario Typology for Crash Avoidance Research*. U.S. Department of Transportation, National Highway Traffic Safety Administration, 2007.

[176] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[177] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5379–5389, 2019.

[178] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7688–7697, 2019.

[179] Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *NeurIPS*, 2018.

[180] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. *arXiv preprint arXiv:2006.00104*, 2020.

[181] Dirk Ormoneit, Hedvig Sidenbladh, Michael J. Black, and Trevor Hastie. Learning and tracking cyclic human motion. In *Advances in Neural Information Processing Systems 13*, pages 894–900, 2001.

[182] Avik Pal, Jonah Philion, Yuan-Hong Liao, and Sanja Fidler. Emergent road rules in multi-agent driving environments. In *International Conference on Learning Representations*, 2020.

[183] Andreas Panayiotou, Theodoros Kyriakou, Marilena Lemonari, Yiorgos Chrysanthou, and Panayiotis Charalambous. Ccp: Configurable crowd profiles. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[184] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

[185] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[186] 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 *Advances in Neural Information Processing Systems*, 2017.

[187] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[188] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018.

[189] Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. Modeling human motion with quaternion-based neural networks. *International Journal of Computer Vision*, pages 1–18, 2019.

[190] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, 2019.

[191] Vladimir Pavlović, James M. Rehg, and John MacCormick. Learning switching linear models of human motion. In *Advances in Neural Information Processing Systems 13*, pages 981–987, 2001.

[192] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pages 261–268. IEEE, 2009.

[193] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, 36(4), 2017.

[194] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Trans. Graph.*, 41(4), July 2022.

[195] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Trans. Graph*, 2018.

[196] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, 40(4), July 2021.

[197] Mathis Petrovich, Michael J. Black, and Gül Varol. TEMOS: Generating diverse human motions from textual descriptions. In *European Conference on Computer Vision (ECCV)*, 2022.

[198] Jonah Philion, Amlan Kar, and Sanja Fidler. Learning to evaluate perception models using planner-centric metrics. In *CVPR*, 2020.

[199] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017.

[200] Lukas Prantl, Nuttapong Chentanez, Stefan Jeschke, and Nils Thuerey. Tranquil clouds: Neural networks for learning temporally coherent features in point clouds. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[201] Maria Priisalu, Ciprian Paduraru, Aleksis Pirinen, and Cristian Sminchisescu. Semantic synthesis of pedestrian locomotion. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[202] Maria Priisalu, Aleksis Pirinen, Ciprian Paduraru, and Cristian Sminchisescu. Generating scenarios with diverse pedestrian behaviors for autonomous vehicle testing. In *Conference on Robot Learning*, pages 1247–1258. PMLR, 2022.

[203] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2020.

[204] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[205] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, 2017.

[206] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *International Conference on Computer Vision (ICCV)*, 2021.

[207] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J. Guibas. Caspr: Learning canonical spatiotemporal point cloud representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[208] Davis Rempe, Leonidas J. Guibas, Aaron Hertzmann, Bryan Russell, Ruben Villegas, and Jimei Yang. Contact and human dynamics from monocular video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[209] Davis Rempe, Zhengyi Luo, Xue Bin Peng, Ye Yuan, Kris Kitani, Karsten Kreis, Sanja Fidler, and Or Litany. Trace and pace: Controllable pedestrian animation via guided trajectory diffusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[210] Davis Rempe, Jonah Philion, Leonidas J. Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[211] Davis Rempe, Srinath Sridhar, He Wang, and Leonidas Guibas. Predicting the physical dynamics of unseen 3d objects. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[212] Zhiguo Ren, Panayiotis Charalambous, Julien Bruneau, Qunsheng Peng, and Julien Pettré. Group modeling: A unified velocity-based approach. In *Computer Graphics Forum*, volume 36, pages 45–56. Wiley Online Library, 2017.

[213] Cinjon Resnick, Or Litany, Amlan Kar, Karsten Kreis, James Lucas, Kyunghyun Cho, and Sanja Fidler. Causal bert: Improving object detection by searching for challenging groups.

In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2972–2981, October 2021.

[214] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[215] Danilo Jimenez Rezende, Sébastien Racanière, Irina Higgins, and Peter Toth. Equivariant hamiltonian flows. *arXiv preprint arXiv:1909.13739*, 2019.

[216] Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

[217] Chris Rockwell and David F Fouhey. Full-body awareness from partial observations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 522–539, 2020.

[218] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), November 2017.

[219] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[220] Yulia Rubanova, Tian Qi Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[221] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[222] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2021.

[223] Carl Runge. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895.

[224] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020.

