Data Processing for Perception of Autonomous Vehicles in Urban Traffic Environments

by

Wonhui Kim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in The University of Michigan
2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AVs	Autonomous Vehicles
LiDAR	Light Detection and Ranging
RADAR	Radio Detection and Ranging
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
BEV	Bird's-Eye View
HD	High-Definition
MoCap	Motion Capture
SLAM	Simultaneous Localization And Mapping
GMM	Gaussian Mixture Model
MPJPE	Mean Per Joint Position Error
IoU	Intersection-over-Union

ABSTRACT

The perception system of Autonomous Vehicles (AVs), consisting of various on-board sensors including cameras and Light Detection and Ranging (LiDAR) scanners, is crucial to perceiving the environment, localizing the vehicle, and recognizing the semantics of traffic scenes. Despite recent advances in computer vision, the perception of AVs has remained challenging, especially in urban environments. One of the reasons is that multiple types of dynamic agents usually coexist in an urban area. The interactions between agents and surroundings are complicated to explicitly model, and the agents' unpredictable behaviors also increase the problem complexity. Moreover, ground truth data with a rich set of labels is not sufficient to cover diverse scenarios, and it is challenging to get data from real scenes.

This dissertation presents research contributions to overcome the challenges of AV perception in urban traffic environments, from data collection and labeling to 3D reconstruction and analysis of intrinsic properties. Mainly focusing on unsignalized urban intersections, we discuss (1) how to obtain valuable data from real urban traffic scenes, (2) how to efficiently process the raw data to produce meaningful labels for the dynamic road agents, (3) how to augment the data with semantic labels to the scenes, and (4) what factors make the reconstruction more realistic.

We first built a data capture system with a multi-modal sensor suite to simulate actual AV perception. We then introduced a 3D model-fitting algorithm to fit parametrized human mesh models to the pedestrians in a scene. The generated 3D models provide free labels, such as human pose and trajectories, with no cost of manual labeling. We proposed performing the entire scene modeling through densely reconstructing the scene and expanding the scope of automatic labeling to scene elements. These include dynamic vehicles and static components, such as roads, buildings, and traffic signs. To do this, we built a simulator that can generate a rich set of labels using virtual sensors. Finally, we tackled the problem of estimating intrinsic properties and discuss ways to achieve realistic 3D reconstruction.

This dissertation understands the AV perception pipeline, explores data preparations at urban traffic scenes, and discusses relevant experiments and applications critical for tackling other problems. We conclude the dissertation with future research directions for further augmenting the data and improving the realism of the reconstructed scene models.

CHAPTER I

Introduction

1.1 Background: Perception in Driving

The perception process of human driving typically follows three steps. A driver observes surroundings using sensory systems, a brain processes sensor inputs into meaningful information, and the driver responds with specific actions. Human driving is the repetition of these steps while maximizing safety. Of the human sensory systems, vision is inarguably the most crucial one. The majority of information necessary to perceive surroundings and to identify potential risks is acquired through human eyes.

Many current AV systems [6,7] are designed to involve perception mechanisms similar to human drivers. Various onboard sensors, such as cameras, LiDAR sensors, Radio Detection and Ranging (RADAR) sensors, Global Positioning System (GPS), and Inertial Measurement Unit (IMU), replace the role of human vision in self-localizing, sensing the environment, and recognizing the semantics of traffic scenes. In particular, LiDAR sensors and cameras have been equipped with most AVs to support real-time computer vision algorithms for higher-dimensional recognition and reasoning.

For example, AV perception systems need lane detection to position vehicles within lanes or plan paths for the later planning stage. Many recently proposed lane detection algorithms use learning-based approaches based on deep neural networks by taking camera images or LiDAR remission maps [8–11] as inputs. Traffic signalization detection algorithms also rely heavily on camera sensor inputs to recognize traffic lights [12] or to detect traffic signs [13,14]. As one of the core components of AV perception systems, object detectors are exploited to perceive environments using camera images or LiDAR points. Detectors for static obstacles, such as curbs, signposts, and lampposts, are used to build offline obstacle maps. Dynamic objects, such as vehicles and pedestrians, need to be detected and tracked as well to avoid collisions.

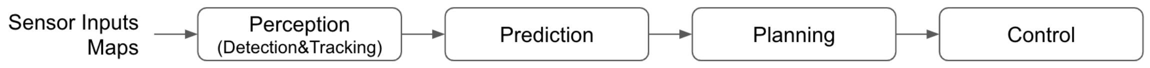


Figure 1.1: Typical AV engineering pipeline.

Perception results can be taken as inputs to other components of AV pipelines. In a typical AV pipeline (Figure 1.1), the perception module (detector and tracker) is followed by prediction and planning modules. Predicting the future states of traffic agents helps understand the future uncertainty of traffic agents and improves the safety and efficiency of navigation during planning.

1.2 Challenges of AV Perception

Perception tasks that involve dynamic objects in a traffic scene notably cause many challenges due to the increased complexity and unpredictable behaviors. To be specific, the motion of agents is stochastic, trajectories are dependent on each agent's goal, and multiple agents are involved with interactions with other agents and scene context. Non-rigid objects such as pedestrians are even more challenging to model and to fully understand their attributes.

Many state-of-the-art perception algorithms are implemented in a data-driven way using learning-based approaches. The key to learning quality models is the availability of unbiased, large-scale, high-quality data with task-specific labels. Gathering data in urban driving scenes, however, is very challenging. Driving scenes generally cover broad areas that can barely be controlled in reality. For example, varying light conditions depending on different times of day and weather conditions may affect algorithm performance, but collecting data under varying conditions is costly and time-consuming. The complexity of dynamic object behaviors also addresses the challenges in labeling.

Once any perception algorithms are developed, it is challenging to test the algorithms in actual end-to-end AV pipelines due to the lack of safe testing environments and the cost of building an AV system. The hardware components and the algorithms for the remaining parts of AV pipelines (Figure 1.1) need to be developed as well to make the pipeline work.

1.3 Problem Statement

1.3.1 Data Collection in Urban Traffic Scenes

The underlying motivation of this dissertation is to learn from the visual perceptions of human drivers in order for AVs to achieve human-like perceptions with a main focus on urban traffic scenes. An essential prerequisite to achieving this goal is collecting meaningful data of real traffic scenarios. However, it is not a simple task to capture real traffic data in an uncontrolled setting, especially in complex urban areas. Configuring a multi-modal sensor suite and collecting data, as in general AV setups, are challenging and require several different engineering efforts. Moreover, most of the existing traffic data capture systems are not designed to observe the detailed motions of dynamic road agents. In Chapter II, we describe our data capture system and the details of the capturing process used for unsignalized intersections in a complex urban area where many dynamic objects interact with surroundings in complicated ways. The sensor suite of our capture system consists of high-resolution stereo camera pairs and multiple spinning LiDAR sensors.

1.3.2 Labeling of Pedestrians at Urban Intersections

The raw captured data can be extensively used with meaningful annotations. In general, the labeling process to obtain annotations is costly and time-consuming. Moreover, the types of annotations needed vary depending on different tasks. In Chapter III, we explore a way to efficiently process the raw data to produce meaningful pedestrian labels. The semi-automatic labeling pipeline introduced is based on a 3D model fitting to generate a rich set of annotations.

1.3.3 Dense Reconstruction of Urban Intersections

We consider the problem of dense 3D reconstruction of real-world urban intersections in Chapter IV. We aim to achieve two major outcomes by reconstruction. First, the reconstructed scene models can produce a rich set of semantic 2D/3D annotations covering various object categories. Second, we can further generate simulated data with free labels configuring virtual sensors to the dense reconstructed scene models. We present the model-based dense reconstruction in Chapter IV and explore potential usages and applications of the reconstructed scene models.

1.3.4 Towards Realistic 3D Scene Models

The 3D mesh models used to perform the model fitting and reconstruction in Chapter III and Chapter IV are texture-less with no materials applied. Therefore, photo-realistic images cannot be rendered, and virtual sensors cannot simulate the color cameras. In Chapter V, we aim to discuss how to overcome this limitation. In particular, we explore one of the factors that can achieve photo-realistic reconstruction. We address the intrinsic image decomposition problem given the stream of images using a probabilistic framework. The experiment with 3D intrinsic models shows that scenes can be relighted by synthetic lighting effects and can be rendered into realistic images with proper reflectance and shading.

1.4 Contributions

The major contributions of this dissertation are as follows:

- A data capture system has been designed and built to capture motions of pedestrians and vehicles at unsignalized urban intersections. Two pairs of high-resolution stereo cameras and four spinning LiDAR sensors have been equipped onto the capture vehicle and configured to observe scenes from a driver's perspective. All the sensors are time-synchronized and are intrinsically and extrinsically calibrated. The collected data using the multi-modal sensor suite contains the natural behaviors of pedestrians and vehicles. (Chapter II)

- A publicly available multi-modal pedestrian dataset has been constructed with a rich set of 2D and 3D annotations. We presented a semi-automatic labeling process to obtain full 3D labels from 2D data. The approach was based on fitting the dense pedestrian mesh models parametrized in high-dimensional pose and shape parameters. The proposed 3D labeling method was validated using a MoCap system in a controlled outdoor environment simulating pedestrians in urban intersections. (Chapter III)

- We performed a model-based dense 3D reconstruction of the entire urban intersection areas. The scope of model-fitting has been extended to other object categories, including vehicles and static obstacles. Using the reconstructed synthetic 3D scene models, we have produced a rich set of semantic 2D and 3D annotations covering various objects in traffic scenes. Furthermore, we have generated simulated sensor data with free labels by configuring virtual sensors to the reconstructed scene model space. Throughout the experiments, which included simulating virtual sensors and predicting ego-vehicle trajectories, we discussed the validation of the simulated data and potential applications of the reconstructed scene models. (Chapter IV)

- To overcome the limitations of texture-less mesh models and achieve more realistic reconstruction, we tackled the intrinsic image decomposition problem for a stream of images using a probabilistic framework. The proposed intrinsic image prediction system contributed to the reduced amount of computation and running time to perform the image decomposition while maintaining the temporal consistency of the resulting decomposed images. The framework was integrated into a real-time Simultaneous Localization And Mapping (SLAM) system to reconstruct 3D intrinsic models of a scene. The experiments showed the possibility of rendering realistic images with varying lighting effects. (Chapter V)

1.5 Dissertation Organization

The rest of this dissertation is organized as follows:

- In Chapter II, we present our data acquisition system at complex urban intersections and describe the details of data collection.
- Chapter III discusses the 3D model-fitting-based labeling of pedestrians at real-world urban intersections.
- In Chapter IV, we present the dense 3D reconstruction of the intersections.
- Chapter V discusses the factors towards realistic 3D scene models.
- Finally, Chapter VI presents conclusions of this dissertation and discusses the future directions.

CHAPTER II

Data Acquisition in Complex Urban Intersections

2.1 Introduction

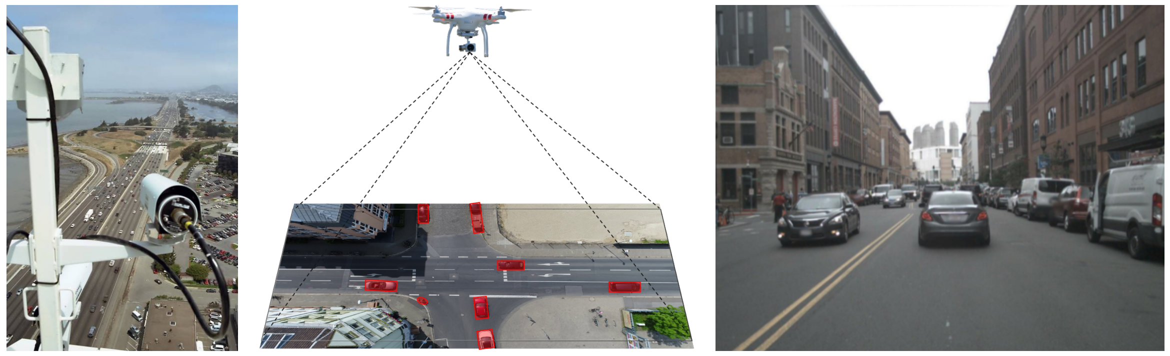
Autonomous driving has been one of the most popular topics in recent years. Many challenges related to AVs have been introduced and extensively studied, especially in robotics and computer vision communities. Despite rapid development of AV technologies, driving in urban environments has remained challenging mainly due to the complexity and the safety of pedestrians. For AVs to be fully automated and to safely operate in urban areas crowded with pedestrians, it is essential to understand the details of scenes and the attributes of dynamic agents.

One of the key prerequisites to understand the behaviors of road agents, particularly to apply data-driven algorithms, is to prepare large-scale real-world data. To develop data-driven perception algorithms, constructing a wide range dataset is critical. However, gathering meaningful data at urban traffic scenes is challenging in practice. To collect data, including road agents with complex patterns and interactions, we focused on unsignalized urban intersections. We aimed to collect data at urban traffic scenes specifically. In this chapter, we present the details of our data capturing system from scene selection to sensor setup and calibration.

2.2 Related Work

Before capturing data, many questions had to be answered:

- Where to install sensors?
- From which perspective to capture data?
- How to observe diverse behaviors of road agents?
- How to obtain labels?



(a) (b) (c)

Figure 2.1: Example ways to install sensors to capture data in traffic scenes. (a) NGSIM Interstate 80 Freeway Dataset [1]. (b) The Intersection Drone (inD) Dataset [2]. (c) nuScenes Dataset [3].

The sensors can be placed in many different ways, and accordingly data can be observed from different perspectives. For example, the sensors can be mounted onto a road fixture at height (Figure 2.1a), or the sensors can be carried by drones to capture bird's-eye view data (Figure 2.1b). The most common way is to install all the sensors onto moving vehicles. Cameras and LiDAR sensors are generally placed around the roof of vehicles, which capture perspective-view data frames. (Figure 2.1c)

Different perspective data has distinct features. For the data captured from a bird's-eye view, the sensors are not in the way of any road agents, so it is less likely to intervene with the road agents' trajectories. The sensors normally look down toward the entire scene, so the captured data can have less occlusion. Additionally, bird's-eye view data is relatively simpler to annotate than first-person view data because of the reduced dimensionality. The top-down bird's-eye views, however, are unrealistic for real-world mobile robot applications such as autonomous vehicles. Moreover, while bird's-eye view data can provide good quality trajectory data and High-Definition (HD) maps, it provides very limited modality data due to the reduced dimensionality. The data usually lacks variations in object appearance and cannot obtain high resolution data due to large distance. On the other hand, first-person view data is a more realistic form of data for mobile robots. Since less limitations exist on configuring different types of sensors, multi-modality data can be possibly captured. Also, details of scene components can be observed and captured such as appearance of pedestrians. However, the biggest challenge comes from incomplete observation of a scene caused by partial occlusions for the LiDAR points and limited view frustum of camera images. Also, it is likely that the data capturing affects to the road agents. The labeling process is costly as well, because of the complexity and diversity of scene appearance.

The data from a bird's-eye view is usually captured statically, which has advantages when observing diverse movement patterns of road agents. The data in a first-person view is generally obtained from a driving vehicle. In many existing urban traffic datasets, for instance, vehicles pass through an intersection while collecting data. While it cannot observe and record diverse agent patterns at a static spot, it can capture various backgrounds with different scene components while traveling.

With the growing significance and popularity of autonomous driving, many interesting datasets have been published. Many trajectory datasets are captured from bird's-eye views [2, 2, 15–18]. Most of recent autonomous driving datasets provide first-person view data with varying camera configurations [3, 19–24]. To overcome challenges of building large-scale real datasets, synthetic datasets from simulation have been released [25–28]. Various types of these datasets have expedited technology development in relevant fields.