[225] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956. IEEE, 2019.

[226] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[227] Hadi Salman, Payman Yadollahpour, Tom Fletcher, and Kayhan Batmanghelich. Deep diffeomorphic normalizing flows. *arXiv preprint arXiv:1810.03256*, 2018.

[228] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020.

[229] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[230] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 4, 2018.

[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 Transactions on Graphics (TOG)*, 40(1):1–15, 2020.

[232] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, Patrick Pérez, and Christian Theobalt. Neural monocular 3d human motion capture with physical awareness. *ACM Transactions on Graphics*, 40(4), aug 2021.

[233] Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Trans. Graph.*, 39(6), November 2020.

[234] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2937, 2013.

[235] Hedvig Sidenbladh, Michael J. Black, and David J. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *ECCV*, pages 702–718, 2000. Part II.

[236] 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. *International journal of computer vision*, 87(1-2):4, 2010.

[237] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2437–2446, 2019.

[238] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[239] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 3483–3491. Curran Associates, Inc., 2015.

[240] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[241] Srinath Sridhar, Davis Rempe, Julien Valentin, Sofien Bouaziz, and Leonidas J. Guibas. Multiview aggregation for learning category-specific shape reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[242] Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6):209–1, 2019.

[243] Colton Stearns, Davis Rempe, Jie Li, Rares Ambrus, Vitor Guizilini, Sergey Zakharov, Yanchao Yang, and Leonidas J. Guibas. Spot: Spatiotemporal modeling for 3d object tracking. In *European Conference on Computer Vision (ECCV)*, 2022.

[244] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539, 2018.

[245] Haoliang Sun, Ronak Mehta, Hao H Zhou, Zhichun Huang, Sterling C Johnson, Vivek Prabhakaran, and Vikas Singh. Dual-glow: Conditional flow-based generative model for modality transfer. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10611–10620, 2019.

[246] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[247] Yu Sun, Yun Ye, Wu Liu, Wenpeng Gao, Yili Fu, and Tao Mei. Human mesh recovery from monocular images via a skeleton-disentangled representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5349–5358, 2019.

[248] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021.

[249] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[250] Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. Modeling human motion using binary latent variables. In *Proc. NIPS*, 2007.

[251] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Amit H Bermano, and Daniel Cohen-Or. Human motion diffusion model. In *International Conference on Learning Representations (ICLR)*, 2022.

[252] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6411–6420, 2019.

[253] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[254] Jonathan Tseng, Rodrigo Castellon, and C Karen Liu. Edge: Editable dance generation from music. *arXiv preprint arXiv:2211.10658*, 2022.

[255] Shuhei Tsuchida, Satoru Fukayama, Masahiro Hamasaki, and Masataka Goto. Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 501–510, Delft, Netherlands, November 2019.

[256] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020.

[257] Christopher Urmson, Joshua Anhalt, J. Andrew (Drew) Bagnell, Christopher R. Baker, Robert E. Bittner, John M. Dolan, David Duggins, David Ferguson, Tugrul Galatali, Hartmut Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Alonzo Kelly, David Kohanbash, Maxim Likhachev, Nick Miller, Kevin Peterson, Raj Rajkumar, Paul Rybski, Bryan Salesky, Sebastian Scherer, Young-Woo Seo, Reid Simmons, Sanjiv Singh, Jarrod M. Snider, Anthony (Tony) Stentz, William (Red) L. Whittaker, and Jason Ziglar. Tartan racing: A multi-modal approach to the darpa urban challenge. Technical report, Carnegie Mellon University, Pittsburgh, PA, April 2007.

[258] Raquel Urtasun, David J Fleet, and Pascal Fua. 3d people tracking with gaussian process dynamical models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 238–245. IEEE, 2006.

[259] Raquel Urtasun, David J. Fleet, and Pascal Fua. Temporal motion models for monocular and multiview 3D human body tracking. *CVIU*, 104(2):157–177, 2006.

[260] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

[261] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*, pages 1928–1935. Ieee, 2008.

[262] Kiran Varanasi, Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Temporal surface tracking using mesh evolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 30–43. Springer, 2008.

[263] Gül Varol, Duygu Ceylan, Bryan Russell, Jimei Yang, Ersin Yumer, Ivan Laptev, and Cordelia Schmid. BodyNet: Volumetric inference of 3D human body shapes. In *ECCV*, 2018.

[264] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 40(6):1510–1517, 2017.

[265] Gül Varol, Ivan Laptev, Cordelia Schmid, and Andrew Zisserman. Synthetic humans for action recognition from unseen viewpoints. *International Journal of Computer Vision*, 129(7):2264–2287, 2021.