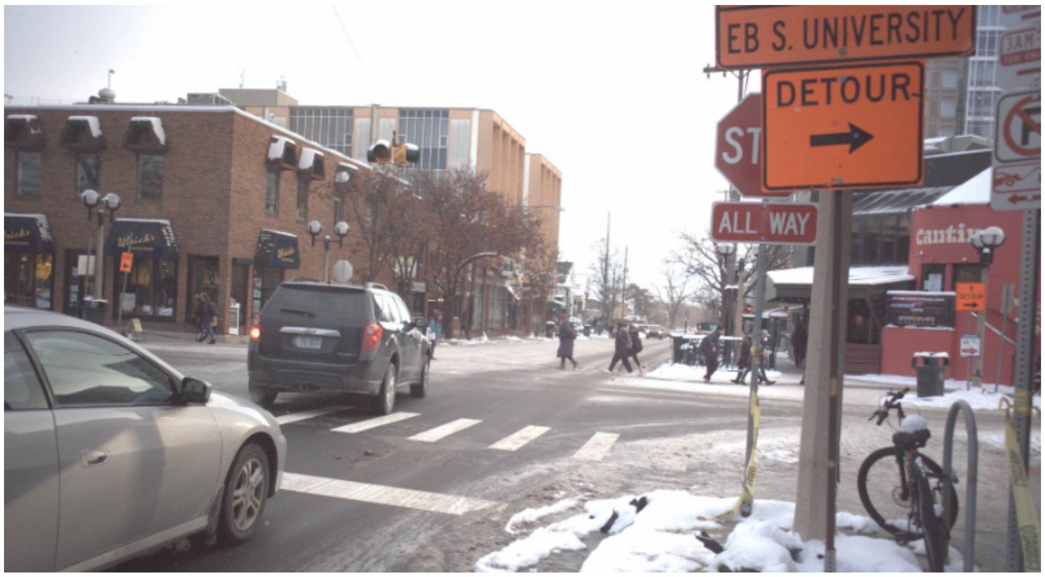
2.3 Scene Selection

We have considered several factors when deciding the capture location. The capture sites and times were selected to maximize the amount of traffic, complexity of crossing patterns, and lighting and weather variation. To capture complex interactions between pedestrians and vehicles, we focused on 4-way stop intersections without any traffic signals. At unsignalized intersections, agents make decisions by heavily relying on interactions with others. To pick the intersections, we also considered the specs of our sensors, such as measurement range of the LiDAR sensors and resolution and field-of-view of the cameras. For example, measurement range the measurement range of Velodyne HDL-32E LiDAR is upto 100 m. To get at least a few LiDAR points returned for each agent, we avoided selecting areas that were too wide. To reduce the intervention of our capture vehicle to the movement of other road agents, a parking spot was also required. Finally, the following unsignalized intersections were selected in downtown Ann Arbor, Michigan:

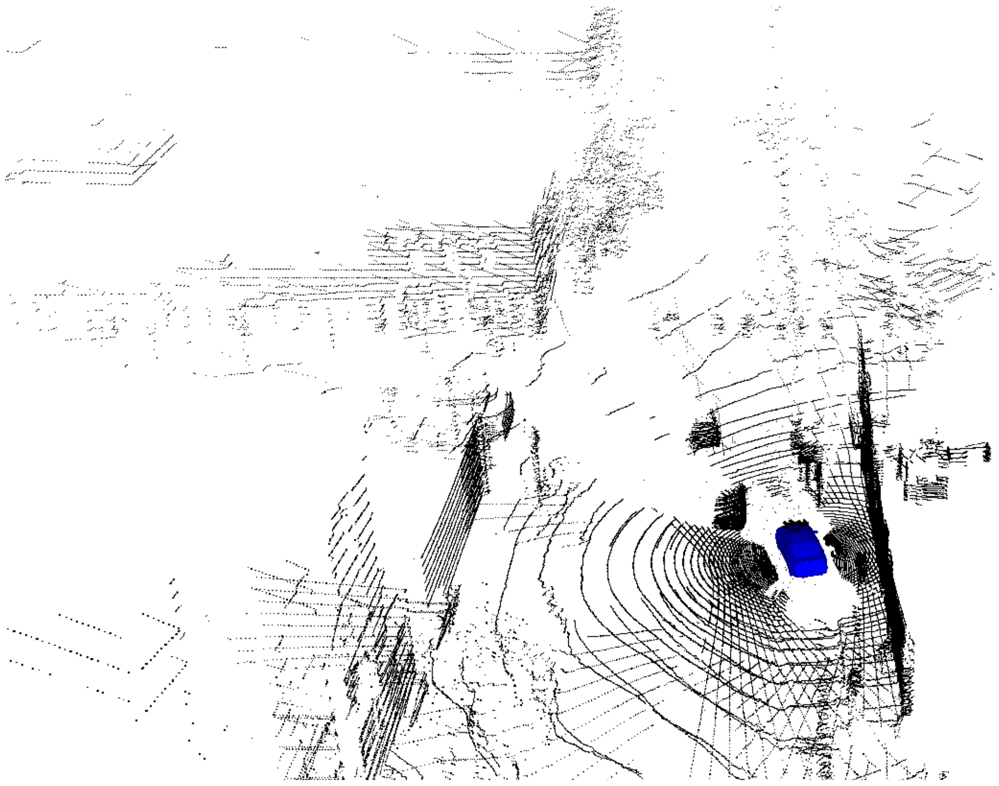
- E William St - Maynard St, Ann Arbor, MI
- S University Ave - Church St, Ann Arbor, MI
- Catherine St - N 4th Ave, Ann Arbor, MI

where the pedestrian-camera distance ranges between 5–45 m. We picked the busiest times of the day in the afternoon when many pedestrians and drivers pass through the intersections to commute. Lighting conditions varied based on cloud cover and shadows cast by the buildings.

Our vehicle was parked at one side of the intersection while capturing the data. Cameras were oriented toward different areas of each intersection to cover the entire region. The cameras captured perspective view images, which can simulate drivers' perspective in real driving scenarios. Figure 2.2 shows an example image and the point cloud captured at one of the intersections.



(a)



(b)

Figure 2.2: An example image and point cloud captured at the S University Ave - Church St intersection. Our capture vehicle is shown as a blue 3D model in the point cloud.

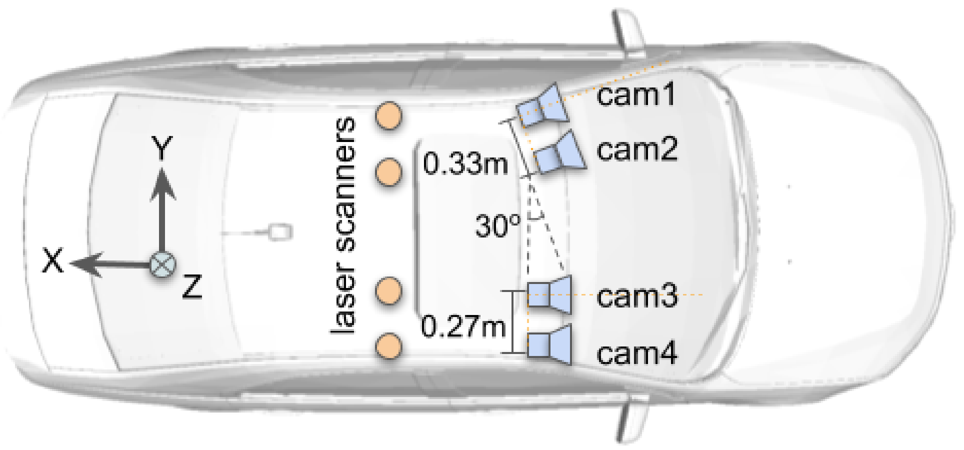


Figure 2.3: A diagram showing the sensor setup of our capturing vehicle. The global coordinate frame is defined at the origin of the INS system. The LiDAR returns from the individual LiDAR sensors are transformed into this global frame and accumulated.

2.4 Sensor Setup

Our data capturing platform was equipped with the following sensors:

- 4 Laserscanners: Velodyne HDL-32E
- 4 Color cameras, 12 Megapixels: Allied Vision Manta G-1236C
- 4 Lenses, 12 mm: V1228-MPY

The vehicle was equipped with four spinning LiDAR scanners (600 rpm, 32 laser beams, $700,000$ points per second, range: 100 m), two at each side of the roof with a roll angle of 45° between them. The cameras (6Hz, resolution: 4112×3008 pixels, opening: $90° \times 60°$) were mounted on top of our vehicle in two stereo pairs. The left pair was mounted on an independent bar rotated by 30° to capture the incoming road from the left and the right pair facing directly forward. We arranged the cameras with a baseline of 0.33 m and 0.27 m for the left and right stereo pairs, respectively.

We triggered the cameras via a trigger signal emitted when the second camera from the left started exposing its sensor. We recorded the timestamp of each image using the cameras' internal clock and these clocks were synchronized via the IEEE1588-2008 PTP protocol. We also synchronized the computer timestamp to the camera clocks using the same method. Using this timestamp, we computed LiDAR returns within a given camera frame. Figure 2.3 and Figure 2.4 show our sensor setup.

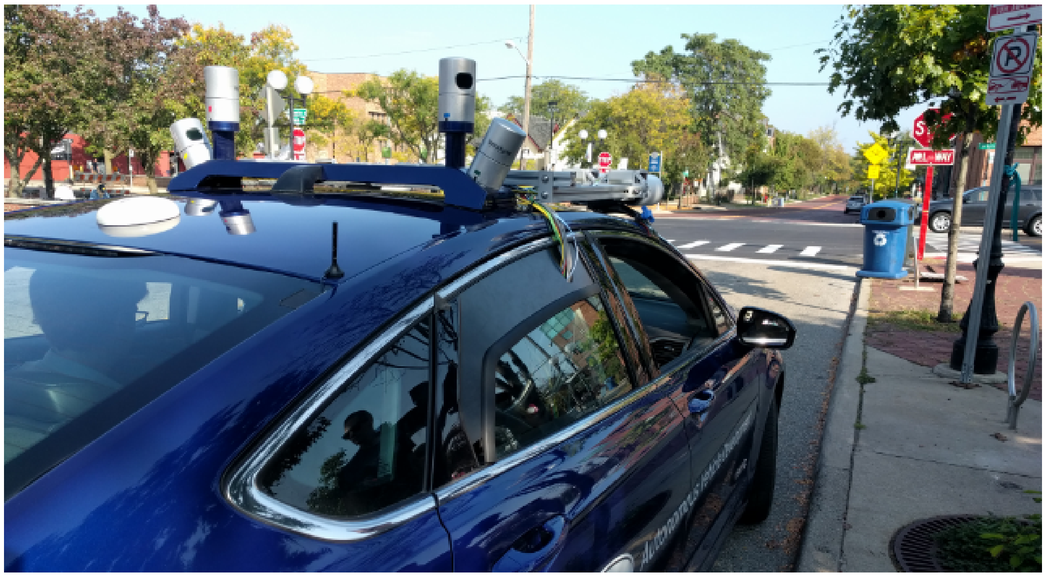


Figure 2.4: A real picture of our Ford Fusion vehicle with sensors installed.

2.5 Sensor Calibration

To be able to transform all the captured data into a unified coordinate frame, we performed sensor calibrations. We defined the global coordinate frame using the axes depicted in Figure 2.3 where the origin is defined to align with the Inertial Navigation System (INS) system. The accumulated LiDAR returns are already defined in this global coordinate frame. The calibrations were performed following the fundamental principles of multiview geometry [29] and using the tools provided by publicly available libraries [30, 31].

2.5.1 Camera Calibration

To use the cameras on the camera rig, we first calibrated intrinsic parameters of each camera and performed the stereo camera calibration for each pair of stereo cameras to find the SE3 pose of a right camera relative to a left camera. Although we tried to make left and right cameras parallel when installing the cameras onto the rig, they are not perfectly parallel. To make two cameras of the same stereo pair share the same image plane, we additionally computed the transformation onto the rectified image plane as part of the results.

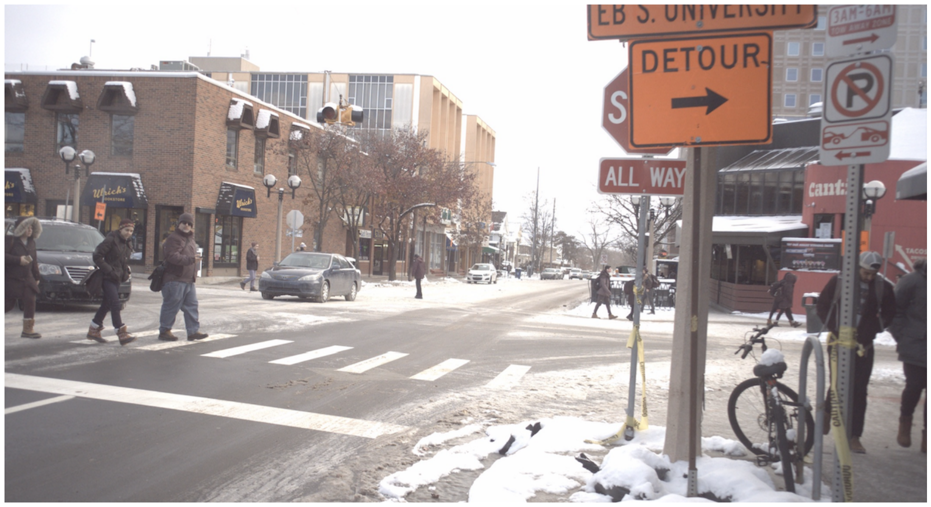


Figure 2.5: An example image captured at an intersection containing many pedestrians and vehicles passing through the intersection.

2.5.2 Camera-LiDAR Calibration

For determining the camera to LiDAR calibration, we used 50 manually selected constraints between 3D LiDAR points and 2D pixel locations in both the left and right images and used the Levenberg-Marquardt algorithm to minimize projection error. The geometry of the sensor setup was used to compute the initial guess for the rigid body transformation. Prominent edges, such as building corners, stop sign poles, and electric poles can be easily identified in the point cloud as well as the image which are used for manually choosing the correspondences.

2.5.3 MoCap-LiDAR Calibration

We used a weighted Iterative Closest Point (ICP) algorithm to determine the rigid body transformation between the MoCap coordinate frame and our world coordinate frame which is defined with respect to the capture vehicle. In further detail, we placed a large planar board with eight LED markers in the capture volume. The LED marker locations are linearly interpolated to sample more points from the planar board. The corresponding point clouds of the LiDAR returns from the planar board were manually segmented. We placed the board in multiple positions to accurately determine the rigid body transformation. Finally, the time-synchronization between the MoCap frames and camera frames were manually determined as well.

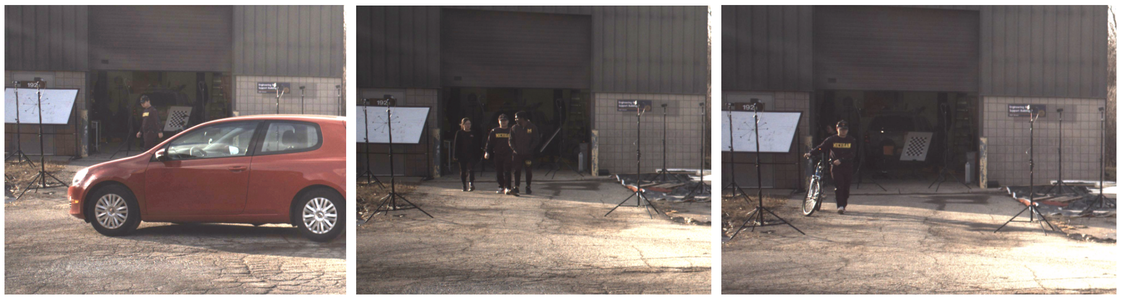


Figure 2.6: Example images with MoCap setup from our evaluation set.

2.6 Raw Image Processing

The raw Bayer 12-bit images were converted into compressed PNG/JPEG image formats. We compressed the raw images into 16-bit PNG files to keep the high dynamic ranges (HDR). To apply any detectors to the images, we further compressed the HDR images into 8-bit images using Durand's tone mapping algorithm [32], and rectified all the images for each stereo pair.

2.7 Conclusion

In this chapter, we built a data capture system and described the steps to collect data at real urban intersections. The captured data consists of high-resolution ($\approx$ 12 MegaPixels) RGB images from two stereo pairs and time-synchronized point clouds accumulated from four spinning LiDAR sensors. Collected during the busy time of the day, the data includes pedestrians and vehicles communicating and interacting to make safe decisions at the selected intersections. To make this data more useful, however, semantic labels need to be annotated. In the upcoming chapters, we explore processing and labeling of the data.

CHAPTER III

3D Model Fitting to Pedestrians at Real-World Urban Intersections

3.1 Introduction

In computer vision, estimating human pose has been a long standing problem. The recent application of deep neural networks has generated state-of-the-art results for 2D body pose estimation [33], which has inspired extensions to the 3D pose estimation [34–38]. However, gathering ground truth 3D pose data is challenging. Motion Capture (MoCap) systems have been the primary generator of ground truth 3D pose data, but have restricted the variety and complexity of the 3D scenes that can be captured [39, 40]. For example, with mocap systems it is difficult to capture naturalistic in-the-wild scenes with groups of people who are moving and interacting. To overcome those technical limitations, this dissertation developed both a dataset and a ground truth generation approach to facilitate generating 3D poses on in-the-wild images without relying on MoCap.

Most AVs have cameras installed, so this data can serve as a primary source for human pose estimation using computer vision algorithms. In addition to cameras, LiDAR sensors have become an essential component for AVs due to its precise depth measurements. This motivated the importance of capturing both modalities for this benchmark set of complex urban intersections.

Our dataset has the three unique properties. First, the data were gathered outdoors with real challenges, such as varied lighting and weather conditions and the presence of occlusions. Second, the pedestrian data were collected at intersection length scales of up to a 45 m range, which is relevant for the deployment of pose estimation systems at application relevant scales. The captured scenes are also naturalistic. The pedestrians in our dataset are not actors, so they move and interact in a myriad of realistic ways. In addition, our dataset includes images capturing crowds of people. Lastly, we released multimodal pedestrian data, including high resolution images and point clouds, that are synchronously captured from stereo cameras and LiDAR sensors.

Annotations of our dataset also have distinctive features. All the annotated 3D pedestrians lie in a global metric-space coordinate frame as opposed to many existing datasets that operate in hip joint or camera center relative coordinate frames. We stress the importance of determining where a person is in the 3D world so one can plan actions around them. In addition, our multimodal data frames are captured in minutes long sequences with unique tracking IDs for each pedestrian, enabling temporal reasoning.

The contributions in this section are summarized as follows:

1. We release a publicly-available large-scale multimodal pedestrian dataset with a rich set of 2D and 3D annotations. The dataset captures the real world challenges that AVs are likely to face at urban intersections.

2. We present an automatic method to obtain full 3D labels from 2D data, enabling labeling of in-the-wild images without MoCap. Our proposed approach allows generating 3D data in a completely unsupervised manner using state of the art algorithms for 2D annotations.

3. Our automatic 3D labeling method is validated using a MoCap system in a controlled outdoor environment that simulates pedestrians in urban intersections.

We present this dataset to enable the study of 3D pose estimation while reasoning about pedestrian behavior around vehicles, particularly in crowded urban areas as depicted in Figure 3.1. We see this as one of the first areas where human pose estimation can have a tremendous impact on safety and intelligence of mobile robot systems. Understanding the pose of road users affords information about activity, attention, and predictions of future position, which are critical to safely navigate around humans.

3.2 Related Work

3.2.1 3D Human Pose Estimation

In many papers, 3D human pose estimation has been formulated as a problem of regressing 3D joint locations by directly extracting visual features from an image

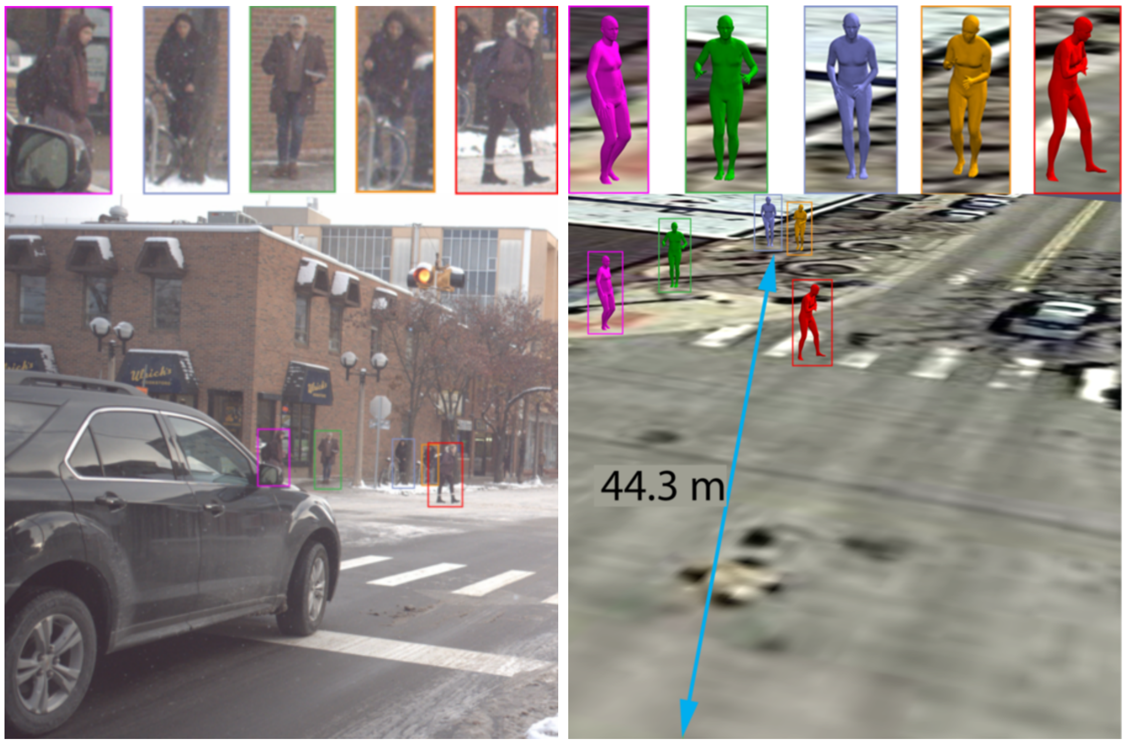


Figure 3.1: An example image from the PedX dataset with bounding boxes around the pedestrians (*left*) and a rendered image with the 3D human mesh models in metric space (*right*).

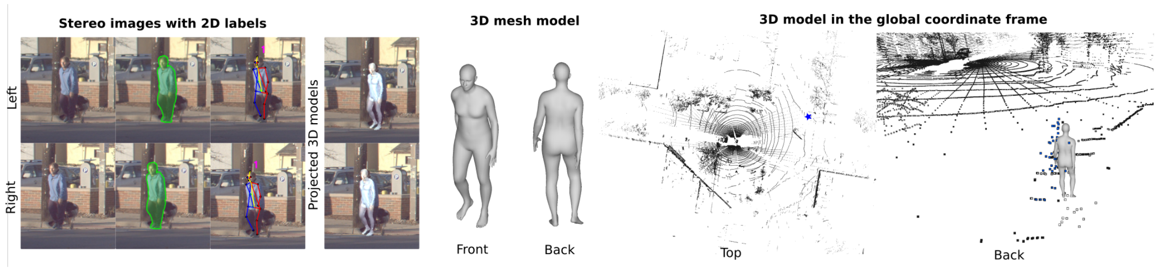

Stereo images with 2D labels
Left
Right
Projected 3D models
3D mesh model
Front
Back
3D model in the global coordinate frame
Top
Back

Figure 3.2: Visualization of our annotations. For each pedestrian, we provide 2D segmentation, 2D joint locations with visibility of 18 body joints, tracking ID, time-synced LiDAR points, and 3D mesh model localized into the global coordinate frame.

[34–37,41–44], or by lifting 2D joint detector outputs to 3D joints in a camera relative frame [38,45,46]. To reduce the inherent ambiguity of 3D human pose estimation from a 2D image, prior knowledge about feasible human poses has been considered [45,47], or a deformable 3D human model, such as Skinned Multi-Person Linear (SMPL) [48] was used to be fit to known 2D joint locations [46].

More recently, 3D poses have been estimated as part of a richer and denser mesh representations. SMPL model parameters are estimated from given dense 2D annotations [49], or the parameters are directly predicted from a single RGB image in an end-to-end manner using an adversarial learning framework [50]. Güler *et al.* [51] use a variant of SMPL parameterization by transforming it into part-based UV coordinates that specify a bijective mapping between mesh surface and an image. The proposed DensePose network assigns a body part to each pixel and regresses the corresponding surface coordinates. As a model-free approach, Rematas *et al.* [52] reconstruct 3D point clouds by estimating a per-pixel depth map for each soccer player from a monocular video using a neural network trained on synthetic data.

Most approaches take as input an image with a single person or a cropped image patch centered around a person and return 3D pose in a root-relative coordinate frame as the output where the camera is facing toward the person. While the joint estimation of 3D pose and virtual camera parameters have been proposed [37, 45], the outputs still suffer from the scale ambiguity. Without knowledge of exactly how far away a person is, controlling a mobile robot safely will be challenging. Most approaches are also unable to handle multi-person images with a single pass of an algorithm. While the multi-person 3D pose estimation problem has recently been addressed [51, 53], the output 3D poses are root-relative and still reliable up to a scale.

A major impediment to predicting metric space pose for multiple people in a scene is the lack of a suitable dataset with reliable 3D annotations. Addressing this limitation is the focus of this paper.

3.2.2 3D Human Pose Datasets

Large scale datasets have played an essential role in fueling recent progress in a variety of computer vision tasks. However, building a ground truth 3D human pose dataset in metric space is challenging since annotation in 3D is far more time consuming than the same task in 2D. To avoid manual labeling in 3D, MoCap systems are typically used to obtain the ground truth 3D human pose [39, 40, 54–56].

The data captured with traditional MoCap systems has many limitations. For instance, markers must be attached to subjects, which makes images look unnatural. Moreover, since MoCap is typically restricted to constrained, mostly indoor areas, image backgrounds for MoCap datasets have limited variability. The number of markers that can be tracked by MoCap systems is also limited, which limits the number of subjects. The recent development of a commercial, markerless MoCap system [57] allowed the data capture with natural-looking subjects [56], but it still suffers from the limited number of subjects and the size of capturing volume. Additional sensors,

Table 3.1: Statistics and characteristics of related datasets.

	num. images	num. instances	num. pixels	points from depth sensor	video	multi-person	real	scene type
H36M [40]	3.6M	0.9M	≈1M	✓	✓		✓	MoCap indoor
HumanEva [39]	≈80K	≈80K	1-1.3M	✓			✓	MoCap indoor
MPI-INF-3DHP [56]	>1.3M	≈93K	≈3M		✓		✓	MoCap in/outdoor
SURREAL [62]	6M	6M	≈80K		✓			indoor
UP-3D [49]	8.5K	8.5K	≈300K				✓	in/outdoor
DensePose-COCO [51]	34K	130K	≈300K	✓		✓	✓	in/outdoor
Ours	10,152	14,091	≈10M	✓	✓	✓	✓	outdoor

such as IMUs [58, 59], have been used to obtain ground truth 3D human poses of in-the-wild images, but the number of subjects per image is small and the data is limited to humans.

To counter the limited variability in subject appearance, camera viewpoints, lighting conditions, and image backgrounds, synthetic 3D datasets have been proposed [60, 61]. While there have been attempts to improve the photo-realism of these synthetic data generation pipelines [56, 62, 63], state-of-the-art synthetic images are still easy to distinguish from real images. Others have explored techniques to automatically do 3D labeling with limited human intervention [49, 64, 65]. For instance, some have taken advantage of a multiview camera setup to obtain reliable 3D annotations using optimization [64]. Others explore fitting parameterized mesh models [46] to monocular images or multiview images to collect ground truth 3D labels [49, 65]. However, these methods provide 3D scale models for a virtual camera with a relative frame and for images from various 2D pose datasets cropped around a single detection. This limits the potential utility of these methods while performing mobile robotic tasks safely. In contrast, we construct 3D human pose models in metric space for crowded urban street intersections that include as many as 15 pedestrians in a single image at distances as far as 45 m from the camera.

3.3 3D Model Fitting: Pedestrians

We performed 3D model fitting on pedestrians in a stereo-LiDAR sequence. In contrast to the previous work [46] that fits a 3D model to a single frame at a time, our approach optimizes over a sequence of stereo images and LiDAR points. We begin

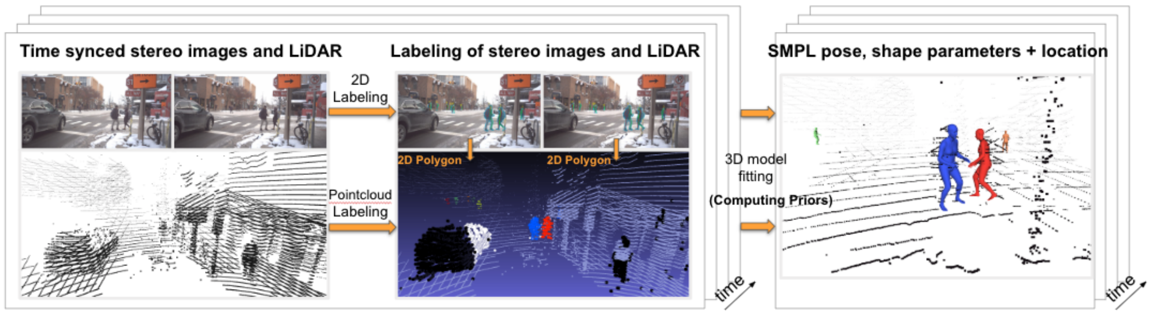

Time synced stereo images and LiDAR
2D Labeling
Pointcloud Labeling
Labeling of stereo images and LiDAR
2D Polygon
2D Polygon
time
3D model fitting
(Computing Priors)
SMPL pose, shape parameters + location
time

Figure 3.3: PedX labeling pipeline.

by per-instance model fitting, which is then extended to optimize over a sequence of instances using an iterative method. We also propose multi-modal and temporal priors. Note that we use a gender-neutral model for model fitting. Before initiating the model fitting pipeline, we preprocess the LiDAR data to identify regions containing potential pedestrians in 3D space. The points that are projected out of images or those that belong to the static objects are ignored. Using 2D segmentation labels for stereo images with known transformation between LiDAR and camera coordinate frames, we performed the point cloud labeling of each pedestrian instance.

3.3.1 Fitting To A Single Instance

We begin by performing 3D model fitting to a single instance at a single time step. For each pedestrian instance, we are given 2D joint locations $\boldsymbol{x}_l$ and $\boldsymbol{x}_r$, and 2D segmentations S_l and S_r for each stereo image. We also have sparse 3D points corresponding to the instance. To find the pose $\boldsymbol{\theta}$, shape $\boldsymbol{\beta}$, and 3D global position $\boldsymbol{t}$ that best fit to the instance, we formulate the problem as:

$$\underset{\boldsymbol{\theta},\boldsymbol{\beta},\boldsymbol{t}}{\text{minimize}}\, E_I\left(\boldsymbol{\theta},\boldsymbol{\beta},\boldsymbol{t}\right) \tag{3.1}$$

where $E_I = E_J + E_{3d} + E_P + E_T + E_D$ represents the sum of multiple energy terms. We verify the effectiveness of each energy term through ablative experiments described in Sec. 3.4.2. E_J is the sum of robust 2D reprojection errors [46] for both left and right images, E_P is the prior term, E_{3d} is the 3D Euclidean distance term between visible SMPL vertices and the LiDAR points, E_T is the translation term to constrain the 3D model location, and E_D is the heading direction term to constrain the body orientation:

$$E_T(t) = \|\boldsymbol{t} - \boldsymbol{t}_0\|_2^2 \tag{3.2}$$

$$E_D(\boldsymbol{\theta}) = \|f(\boldsymbol{\theta}) - \boldsymbol{d}\|_2^2 \tag{3.3}$$

$$E_{3d} = \tfrac{1}{N_v} \textstyle\sum_i^{N_v} \min_j \|\boldsymbol{X}_i - \boldsymbol{V}_j\|_2^2 \tag{3.4}$$

where $\boldsymbol{t}_0$ is the mean of 3D points, f is a function to convert the axis-angle representation of body orientation to xyz- directional vector, $\boldsymbol{d}$ is a known heading direction vector, $\boldsymbol{X}_i$ is the i-th LiDAR point, N_v is the total number of 3D points that belong to the instance, and $\boldsymbol{V_j}$ is the j-th point of SMPL model vertices.