[266] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 109–117, 2017.

[267] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1999.

[268] Sai Vemprala and Ashish Kapoor. Adversarial attacks on optimization based planners. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9943–9949. IEEE, 2021.

[269] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[270] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[271] Matt Vitelli, Yan Chang, Yawei Ye, Maciej Wołczyk, Błażej Osiński, Moritz Niendorf, Hugo Grimmett, Qiangui Huang, Ashesh Jain, and Peter Ondruska. Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. *arXiv preprint arXiv:2109.13602*, 2021.

[272] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[273] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2642–2651, 2019.

[274] Jack M. Wang. Gaussian process dynamical models for human motion. Master's thesis, University of Toronto, 2005.

[275] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007.

[276] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021.

[277] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[278] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019.

[279] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 91–98, 2020.

[280] Xinshuo Weng and Kris Kitani. A baseline for 3d multi-object tracking. *arXiv preprint arXiv:1907.03961*, 1(2):6, 2019.

[281] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Physics-based character controllers using conditional vaes. *ACM Transactions on Graphics (TOG)*, 41(4):1–12, 2022.

[282] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[283] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10974, 2019.

[284] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. *arXiv preprint arXiv:2208.12403*, 2022.

[285] Dongseok Yang, Doyeon Kim, and Sung-Hee Lee. LoBSTr: Real-time Lower-body Pose Prediction from Sparse Upper-body Tracking Signals. *Computer Graphics Forum*, 2021.

[286] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2863–2873, 2022.

[287] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4541–4550, 2019.

[288] Zhangsihao Yang, Or Litany, Tolga Birdal, Srinath Sridhar, and Leonidas Guibas. Continuous geodesic convolutions for learning on 3d shapes. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[289] Vickie Ye, Georgios Pavlakos, Jitendra Malik, and Angjoo Kanazawa. Decoupling human and camera motion from videos in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[290] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11824–11833, 2020.

[291] Hongwei Yi, Chun-Hao P. Huang, Shashank Tripathi, Lea Hering, Justus Thies, and Michael J. Black. MIME: Human-aware 3D scene generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[292] Hongwei Yi, Chun-Hao P. Huang, Dimitrios Tzionas, Muhammed Kocabas, Mohamed Hassan, Siyu Tang, Justus Thies, and Michael J. Black. Human-aware object placement for visual environment reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3959–3970, June 2022.

[293] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[294] Wenhao Yu, Greg Turk, and C. Karen Liu. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)*, 37:1 – 12, 2018.

[295] Ye Yuan and Kris Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. In *European Conference on Computer Vision*, pages 346–364. Springer, 2020.

[296] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. *arXiv preprint arXiv:2212.02500*, 2022.

[297] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[298] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[299] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12224–12233, 2020.

[300] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. In *European Conference on Computer Vision*, pages 465–481. Springer, 2020.

[301] Andrei Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. Monocular 3d pose and shape estimation of multiple people in natural scenes-the importance of multiple scene constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2148–2157, 2018.

[302] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.

[303] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[304] Chaoyun Zhang, Marco Fiore, Iain Murray, and Paul Patras. Cloudlstm: A recurrent neural model for spatiotemporal point-cloud stream forecasting. *arXiv preprint arXiv:1907.12410*, 2019.

[305] Jason Y Zhang, Panna Felsen, Angjoo Kanazawa, and Jitendra Malik. Predicting 3d human dynamics from video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7114–7123, 2019.

[306] Linfeng Zhang, Lei Wang, et al. Monge-ampere flow for generative modeling. *arXiv preprint arXiv:1809.10188*, 2018.

[307] 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 preprint arXiv:2208.15001*, 2022.

[308] Shiwen Zhang, Sheng Guo, Weilin Huang, Matthew R. Scott, and Limin Wang. V4d: 4d convonlutional neural networks for video-level representation learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[309] Tianshu Zhang, Buzhen Huang, and Yangang Wang. Object-occluded human shape and pose estimation from a single color image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[310] Xiaohan Zhang, Bharat Lal Bhatnagar, Sebastian Starke, Vladimir Guzov, and Gerard Pons-Moll. Couch: Towards controllable human-chair interactions. In *European Conference on Computer Vision (ECCV)*. Springer, October 2022.

[311] Yan Zhang, Michael J Black, and Siyu Tang. We are more than our joints: Predicting how 3d bodies move. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3372–3382, 2021.

[312] Yan Zhang and Siyu Tang. The wanderings of odysseus in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20481–20491, 2022.

[313] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1009–1018, 2019.

[314] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. *International Conference on Robotics and Automation (ICRA)*, 2023.