3.3.2 Fitting To A Sequence Of Single Instances

To fit 3D models to a sequence of detections, we developed *shape* and *temporal consistency* constraints across frames in addition to the per-frame constraints.

3.3.2.1 Global Shape Consistency

Suppose one pedestrian appears in N consecutive frames with full 2D labels. In this instance, while pose parameters and translations change, the shape parameters should remain unchanged across the sequence. To find the pose, shape parameters, and translations, we formulate the problem as:

$$\underset{\boldsymbol{\theta},\boldsymbol{\beta},\boldsymbol{T}}{\text{minimize}}\; E_{seq}\left(\boldsymbol{\Theta}, \boldsymbol{\beta}, \boldsymbol{T}\right) \tag{3.5}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ and $\boldsymbol{T} = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_N\}$ are the set of pose parameters and the set of translations for all N frames. $\boldsymbol{\beta}$ is the shape parameters shared by all frames where $\boldsymbol{\beta} = \boldsymbol{\beta}_1 = \cdots = \boldsymbol{\beta}_N$. Optimizing over the entire sequence is challenging due to the high dimension of the decision variables. Since the objective E_{seq} is separable in terms of a per-frame objective, we decomposed this large problem into a set of smaller problems. We rewrote the unconstrained minimization problem over the entire sequence by introducing the consensus variable $\boldsymbol{\beta}$:

$$\begin{aligned} &\underset{\boldsymbol{\theta}_{1:N},\boldsymbol{\beta}_{1:N},\boldsymbol{t}_{1:N},\boldsymbol{\beta}}{\text{minimize}}\; \textstyle\sum_{k=1}^{N} E_{I,k}\left(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k, \boldsymbol{t}_k\right) \\ &\text{subject to}\;\; \boldsymbol{\beta}_k - \boldsymbol{\beta} = 0,\; k \in \{1, \ldots, N\} \end{aligned} \tag{3.6}$$

where k denotes the frame ranging from 1 to N frames in a sequence. This optimization is a constrained minimization problem with a separable objective function and multiple constraints requiring per-frame shape parameters $\boldsymbol{\beta}_k$ to be equal. The advantage of introducing a consensus variable $\boldsymbol{\beta}$ is that we can enforce all the frames to have common shape parameters while exploiting parallelism. We solve the problem by using the alternating direction method of multipliers (ADMM) [66]. The augmented Lagrangian to be minimized is:

$$\mathcal{L}_\rho\left(\boldsymbol{\beta}, \boldsymbol{\beta}_k; \boldsymbol{\Theta}, \boldsymbol{T}\right) = \sum_{k=1}^{N} E_{I,k}\left(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k, \boldsymbol{t}_k\right) + \boldsymbol{y}_k^T\left(\boldsymbol{\beta}_k - \boldsymbol{\beta}\right) + \frac{\rho}{2}\left\|\boldsymbol{\beta}_k - \boldsymbol{\beta}\right\|_2^2 \tag{3.7}$$

$\boldsymbol{y}_k$ is the dual variable for $\boldsymbol{\beta}_k$ and ρ is a positive constant that is experimentally selected. ρ=2 was used for the results reported in this chapter. The objective $\mathcal{L}_\rho$ is optimized using an alternating optimization for the local and global shape parameters with variables $\{\boldsymbol{u}_k\}_{k=1}^N$ where $\boldsymbol{u}_k = \frac{\boldsymbol{y}_k}{\rho}$. The updated equations from each iteration are as follows:

$$\boldsymbol{\beta}_k^{t+1} := \text{argmin}_{\boldsymbol{\beta}_k}\ E_{I,k}(\boldsymbol{\theta}_k, \boldsymbol{\beta}_k, \boldsymbol{t}_k) + \tfrac{\rho}{2}\left\|\boldsymbol{\beta}_k - \boldsymbol{\beta}^t + \boldsymbol{u}_k^t\right\|_2^2 \tag{3.8}$$

$$\boldsymbol{\beta}^{t+1} := \tfrac{1}{N}\textstyle\sum_{k=1}^{N}\left(\boldsymbol{\beta}_k^{t+1} + \boldsymbol{u}_k^t\right) \tag{3.9}$$

$$\boldsymbol{u}_k^{t+1} := \boldsymbol{u}_k^t + \boldsymbol{\beta}_k^{t+1} - \boldsymbol{\beta}^{t+1} \tag{3.10}$$

We perform the synchronous update for the global shape parameters. The iteration is stopped when $\|\boldsymbol{\beta}_k^{t+1} - \boldsymbol{\beta}^{t+1}\|_2 < 0.05$ and $\rho\|\boldsymbol{\beta}^t - \boldsymbol{\beta}^{t+1}\|_2 < 0.05$ or the maximum iteration is reached. (3.8) is similar to per-frame minimization in (3.1) with an additional term in the objective function.

3.3.2.2 Temporal Pose Prior

In addition to enforcing per-frame shape parameters to share the common values across the sequential frames, we penalized unlikely sequences of poses by using a temporal pose prior.

A 72-dimensional pose vector consists of the x, y, and z rotation angles of 23 body joints relative to each of their parent nodes, plus the orientation of the root hip in a 3D angle-axis representation. We observed that pose transitions between two consecutive frames are close to periodic for most body joints. In particular, since most pedestrians at the intersections involve in walking motion, there exist certain patterns for each body joint. Inspired by that, we define the pose transition vector and fit a Gaussian Mixture Model (GMM) with K multivariate distributions to the pose transition vectors of known walking sequences. We selected the sequences that were related to walking motion from the CMU mocap dataset [55].

$$\boldsymbol{x} = [\boldsymbol{t}; \boldsymbol{\theta}_{pose}] \tag{3.11}$$

$$\Delta\boldsymbol{x} = \boldsymbol{x}^t - \boldsymbol{x}^{t-1} \;\sim\; \textstyle\sum_i^K \phi_i \, \mathcal{N}\left(\Delta\boldsymbol{x}; \mu_i, \Sigma_i\right) \tag{3.12}$$

A GMM with K distributions can be written in terms of K sets of mean and covariance matrix, and the probability density function (pdf) of the Gaussian mixture distributions evaluated at $\Delta\boldsymbol{x}$ is as follows:

$$f(\Delta\boldsymbol{x}) = \textstyle\sum_{i=1}^K \phi_i \, \mathcal{N}(\Delta\boldsymbol{x}; \mu_i, \Sigma_i) \tag{3.13}$$

where $\sum_{i=1}^K \phi_i = 1$. If the GMM is correctly trained to represent the true pose transition vectors, the pose transition vector from the positive set (S_{pos}) will have a large probability density value while the one from the negative set (S_{neg}) will have a small probability density value. Note that maximizing this probability density can be rewritten as the following equivalent minimization problem so that the temporal pose prior can be easily incorporated into the objective function:

$$\underset{\Delta\boldsymbol{x}}{\text{maximize}}\, f(\Delta\boldsymbol{x}) \equiv \underset{\Delta\boldsymbol{x}}{\text{minimize}} \; -\log f(\Delta\boldsymbol{x}) \tag{3.14}$$

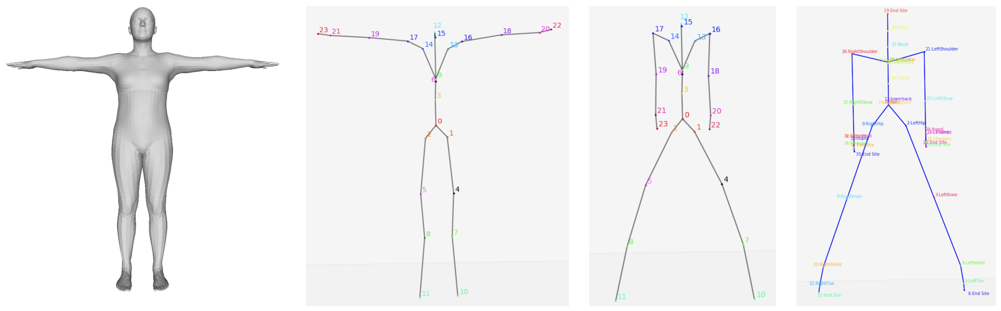
The negative log of the multivariate normal probability distribution is approximated as follows:

$$\begin{aligned} -\log f(\Delta\boldsymbol{x}) &\approx \max_{i\in\{1,\ldots,K\}} \; -\log\left(\phi_i \, \mathcal{N}(\Delta\boldsymbol{x}; \mu_i, \Sigma_i)\right) \\ &= \max_{i\in\{1,\ldots,K\}} \; -\log\phi_i + \frac{1}{2}(\Delta\boldsymbol{x} - \mu_i)^T \Sigma_i^{-1} (\Delta\boldsymbol{x} - \mu_i) + \log\left((2\pi)^K \left|\Sigma_i\right|\right)^{\frac{1}{2}} \end{aligned} \tag{3.15}$$

The final temporal pose prior that will be added to the objective function is:

$$E_{tp}(\boldsymbol{t}_k, \boldsymbol{\theta}_k; \boldsymbol{t}_{k\text{-}1}, \boldsymbol{\theta}_{k\text{-}1}) = -\log \textstyle\sum_i^K \phi_i \mathcal{N}\left(\Delta\boldsymbol{x}^t; \mu_i, \Sigma_i\right) \tag{3.16}$$

where μ_i, Σ_i are the mean and covariance of the pose difference vector Δx and ϕ_i is the weight for the i-th Gaussian mixture component. More details about extracting pose transition vectors from CMU mocap dataset is presented in the following section.



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

Figure 3.4: (a) Initial SMPL template model for the neutral gender where all the pose and shape parameters are set as zero. (b) Joints from the initial SMPL template model. (c) Transformed joints. (d) Joints from the CMU MoCap template in BVH conversions.

3.3.2.3 CMU MoCap To SMPL Conversion

To define pose transition vectors (3.12) from the CMU MoCap data sequences, we needed to convert the data into SMPL parameters. We first selected the motions from the CMU MoCap dataset that were related to the pedestrians, such as walking and running, and used the BVH conversions of the data [67].

72-dimensional SMPL pose parameters encode the information about how to deform the initial template model. For each joint, the transformation is represented as a three-dimensional vector using the axis-angle representation. Figure 3.4b shows 24 joints defined in the initial SMPL template model. In the BVH conversions of the CMU MoCap dataset, however, the initial pose of the skeleton is defined in a different way (Figure 3.4d). We first transformed the initial SMPL joints similar to the BVH template. Figure 3.4c shows the transformed SMPL joints. Starting from this transformed template model (Figure 3.4c), we converted BVH data into 72-dimensional SMPL parameter vectors.

In both BVH format and the SMPL parametrization, body joints are represented as a tree structure with the hip as a root. For the BVH data, three-dimensional Euler angles are stored per joint from which a 3×3 rotation matrix relative to its parent joint can be computed. The tree defined for the SMPL parameters has similar structure to the one of BVH data but with a fewer amount of nodes. For each joint, a transformation relative to its parent can be computed using a three-dimensional parameter vector in an axis-angle representation. For each of the 24 SMPL joints, we obtained a rotation matrix relative to its parent, which can be converted into a three-dimensional SMPL parameters in an axis-angle representation using Rodrigues'

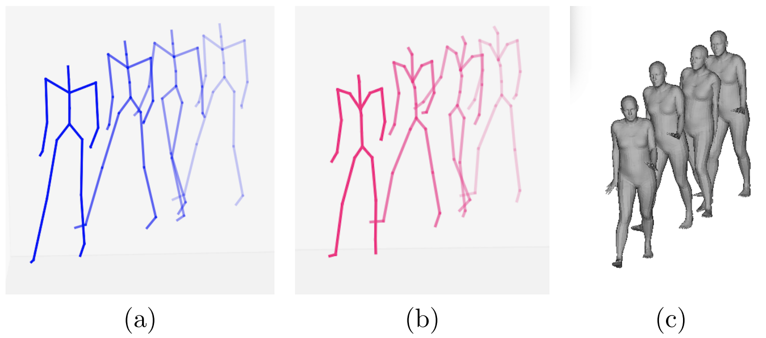


Figure 3.5: Example results of converting CMU MoCap BVH data into SMPL pose parameters. (a) CMU MoCap skeletons in BVH format data. (b) Skeletons obtained from the corresponding SMPL parameters. (c) SMPL mesh visualization with zero shape parameters.

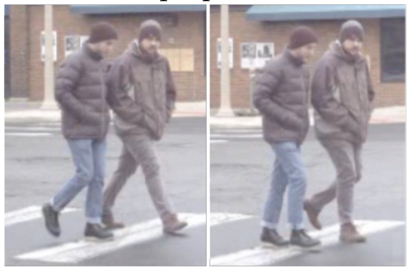


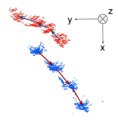
Figure 3.6: Two consecutive frames from PedX dataset.

rotation formula [68]. Figure 3.5 shows example results of converting BVH data into SMPL parameters. For the 3D mesh visualization in the figure, we used zero shape parameters.

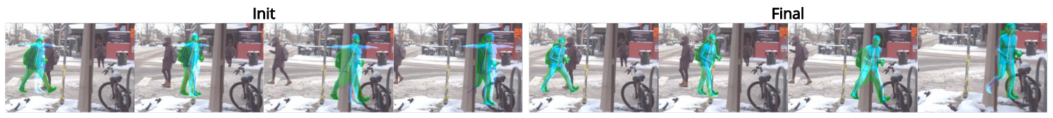
MoSh can be another way to obtain the SMPL parameters corresponding to the CMU MoCap sequences. Since we are interested in the difference between two neighboring pose parameter vectors, we performed a simple conversion of poses without considering the shape parameters.

After converting CMU MoCap sequences into SMPL parameter vectors, we generated samples to train the GMM for temporal pose prior. Note that the frequency of CMU MoCap sequences are $120Hz$ while our camera frequency was $6Hz$. We selected two neighboring CMU MoCap frames which have 20 frames in between to compute the pose difference vector $\Delta x = (\Delta t, \Delta\theta)$.

Regarding the temporal prior, we considered using low-D DCT basis vectors [65] that do not require a training process. They use the Human3.6M dataset [40] where the data was captured with $50Hz$ frame rate. On the other hand, the frame rate of our dataset is lower than that, and the pedestrian's pose can have more changes between two consecutive frames. (See Figure 3.6.)



(a)



(b)

Figure 3.7: (a) Overhead view of the point cloud trajectories of pedestrians in the global frame. (b) Initialized body orientation based on heading direction, and the final 3D models after the iterations.

3.3.3 Initialization

One of the challenges while fitting a 3D model is estimating a body orientation with the incorrect sign [65]. To avoid getting a flipped model, we computed the heading direction of each pedestrian from a sequence of LiDAR points and use it to initialize the body orientation via a three-dimensional angle-axis representation. We assumed that pedestrians never walk backwards, which held true in our data capture.

Figure 3.7a shows some example trajectories from LiDAR points, and Figure 3.7b shows the projected 3D model onto the left images at four sequential time frames after the initialization. When a sequence is only a single frame, we found the initial body orientation with the template pose by minimizing the stereo reprojection error E_J only using torso body joints and the translation error E_T around the mean LiDAR points.

3.4 Experiments

The data in our dataset is captured at complex outdoor urban intersections where pedestrian-to-camera distances are large (5–45 m), with multiple subjects who are often heavily occluded. In contrast, publicly available 3D datasets rely on MoCap systems [39, 40, 54] and guarantee less than a few millimeters accuracy for a single subject appearing within a controlled indoor capture volume of a few meters in radius. Given these distinctions, the accuracy of our dataset needs to be evaluated under these realistic conditions. While comparing the accuracy of our proposed approach with a MoCap system would reliably validate our dataset, MoCap systems cannot be practically setup at urban intersections.

We address this challenge by leveraging the fact that our proposed approach only requires 2D labels and LiDAR data without using image features. The key factors affecting our approach include the density of the LiDAR returns due to the large distance from the capture vehicle, occlusions by other objects that affect the LiDAR segmentation, the precision of the manual annotation when the pedestrian occupies a small portion of the image and the calibration between LiDAR and camera. The lighting conditions, background appearance, clothing and weather conditions do not affect our approach as we operate on manually labeled joint locations. Therefore, we collect and annotate an evaluation dataset in a controlled outdoor environment with a MoCap system and get manual 2D labels while replicating the vehicle to target distances and the clutter and occlusion of the intersection data. By showing that the 3D labels generated by our method are comparable to the MoCap ground truth, we verify the fitting approach and the annotation process against a traditional MoCap source.

3.4.1 Data Verification

3.4.1.1 Evaluation Dataset

We use the PhaseSpace MoCap system with active LED markers. which can be used in outdoor environments. The subject wears a suit with markers placed around body parts and repeats actions such as walking, jogging, and waving that are common for pedestrians. The capture vehicle was parked about 20 m away from the MoCap setup. To replicate typical occlusions, we parked another car between the capture vehicle and the MoCap setup as well as having groups of pedestrians walking. We selected 626 frames and obtained manual 2D labels for the images. Since the visual appearance does not affect the evaluation, we restrict the evaluation to a single subject with a single background and focus more on variation in poses and occlusions.

3.4.1.2 Evaluation metric

The 3D Mean Per Joint Position Error (MPJPE) is a standard metric to evaluate pose estimation algorithms, which is a mean over all joints of the Euclidean distance between ground truth and prediction. In cases where the prediction is not in metric space, the error is computed for a root-relative coordinate frame after allowing a similarity transform to register the prediction to the ground truth. In cases where the prediction is in metric space, we computed MPJPE in global coordinate frame without any registration. We reported the per joint position errors for both frames. Note that, given the geometry of capture, markers can get completely occluded from the MoCap system. Consequently, not all joints are visible in all frames. Moreover, some methods may not predict invisible joints. Therefore, we take the weighted mean while computing the MPJPE where the weight is equal to the number of frames in

Table 3.2: 3D MPJPE in root-relative coordinate frames.

Relative (mm)	rknee	lknee	rankl	lankl	rsho	lsho	relb	lelb	rwri	lwri	head	neck	hip	mean
[35]	113	153	203	184	141	120	130	132	203	194	134	96	88	147
[38]	107	116	172	159	159	160	142	127	197	178	84	88	87	137
[46]	**103**	89	139	158	77	59	**74**	71	144	145	85	35	**87**	97
Ours	104	**71**	**126**	**136**	**62**	**50**	83	**67**	**118**	**130**	**66**	**29**	106	**88**

Table 3.3: 3D MPJPE in global coordinate frames.

Global (mm)	rknee	lknee	rankl	lankl	rsho	lsho	relb	lelb	rwri	lwri	head	neck	hip	mean
Triangulation	1178	1307	1617	1489	1205	1164	1130	1168	1206	1111	842	1029	1111	1194
Left+disp	766	977	1229	1018	702	648	756	766	825	789	482	437	972	794
[46]+disp	535	576	627	591	620	574	561	570	652	594	588	587	598	593
Ours	**205**	**179**	**250**	**255**	**183**	**177**	**187**	**182**	**221**	**204**	**169**	**155**	**161**	**194**

which the joint was visible in the ground truth and was predicted.

3.4.1.3 Baseline Methods

We consider three different families of baseline methods. First, we consider a method that predicts 3D joint coordinates (up to scale) directly from 2D images [35]. Second, we consider methods that take manual 2D joint annotations as inputs [38,46]. We evaluate these methods in the root relative frame alone. Third, we consider three naive baselines that use stereo geometry information. As we have 2D joint locations for a calibrated rectified stereo pair of images, we directly triangulate these 2D joint locations for visible joints. We refer to this method as *Triangulation*. For the second naive baseline, we use disparity values and 2D joint locations in the left image for the visible joints and the previous triangulation result for invisible joints. We refer to this as *Left+disp*. Finally, we consider a baseline that modifies an existing technique [46], this approach uses the calibrated camera parameters and the estimated skeletons, which are scaled to metric space by using the average disparity values at the visible joint locations. We refer to this as *SMPLify [46]+disp*.

3.4.1.4 Accuracy

Table 3.2 and Table 3.3 summarize the results for root-relative and global frames respectively. Although fair comparisons can only be done between methods separated by horizontal lines, the objective of these tables is to highlight that the current state-of-the-art still has room for improvement and the utility of the proposed dataset in closing this gap. The proposed approach achieves lower MPJPE for the majority of joints in the root-relative frame. This is expected as our approach leverages additional information, including LiDAR returns, temporal priors as well as stereo annotations. The gains while using this additional data are most prominent in the global coordinate frame as no registration is involved and consequently global translation, orientation and scale errors can be seen in the global MPJPE. While naive baselines such as triangulation and left+disp perform poorly because they do not leverage any prior about proportions of a typical human skeleton, SMPLify+disp which leverages priors about human skeletons still suffers from large errors. In contrast, our proposed approach achieves an MPJPE of 194 mm for an average camera-to-pedestrian distance of 20×10^3 mm.

3.4.2 Ablation Study

Table 3.4 summarizes the results for using different subsets of energy terms in the optimization. Note that $E_{J,l}$ and $E_{J,r}$ represents the reprojection error on left and right images. E_T, E_{3D} and E_{tp} are defined in Sec. 3.3. Each column shows per-joint errors in root-relative frame except the last column, which shows the MPJPE in global frame.

3.4.2.1 Effect of Stereo

To see how using a stereo reprojection error affects the resulting 3D models, we computed reprojection errors for only the left images (i.e., row 4 of Table 3.4) and for both stereo images (i.e., row 5 of Table 3.4). Stereo imagery reduced both global translation error and root-relative pose error as it reduces the depth ambiguities that exist for the monocular approach. The second row of Figure 3.9 shows the results from monocular approach, which estimates 3D models from the left images. Notice that in several frames, the legs are swapped or the body orientation is estimated incorrectly. Figure 3.8 presents additional results with occlusions, and illustrates similar limitations of the existing approach. We can see that the projection of the estimated 3D models onto the right image does not align exactly. Estimating 3D pose from 2D joints is an inherently ill-posed problem because many feasible body configurations exist. Using stereo information provides more constraints during the optimization and reduces such ambiguities overall. Furthermore, when some joints are occluded in a single image, when using stereo pairs, the second image of the pair may observe these joints, which can reduce the uncertainty and produce better 3D models.

3.4.2.2 Effect of Using LiDAR Points

Although stereo images provide reasonable depth estimation in a global coordinate frame, the translation error may be too large since an error of a few pixels during labeling may create a large resulting error in fit. To place 3D models at the correct location in metric space, we included LiDAR information as a form of the translation prior E_T. The translation prior term localizes the 3D models at the distance observed by the LiDAR. As shown in the first two rows in Table 3.4, adding this translation prior provides an improvement in estimating global 3D pose.

The translation prior only constrains the location of the root joint (hip) in the 3D metric space. Therefore, depth ambiguities in other parts of the body may still exist, especially when the subject appears sideways with respect to the camera. Adding the 3D distance term E_{3D} helps to adjust the pose or body orientation to fit to the observed LiDAR points. Consequently the mean column in the root-relative frame significantly reduces from row 2 to row 4.

3.4.2.3 Temporal Prior

The temporal prior term, E_{tp}, penalizes unlikely transitions of poses and translations between consecutive frames. It also makes the resulting 3D pose between consecutive frames appear smooth. In Table 3.4, row 2 and row 3, and row 4 and row 5 show the errors without and with the temporal prior, respectively. By adding this term, we obtain similar values for the root-relative errors, and achieve lower global error. Another advantage of the temporal prior is that it makes the model robust to 2D labeling noise for the occluded joints. Figure 3.8 illustrates examples with severe occlusions. It is likely that the 2D body joint labels are inconsistent across the frames under severe occlusion, and that affects the estimated pose. As seen in rows 4 and 6 of Figure 3.8, the resulting 3D poses are smoother when this temporal prior term is included. However, if the weight of the temporal prior is set too high, the transition of poses between the frames can become too restricted.

Table 3.4: Ablation study on energy terms. Each column shows per-joint errors in mm in root-relative frames. The last column shows the MPJPE in global frame.

	$E_{J,l}$	$E_{J,r}$	E_T	E_{3D}	E_{tp}	rknee	lknee	rankl	lankl	rsho	lsho	relb	lelb	rwri	lwri	head	neck	hip	mean	global
1	✓	✓				100	79	128	145	66	54	85	66	147	165	72	38	134	98	1050
2	✓	✓	✓			98	76	**124**	143	68	55	84	66	141	157	73	37	121	95	277
3	✓	✓	✓		✓	117	76	126	138	64	50	81	**64**	123	137	68	34	116	91	248
4	✓	✓	✓	✓		104	72	125	**135**	**62**	**49**	83	67	122	134	**65**	**29**	107	**88**	250
5	✓		✓	✓	✓	**89**	84	132	146	68	57	**74**	65	143	158	80	38	122	97	240
6	✓	✓	✓	✓	✓	104	**71**	126	136	63	50	83	67	**118**	**130**	66	30	**106**	**88**	**194**

Table 3.5: Comparison of 3D MPJPE in root-relative coordinate frame for automatic and manual 2D labels. The last column is the MPJPE (mm) in global coordinate frame.

2D labels	rknee	lknee	rankl	lankl	rsho	lsho	relb	lelb	rwri	lwri	head	neck	hip	mean	global
[33]+ [69]	**104**	89	143	156	81	61	89	75	143	134	112	45	**59**	99	224
GT 2D joints	**104**	**71**	**126**	**136**	**62**	**50**	**83**	**67**	**118**	**130**	**66**	**29**	106	**88**	**194**

3.4.2.4 Global Shape Consistency

Figure 3.9 compares the results from SMPLify [46] with those from our method, which enforces consistency of the global shape parameters across multiple frames. As expected, using the global shape consistency constraint produces more consistent shapes across the frames, while the resulting 3D models from SMPLify, which only uses per-frame information, looks inconsistent. Moreover, the models are too skinny especially when the desired pose is far from the template pose.

3.4.3 Effect of Noisy 2D Labels

The size of manually labeled datasets is always limited by the time-consuming and costly annotation process. Luckily, recent methods have made great progress in 2D visual recognition tasks. Using pre-trained networks [33, 69], we computed instance-level segmentation masks and 2D joint detections on our evaluation dataset in place of manual labels. Since only a single subject exists in a scene wearing the MoCap suit, the tracking ID is trivially obtained.

Table 3.5 shows that the 3D MPJPE error for using the 2D estimates is larger than that for using manual labels. For the hip joint, manual labels by annotators can be less consistent across frames than the 2D algorithm outputs, which might result in a large error. Although using manual labels produced higher accuracy than that of using 2D estimates, the method still achieved greater accuracy than ad-hoc versions of monocular methods shown in Table 3.3. This illustrates that our proposed approach is robust to noisy 2D labels and could potentially be scaled for larger datasets using state-of-the-art 2D visual recognition algorithms.

3.4.4 Qualitative Results

Figure 3.10 shows some representative examples from *PedX* that illustrate the uniqueness and variety of our dataset. Our dataset covers various actions and poses that are frequently encountered at intersections. Examples include walking, jogging, waving, using a phone, cycling, carrying objects, and talking. Occlusions are another challenge in estimating 3D pose. Our dataset contains many pedestrian instances where they are severely occluded by surrounding objects or by other pedestrians. In addition, most frames of the dataset contain more than one pedestrian at a time. Our dataset also contains different weather conditions and rare occurrences, such as people in wheelchairs or pedestrians jaywalking.

3.4.5 Runtime and Sequence Length

Our 3D model fitting approach involves the high-dimensional parameter optimization. The runtime of fitting 3D models per sequence goes up as the number of frames per sequence increases. The number of frames per sequence of a pedestrian vary, and it is more than a hundred frames for most sequences. To keep the runtime feasible, we use up to ten neighboring frames per sequence. It takes less than five minutes to process a single sequence.

3.5 Conclusion

This chapter presented a novel large-scale multimodal dataset of pedestrians at complex urban intersections with a rich set of 2D/3D annotations. The *PedX* dataset provides a platform for understanding pedestrian behaviors at intersections with real-life challenges. This dataset can be used to solve 3D human pose estimation, pedestrian detection, and tracking in the wild, as well as used to further expand the research on other problems.

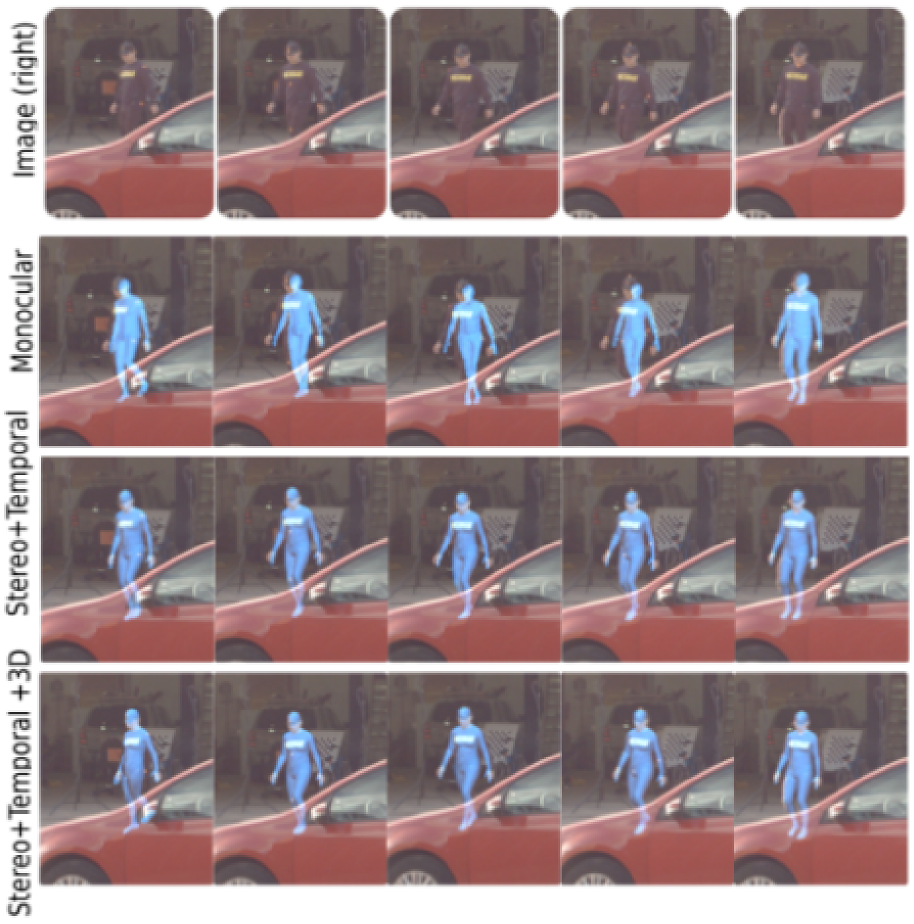


Figure 3.8: Results from using different cost terms. The resulting 3D models in the temporal walking sequence were projected and overlaid onto the images.

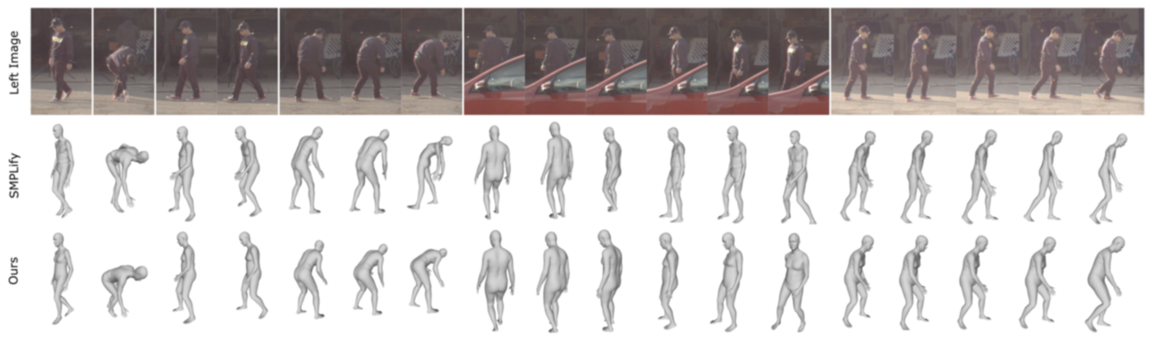


Figure 3.9: Results from monocular SMPLify and from our method.

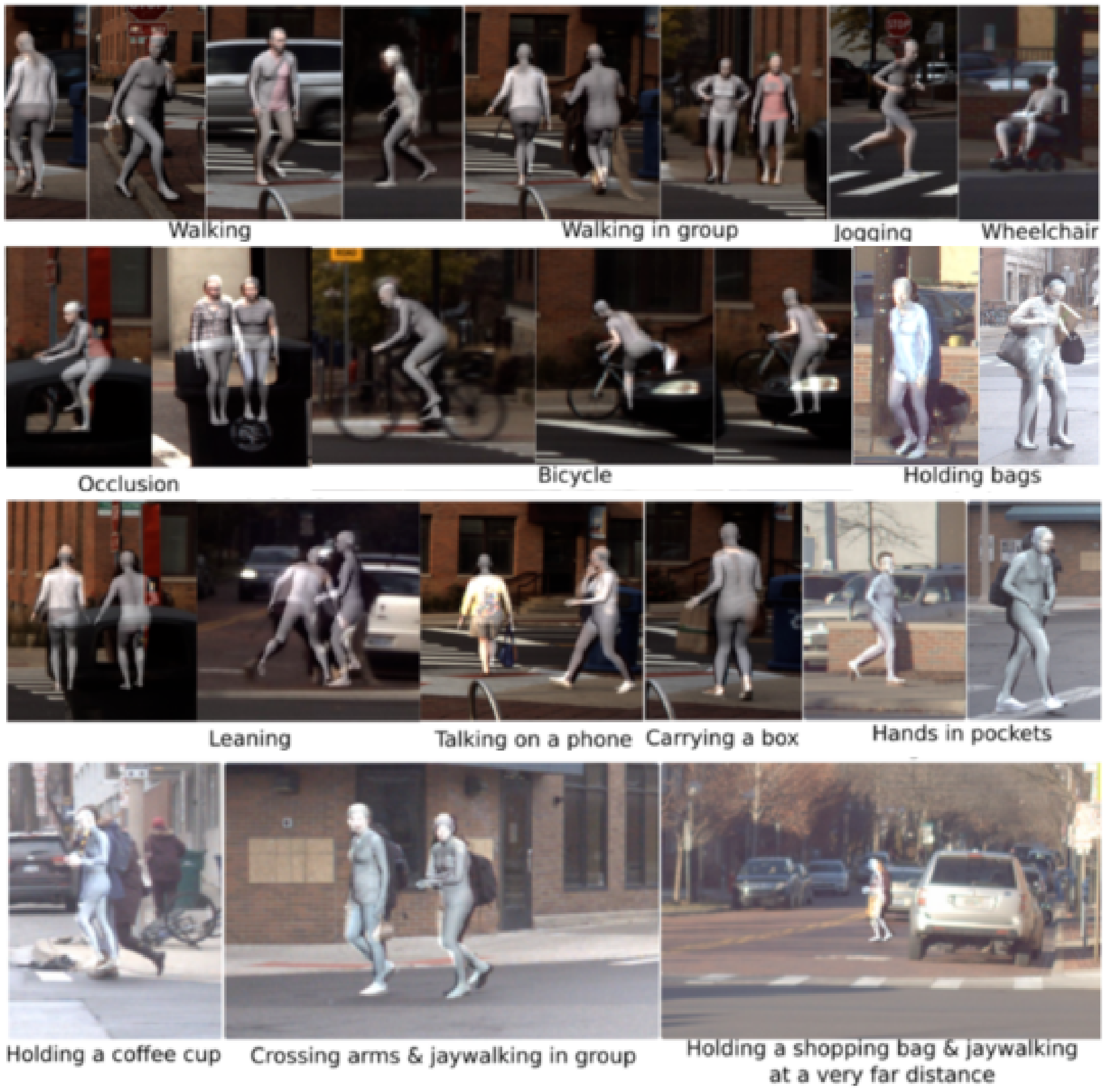


Figure 3.10: Representative samples from our dataset. 3D models are rendered onto the images.

CHAPTER IV

3D Model Fitting to the Real-World Urban Intersections

4.1 Introduction

Despite the rapid development of AV technologies in recent years, urban intersections have remained challenging for AVs to be fully automated mainly because of the large complexity of urban intersection scenes, which results in a large uncertainty in the motions of road agents. For further improvements in many data-driven perception algorithms or testing beds of AV simulations, constructing a wide range of datasets is critical; however, capturing multimodal data at urban intersections is challenging in practice. In many urban traffic datasets [3, 20–24], for instance, vehicles pass through an intersection while gathering data. Due to the occlusions and limited view frustum of cameras, the vehicles can only capture data of the partial scene. Moreover, a limited variety of agent behaviors can be collected during a limited duration of time. Using bird's-eye view sensors attached to drones is one of the workarounds to capture complete intersection data with reduced occlusions and little influence on road agents [2, 70–73]. However, view frustum and the camera perspective are far apart from real AVs, so such data might not be appropriate to simulate actual AV perception sensors.

The *PedX* dataset [74] provides multimodal data captured at urban intersections in an AV's perspective. The capture vehicle was static while collecting data; thus, the sensors observed diverse behaviors of road agents. In this paper, we extend the pedestrian-only dataset to include full-scene data. We expand the scope of automatic labeling to broader scenes, including moving vehicles and static components such as lanes, buildings, and traffic signs. Given prior knowledge about the map of a static scene and 3D template models of the rigid objects, we aimed to fit 3D models to all

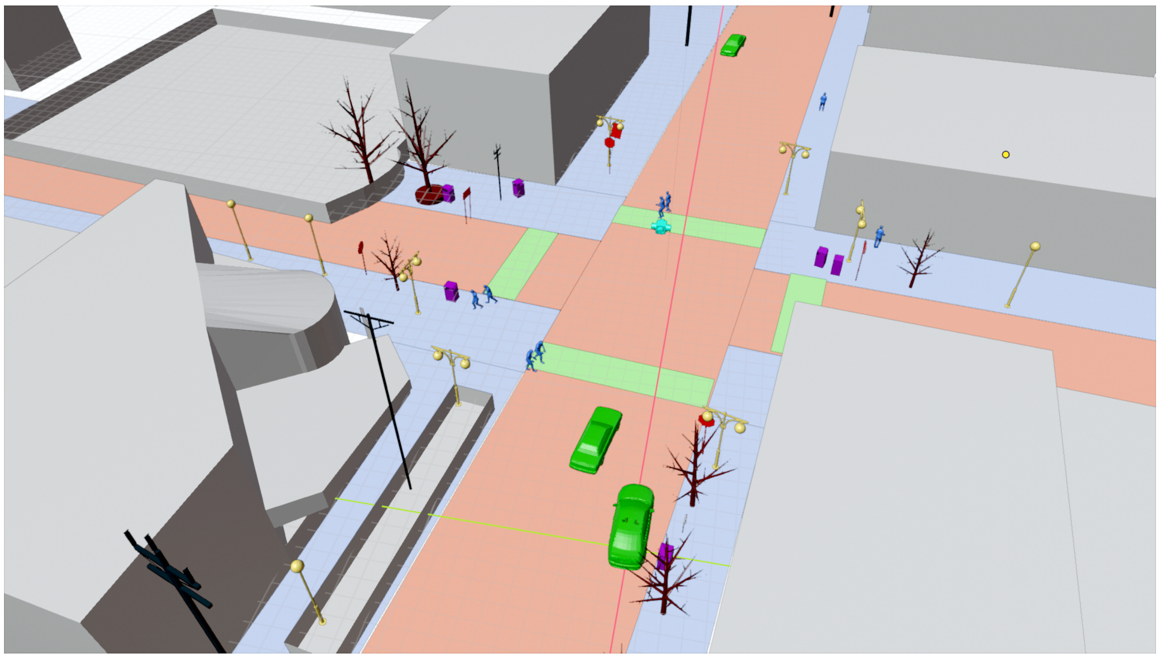


Figure 4.1: A reconstructed dense 3D scene rendered from a top-down perspective. Objects in different categories are shown in distinct colors.

the scene components and reconstruct the entire scene in dense 3D mesh models.

One of the benefits of reconstructing a whole 3D scene is that we can overcome the challenge of our real data capturing process. During the data capturing process [74], the capture vehicle was parked at a curb of the intersections. Our captured data has limited camera perspective images and LiDAR point clouds, lacking variability in image appearance. Moreover, it is practically challenging to capture real data covering the entire region of the intersection due to the limited number of sensors placed concurrently around wide intersections. By placing virtual cameras at arbitrary positions in a reconstructed scene, we can generate the simulated data, such as semantic maps, depth images, and LiDAR point clouds. The key difference from the existing synthetic datasets [25, 75, 76] is that our simulated data is generated using the real trajectories of dynamic agents in real scene structures.

The contributions of this chapter are: (1) reconstructing urban intersection scenes using dense 3D models, (2) proposing a distance metric between 3D mesh and a set of points, (3) simulating real sensors placed at reconstructed scenes, and (4) proposing an application to solving ego-vehicle trajectory prediction problem using the simulated data from the reconstructed scenes.

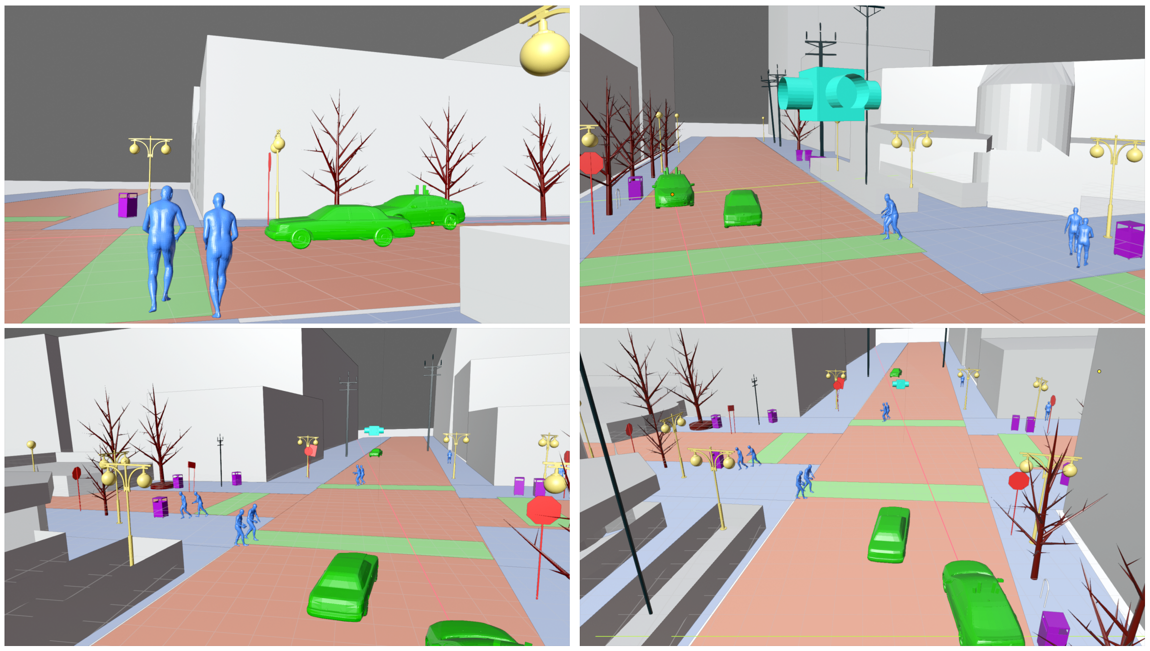


Figure 4.2: A reconstructed dense 3D scene rendered from varying perspectives.

4.2 Related Work

4.2.1 Real Urban Traffic Datasets

As AVs expand the scope of automated driving to complex urban areas, the importance of large-scale datasets in natural urban driving scenes increases. Despite many challenges, there has been extensive research on building urban driving datasets in recent years.

The KITTI dataset [77, 78] provides a rich set of labels for various perception tasks of autonomous driving. The data was captured from a moving vehicle equipped with stereo cameras and a LiDAR sensor. The Cityscape dataset [79] contains stereo images of a wide urban area with semantic labels. The ApolloScape dataset [20] also provides sequences of LiDAR points and RGB images captured from a moving vehicle along with labels for perception tasks. The dataset covers various traffic conditions with large complexity. The Oxford RobotCar dataset [21] offers data captured from a vehicle traveling around a city in various weather conditions. The data was captured while revisiting the same location at different times for evaluating localization. The BDD100K dataset [22] is a large-scale driving video dataset for various image recognition tasks.

Datasets more recently released were used as well. The Honda 3D (H3D) dataset [23] is a large-scale dataset consisting of dense LiDAR point clouds captured at highly interactive urban scenes. The dataset provides 3D labels for object detection and tracking. The Waymo Open dataset [24] contains large-scale multimodal data captured by a vehicle traveling around a wide range of areas across multiple urban cities. The dataset provides a rich set of 2D and 3D labels. The nuScenes dataset [3] is a multimodal dataset captured while a vehicle was operating in dense urban areas. The dataset provides labels for many tasks, including 3D detection and tracking.

4.2.2 Synthetic Traffic Datasets

In order to reduce the cost of manual labeling, large-scale synthetic datasets have been publicly released. An automated system based on deep learning algorithms was proposed [25] to generate synthetic data from the video game. A real-to-virtual world cloning method is proposed to construct a Virtual KITTI dataset [75] automatically labeled for many computer vision tasks. The SYNTHIA dataset [76] consists of synthetic data and semantic scene labels in the context of driving scenarios generated using the Unity game engine.

4.2.3 Intersection Datasets with Vehicles and Pedestrians

Urban intersection scenes are captured as part of the aforementioned real urban traffic datasets. Since data is captured from a non-static vehicle traveling in urban areas, the datasets usually lack the frames to observe interesting behaviors of dynamic agents at urban intersections. At some frames, vehicles stay static at an intersection, waiting for the traffic signals. However, in such cases, vehicle motions are likely to be more dependent on traffic signals rather than other agents.

Capturing multimodal data covering an entire intersection is challenging in practice. A vehicle traveling an intersection can only capture partial data during a limited duration of time. Moreover, urban intersections are usually too wide to cover with a limited number of sensors. Even if many sensors are available to use, calibrating and time-synchronizing all the sensors is troublesome.

One of the most common ways to gather meaningful intersection data is to capture bird's-eye view data using drones or installing sensors at a fixed height. The inD dataset [2] provides video recordings of unsignalized real urban intersections captured from a camera-equipped drone. The trajectories of road users are labeled using deep learning algorithms. CITR and DUT datasets [70] were constructed to capture vehicle-crowd interaction (VCI). The CITR dataset was captured with controlled VCI scenarios, while the DUT dataset consists of natural VCIs in crowded university campus. While a little different from intersections, several roundabouts datasets exist with vehicles and pedestrians in real driving scenes [71–73].

By capturing top-down view data, the influence of capturing process on the behavior of road agents can be reduced, and the occlusion of target objects can be minimized. Nevertheless, one downside of collecting bird's-eye view data is that view frustum and perspective are far apart from those of real AVs. Most AVs have perception sensors such as rotating LiDAR sensors and cameras on their roofs.

4.2.4 Pedestrian Trajectory Prediction

The problem of predicting future pedestrians' trajectories has been explored in many recent studies with a growing number of applications, from social robot navigation to autonomous driving. However, predicting the trajectories of pedestrians is a challenging problem, especially in crowded environments, because complex interactions cause high uncertainty of future pedestrians' motions. One of the most recent breakthroughs to tackle the trajectory prediction problem has been made by formulating the problem as a sequence generation task using recurrent neural networks. In recent studies, various types of observations have been considered for prediction: the past spatial coordinates, contextual information about agents' surroundings, and the information about their destinations.

Social Interactions: As one of the most important contextual cues, social interactions between pedestrians have been considered in multiple recent works [80–84]. To model the social intersections, social pooling layers have been widely used [80–82] to capture dependencies between sequences of other nearby pedestrians. The social pooling mechanisms enable networks to learn human-to-human interactions and jointly predict all the pedestrians' trajectories in a scene. As another way of modeling human-to-human interactions, attention layers have been introduced to predict future trajectories where pedestrians in a sequence are modeled as a spatio-temporal graph [83–89]. While the social pooling-based architectures uniformly weigh other pedestrians, modeling pedestrians using a spatio-temporal graph with attention layers puts different weights on other pedestrians.

Static Scene Context: As another type of contextual information, static scene context has been incorporated as additional inputs [82, 83, 90–93]. Pedestrians' trajectories can be affected by scene components, from static obstacles to background scene information. For example, pedestrians can change their trajectories to avoid static obstacles or non-reachable areas. Scene characteristics can also affect the trajectories. To incorporate scene context, scene-scale information was directly extracted from input images [82, 90, 92]. Instead of raw images, simplified images, such as HD maps, have been used to extract scene context [88, 94] for predicting future trajectories.

Other Context: In addition to social interactions and the scene context, some other forms of information have been explored. For example, considering pedestrians' view frustum was proved effective for trajectory prediction to simulate the limited visibility of pedestrians in actual scenarios [95]. First-person view images rendered using a synthetic simulator FvSim were proved effective to improve the trajectory prediction [96]. Pedestrians' long-term destination or vehicle's ego-agent motion plans have been recently considered [88, 97, 98] to explain the inherent multi-modality of human trajectories.

4.2.5 Vehicle Trajectory Prediction

There exist a few notable differences between vehicle and pedestrian trajectory prediction. Vehicle trajectory prediction algorithms usually produce predictions in longer time horizons than pedestrian ones due to faster speeds. Also, vehicle trajectory is generally considered more straightforward than pedestrian due to less uncertainty and complexity. That is because vehicles have limited maneuvers in practice, which limit their short-term motions. Also, the vehicles without any traffic violations stay on lanes or drivable areas. Despite the differences, the vehicle trajectory prediction can be tackled using approaches similar to the pedestrian trajectory prediction with vehicle scene datasets [94, 99–105]. Alternatively, there has been work to jointly predict trajectories for heterogeneous traffic agents, including vehicles as well as pedestrians [85, 106].

As one of the key features for vehicle trajectory prediction, many recent works have focused on the multimodality of vehicle trajectories. For example, vehicle motion is classified into semantically interpretable maneuver classes to predict future vehicle trajectories conditioned on each maneuver class [99, 102, 107, 108]. Destination or planning have been coupled to predict future vehicle motions [105].

Table 4.1: A list of scene components in our selected intersections.

Background	Static objects	Dynamic objects
	Buildings	
Ground	Trees	
Lanes	Lampposts	Pedestrians
Sidewalks	Road poles	Vehicles
Crosswalks	Traffic signs	
Parking lots	Trash bins	
	Bike racks	

4.2.6 Ego-vehicle Trajectory Prediction

The trajectory prediction problem has been formulated in a slightly different setting with ego-vehicle [109] perspectives. Instead of predicting future trajectories of observed road agents, the problem aims to predict short-term future trajectories of the ego-vehicle. It is beneficial to compute the probabilities of ego vehicle trajectories for collision avoidance and also for short-term driver assistance. It can be a meaningful signal to AV planning.

4.2.7 Datasets for Trajectory Prediction

Common benchmark datasets for pedestrian trajectory prediction consist of 2-dimensional spatial coordinates of agents from a Bird's-Eye View (BEV) along with RGB images [110], Traffic datasets exist that contain multiple object categories, including pedestrians and vehicles [2, 3, 20, 106]. The BEV images [2] or perspective view images [3, 20] are provided for those datasets.

4.3 Dense 3D Reconstruction

Urban intersections consist of many objects, from static objects such as buildings and traffic signs, to dynamic objects, including pedestrians and vehicles. In addition to volumetric objects, other components comprise the background of the scene, such as the ground and crosswalks. We have selected the objects observed from our urban intersections and are likely to affect the actions of dynamic agents. Table 4.1 shows the list of scene components.

In this section, we summarize the characteristics of each scene component and present the process of our dense 3D reconstruction based on model fitting.

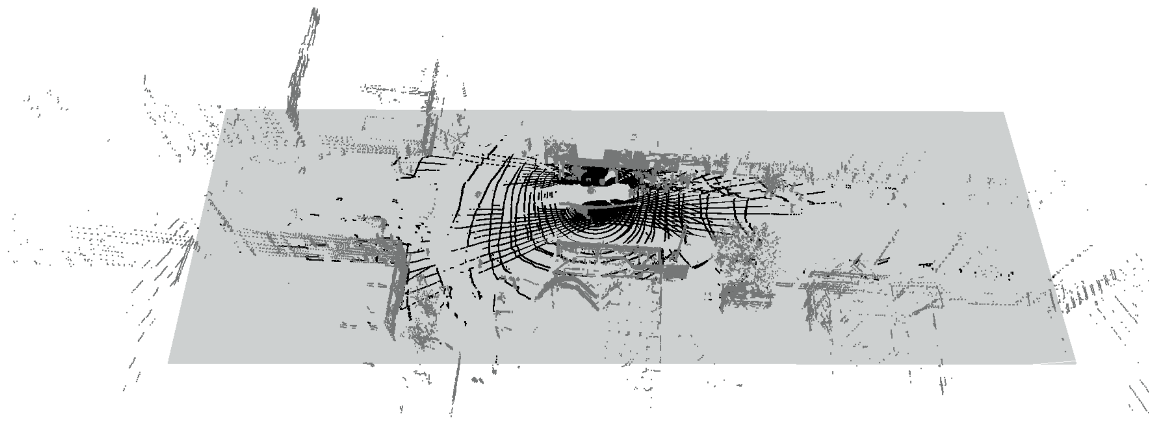


Figure 4.3: Ground plane fitting. Points on the ground (black), points not on the ground (gray).

4.3.1 Scene Background

4.3.1.1 Ground Fitting

Our data covers a wide area at an intersection. We fit a single plane to the ground for simplicity while the actual ground plane has an uneven surface. We manually segment the 3D points on the ground plane and find the plane parameters that minimize the root mean squares (RMS) error:

$$\boldsymbol{a}^*, b^* = \operatorname*{argmin}_{\boldsymbol{a},b} \sum_{i=1}^{N} \left(\mathbf{x}_i^T \boldsymbol{a} + b\right)^2 \tag{4.1}$$

$$\mathcal{P} = \left\{\boldsymbol{z} \in \mathbb{R}^3 \mid \boldsymbol{a}^T \boldsymbol{z} + b = 0\right\} \tag{4.2}$$

where N is the number of points in the segmented point cloud, $\mathbf{x}_i$ is the coordinate of the i^{th} point, and $\boldsymbol{a}$ and b define the 3D ground plane $\mathcal{P}$. The least-squares solution gives the plane parameters.

4.3.1.2 Lanes, Sidewalks and Crosswalks

Non-volumetric objects, such as lanes, sidewalks, and crosswalks, do not occupy 3D space. We represent these planar objects as plane segments on the ground plane. For each of our intersections, we fit two lanes, four sidewalks, and four crosswalks. (Figure 4.4)

Priors on lanes, sidewalks, and crosswalks encode reachable areas for dynamic agents in a scene. These planar components can be one of the critical signals for

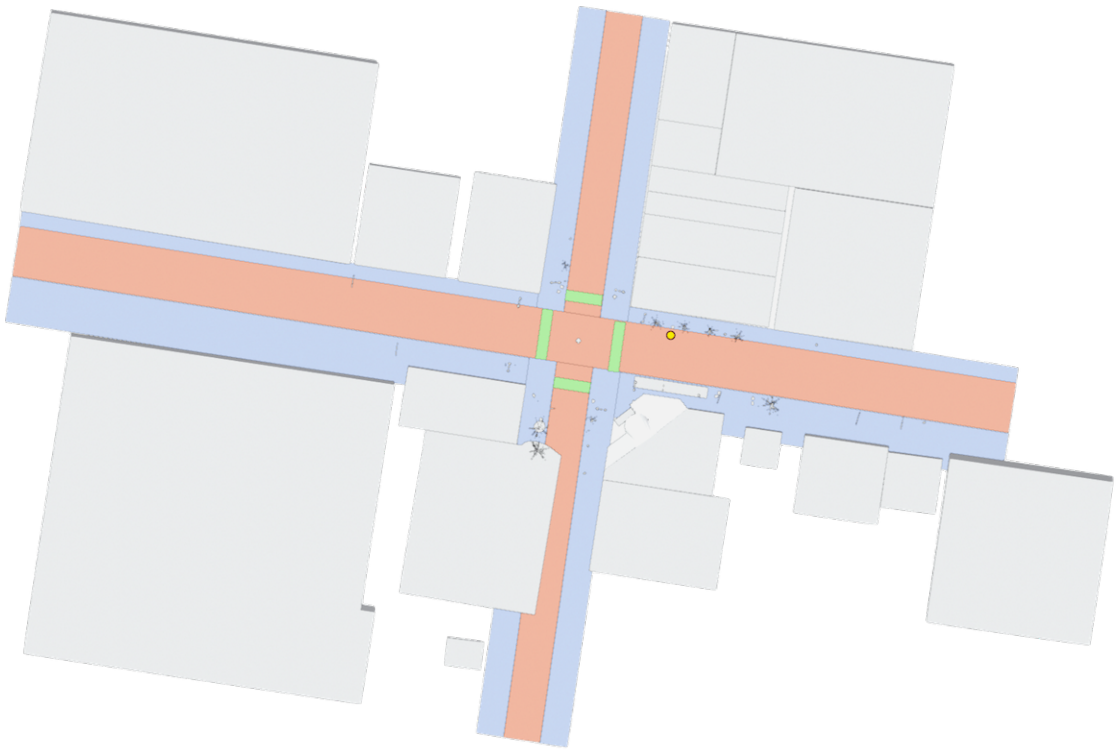


Figure 4.4: An example rendered view of a scene from the camera placed 500m away from the ground and heading towards $+z$ direction of the global coordinate frame. Lanes, sidewalks, and crosswalks are shown in orange, blue, and green.

vehicles or pedestrians to decide their trajectories. For example, sidewalks are not for driving; thus, vehicles do not pass on them. Pedestrians and vehicles usually slow down their speed before crosswalks to check their safety.

4.3.2 Static Objects

Our intersection scenes contain multiple types of static objects. For the objects with a rigid shape, such as lampposts, road poles, traffic signs, and trash bins, we collected the template 3D mesh models from the existing dataset [111]. Since the 3D models from the dataset are normalized to fit into a unit sphere, we scaled the models to fit into the real-world metric space.

For the buildings and other objects that are not supported by the existing datasets, we created our own triangulated mesh models using one of the open-source computer graphics softwares [112].

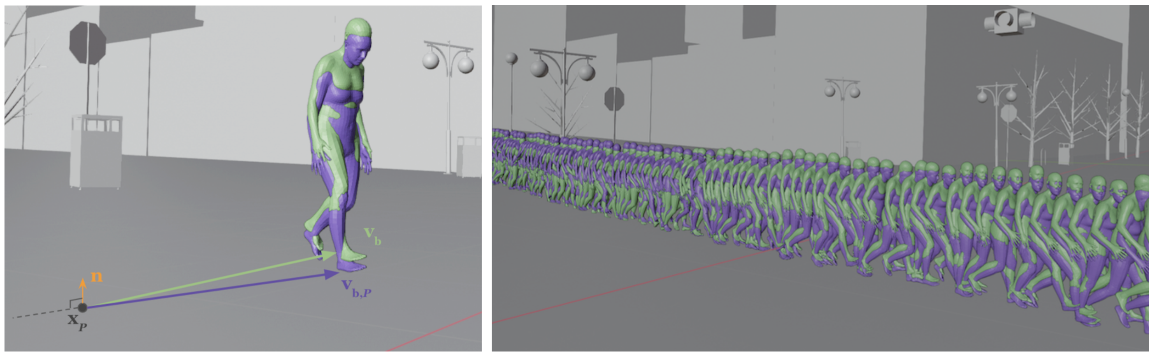


Figure 4.5: Before (green) and after (purple) the refinement

4.3.3 Dynamic Objects

4.3.3.1 Pedestrians

We use 3D pedestrian mesh models from the *PedX* dataset [74]. To make the models consistent with the reconstructed scene, we adjusted the models by enforcing a few constraints given by the known scene structure.

Assuming that no pedestrians jump in the air or stay floated above the ground, we enforce the bottom-most vertex of a pedestrian mesh to be right above the ground plane with added translation. The bottom-most vertex $\mathbf{v}_b$ is determined as follows:

$$\mathbf{v}_b = \underset{\mathbf{v}\in Mesh}{\operatorname{argmin}} \left(\mathbf{v} - \overline{\mathbf{v}}\right)^T \hat{\mathbf{n}} \tag{4.3}$$

where $\overline{\mathbf{v}}$ is the centroid of all the vertices on the mesh surface and $\hat{\mathbf{n}}$ is the unit normal vector of the ground plane $\mathcal{P}$ towards the sky. To snap the model to the ground, we shifted the mesh model by the following amount:

$$\Delta\mathbf{v} = \mathbf{v}_{b,\mathcal{P}} - \mathbf{v}_b = -\left(\mathbf{v}_b - \mathbf{x}_{\mathcal{P}}\right)^T \mathbf{n} \tag{4.4}$$

where the $\mathbf{v}_{b,\mathcal{P}}$ is the bottom-most vertex snapped to the ground plane $\mathcal{P}$. (Figure 4.5)

To ensure the pedestrians do not overlap with the reconstructed scene, we generate the ground-level occupancy map and check the occupancy. The map encodes whether any static object occupies the ground surface. Note that the shape of most objects is not uniform. For example, trees only take up a small area on the ground plane while occupying a large area from the top-down perspective. To make the map encode only the ground-level occupancy, we slightly shift the ground plane along the normal direction to slice the meshes of all the static scene components. Projecting the sliced

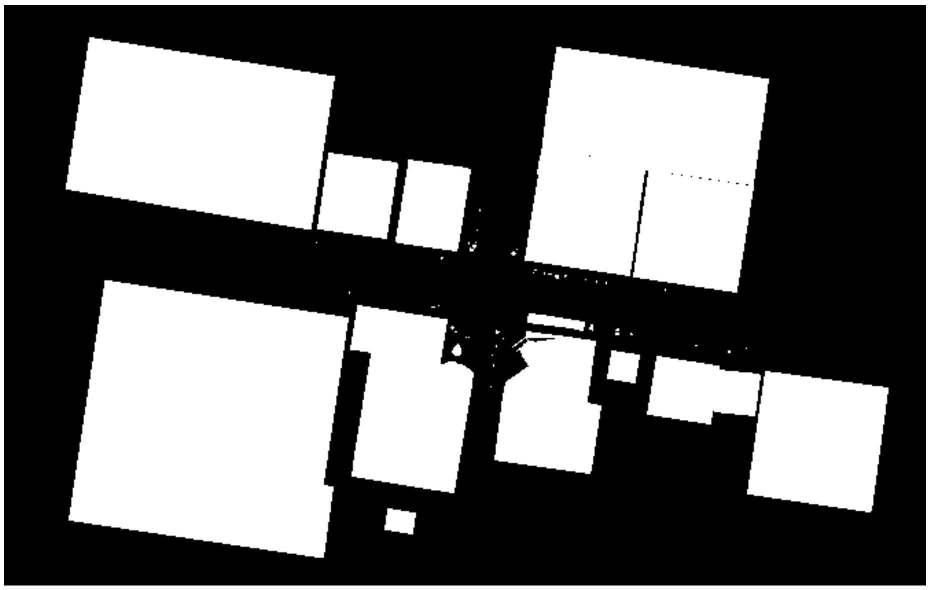


Figure 4.6: An example ground-level binary occupancy map.

mesh onto the ground plane gives the occupancy map. (Figure 4.6)

4.3.3.2 Vehicles

We make several assumptions on vehicle motions at our unsignalized intersections. We assume that vehicles do not move backward while passing through the intersections. All the vehicles are assumed to follow the traffic rules and standard norms such that they follow the driving lanes, temporarily stop before the stop sign and avoid hitting other objects. The vehicles are also assumed to move parallel to the ground plane. We ignore the vehicles parked alongside curbs.

Many subcategories exist for vehicles. Following the subcategories defined in several existing 3D object benchmark datasets [111, 113], we consider the vehicle subcategories shown in Figure 4.7. We present the details of the vehicle model fitting process in Section 4.4.

4.4 3D Vehicle Fitting

Given a sequence of data frames consisting of stereo images and LiDAR point clouds, our vehicle fitting process aims to localize moving vehicles in a scene, estimate their pose, and fit 3D template models. We perform the vehicle fitting in the following steps: point cloud segmentation, global trajectory fitting, pose estimation, and 3D model fitting. The entire pipeline for the 3D vehicle fitting is depicted in Figure 4.8.

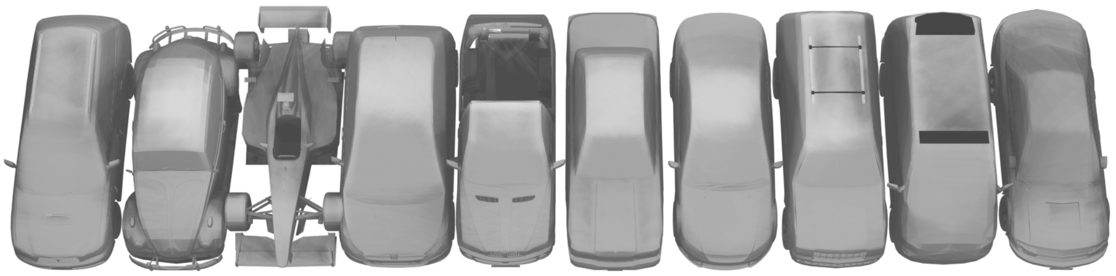


Figure 4.7: Template 3D vehicle models

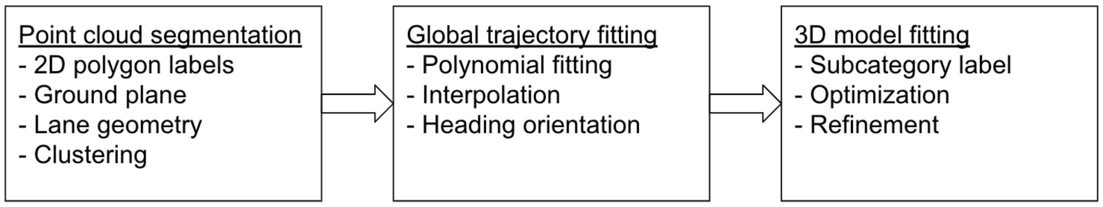


Figure 4.8: Pipeline of fitting 3D vehicle mesh models.

The pose of a vehicle is represented in terms of translation and heading orientation. 3D template models are fitted to the instances of moving vehicles with additionally knowing vehicle subcategories. The translation and heading orientation are represented as three-dimensional unit vectors defined in the PedX global coordinate frame. Note that the LiDAR point clouds reside in the same coordinate frame.

4.4.1 Point Cloud Segmentation

We begin with segmenting distinct vehicle instances from LiDAR point clouds. With the known geometry of a scene from the reconstructed 3D scene models, we first remove the points that belong to the scene backgrounds. For example, the points below the ground plane or the points whose height is larger than the vehicle height can be discarded. Using the known positions of static objects from the reconstructed 3D scene models, the points projected onto the sidewalk or other static objects from the top-down perspective can also be deleted because vehicles do not pass the sidewalks and are unable to share the space with other objects. To finally identify the points that belong to each instance, the points are projected onto the image planes and are compared with instance-level 2D polygon labels on camera images.

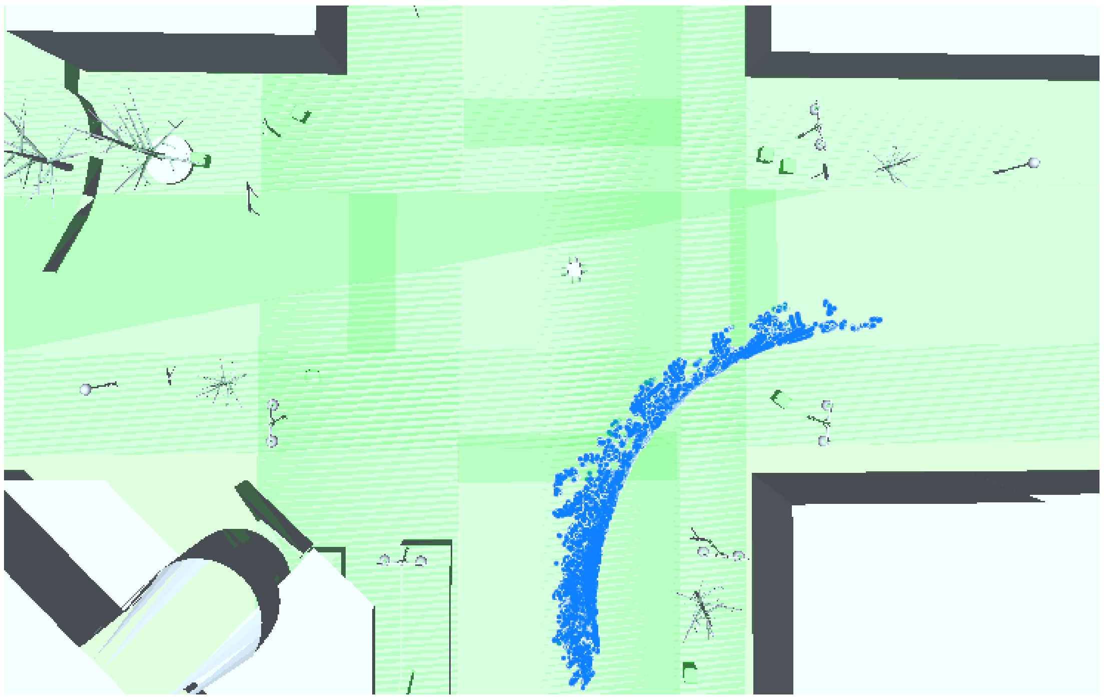


Figure 4.9: Point cloud segments (blue) show smooth trajectory over time.

4.4.2 Global Trajectory Fitting

After obtaining instance-level point cloud segments for all the frames in a sequence, we fit a polynomial to find the long-term global trajectory of a vehicle. The fitted global trajectory is used to initialize the heading orientation of vehicle instances in a sequence. Unlike the pedestrians, vehicles do not change their heading orientation rapidly due to their faster speed and rigid object shape. Therefore, heading orientations tend to be consistent with the global trajectory, as shown in Figure 4.9.

Based on the fitted global trajectory, we initialize both heading orientation and translation vectors of a vehicle by interpolation, which we denote by $\mathbf{h}_0$ and $\mathbf{x}_0$.

4.4.3 3D Model Fitting as Optimization

The previous steps of segmenting point clouds and fitting global trajectories involve some errors due to the noise in 2D polygons and partially observed LiDAR points. To find the heading orientation and translation vectors of a consistent vehicle to the LiDAR points and the reconstructed scene models, we fit 3D vehicle mesh models to the sequence.

Since vehicles move parallel to the ground plane, we enforce heading orientation vectors and the difference in two consecutive translation vectors parallel to the ground

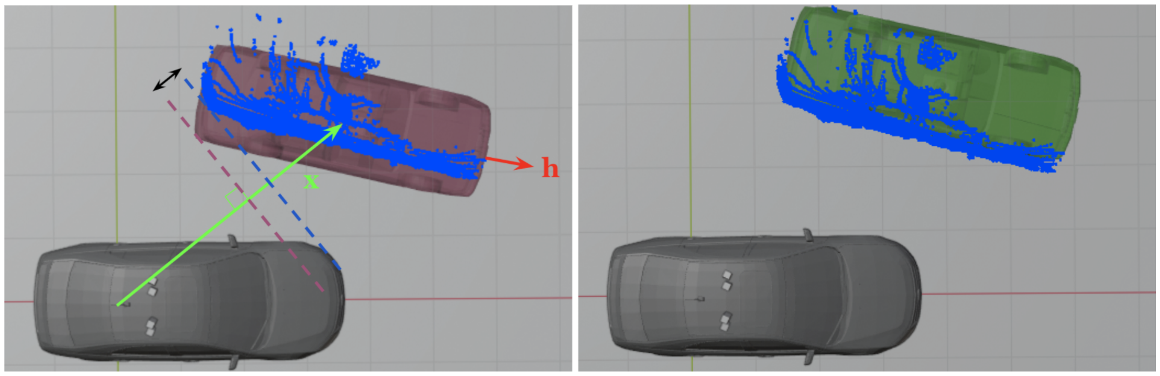


Figure 4.10: An instance-level 3D vehicle fitting by snapping the model to the point cloud segment. The vehicle in gray represents our capture vehicle where the origin of global coordinate frame is located. The red and green vehicles show the fitting results before and after snapping to the point cloud segment.

plane. By enforcing the ground plane constraint, we can reduce the 3D model fitting problem as an optimization problem in 2-dimensional translation and heading orientation vectors. In this subsection, we propose multiple prior terms to formulate the optimization problem over the translation and heading vectors of a vehicle in a sequence.

4.4.3.1 Notation

We represent the translation and heading vectors in the world coordinate frame as ${}^{W}\mathbf{x}_t$ and ${}^{W}\mathbf{h}_t$ at time t. Note that the z-axis of the world coordinate frame does not perfectly align with the ground normal. Using the rigid body transformation in SE(3), we transform the vectors to the coordinate frame where the z-axis is parallel to the ground normal and its origin is on the ground plane:

$$ {}^{G}\mathbf{x}_t = {}^{G}_{W}R\left({}^{W}\mathbf{x}_t - {}^{W}\mathbb{O}_G\right) \in \mathbb{R}^3 \tag{4.5} $$

where W and G denote the world coordinate frame and the coordinate frame whose z-axis is aligned to the ground plane normal, respectively. ${}^{G}\mathbf{x}_t$ is the position of the vehicle center in the frame G and ${}^{G}_{W}R \in \mathrm{SO}(3)$ is the rotation matrix that converts the coordinates from the initial world frame W to the frame G. ${}^{W}\mathbb{O}_G$ represents the origin of the frame G observed in the frame W. We use ${}^{W}\mathbb{O}_G = [0, 0, z]^T$ which is a point on the ground plane.

We can obtain the 2-dimensional vector projected onto the ground plane by taking the first two components of ${}^{G}\mathbf{x}_t$:

$$
{}^{\mathcal{P}}\mathbf{x}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} {}^{G}\mathbf{x}_t \in \mathbb{R}^2 \tag{4.6}
$$

Conversely, the 2-dimensional vector ${}^{\mathcal{P}}\mathbf{x}_t$ can be converted back into the three-dimensional vector in the world frame using known height z of the vehicle center from the ground plane:

$$
{}^{W}\mathbf{x}_t = {}^{W}_{G}R \begin{bmatrix} {}^{\mathcal{P}}\mathbf{x}_t \\ \hdashline z \end{bmatrix} + {}^{W}\mathbb{O}_G \tag{4.7}
$$

Similarly, the 2-dimensional heading vector ${}^{\mathcal{P}}\mathbf{h}_t$ projected onto the ground plane can be converted into the three-dimensional vector in the world frame. Since the heading vectors are parallel to the ground plane, we can ignore the z component by setting $z = 0$ in (4.7):

$$
{}^{W}\mathbf{h}_t = {}^{W}_{G}R\, {}^{G}\mathbf{h}_t = {}^{W}_{G}R \begin{bmatrix} {}^{\mathcal{P}}\mathbf{h}_t \\ \hdashline 0 \end{bmatrix} \tag{4.8}
$$

A sequence consists of T continuous frames. For simplicity, we represent the concatenation of T continuous translation and heading vectors as:

$$
\mathbf{x}_{1:T} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \quad \mathbf{h}_{1:T} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_T \end{bmatrix} \tag{4.9}
$$

In the following sections, we introduce the constraints on translation and heading vectors used to formulate the optimization problem over $\mathbf{x}_{1:T}$ and $\mathbf{h}_{1:T-1}$. Note that we skip optimizing the heading vector of the last frame at time $t = T$ by assuming $\mathbf{h}_{T-1} = \mathbf{h}_T$.

4.4.3.2 Vehicles Never Heading Backwards

Since our captured data does not contain any traffic violations, we assume all the vehicles do not move backward. The constraint can be written as (4.10). To enforce this constraint, we minimize the term in (4.11):

$$\mathbf{h}_t^T\left(\mathbf{x}_{t+1}-\mathbf{x}_t\right) \geq 0 \tag{4.10}$$

$$E_{forward}(\mathbf{x}_{1:T}, \mathbf{h}_{1:T-1}) = \sum_{t=1}^{T-1} \left(-\mathbf{h}_t^T\left(\mathbf{x}_{t+1}-\mathbf{x}_t\right)\right)_+$$

$$\text{where } (x)_+ = \max\{x, 0\} \tag{4.11}$$

The gradient vectors of $E_{forward}$ can be evaluated as follows:

$$\frac{\partial E_{forward}}{\partial \mathbf{x}_t} = \begin{cases} \mathbf{h}_t \cdot \mathbb{1}_t & \text{if } t = 1 \\ \mathbf{h}_t \cdot \mathbb{1}_t - \mathbf{h}_{t-1} \cdot \mathbb{1}_{t-1} & \text{if } 1 < t < T \\ -\mathbf{h}_{t-1} \cdot \mathbb{1}_{t-1} & \text{if } t = T \end{cases} \tag{4.12}$$

$$\frac{\partial E_{forward}}{\partial \mathbf{h}_t} = -(\mathbf{x}_{t+1}-\mathbf{x}_t) \cdot \mathbb{1}_t$$

$$\text{where } \mathbb{1}_t = \begin{cases} 1 & \text{if } -\mathbf{h}_t^T(\mathbf{x}_{t+1}-\mathbf{x}_t) > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } 1 \leq t < T \tag{4.13}$$

4.4.3.3 Consistency between heading and translation vectors

To enforce a vehicle to translate parallel to its heading direction, we correlated translation and heading vectors as the following relation:

$$\mathbf{h}_t \;||\; (\mathbf{x}_{t+1}-\mathbf{x}_t) \tag{4.14}$$

where $\mathbf{h}_t$ is a heading vector at frame t with $\|\mathbf{h}_t\|_2 = 1$. We can rewrite this constraint in a quadratic form by relaxing it:

$$0 \leq (\mathbf{x}_{t+1}-\mathbf{x}_t)^T \mathbf{h}_t \leq \|\mathbf{x}_{t+1}-\mathbf{x}_t\|_2 \tag{4.15}$$

$$(\mathbf{x}_{t+1}-\mathbf{x}_t)^T H_t \left(\mathbf{x}_{t+1}-\mathbf{x}_t\right) \geq 0 \tag{4.16}$$

$$H_t = \mathbb{I}_2 - \mathbf{h}_t\mathbf{h}_t^T \tag{4.17}$$

for $\forall t \in \{1, \ldots, T-1\}$. $\mathbb{I}_2$ is a 2×2 identity matrix. For all the translation and heading vectors in a sequence, we can further simplify the constraint:

$$E_{parallel}(\mathbf{x}_{1:T}, \mathbf{h}_{1:T-1}) = \mathbf{x}_{1:T}^T \, H_{parallel} \, \mathbf{x}_{1:T} \tag{4.18}$$

which has a quadratic form in terms of $\mathbf{x}_{1:T}$. The term (4.18) becomes zero when (4.14) holds for all t. The matrix $H_{parallel}$ is defined as follows:

$$H_{parallel} = \begin{bmatrix} H_1 & -H1 & 0 & \cdots & 0 & 0 & 0 \\ -H1 & H_1 + H_2 & -H_2 & \cdots & 0 & 0 & 0 \\ 0 & -H_2 & H_2 + H_3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -H_{T-2} & 0 \\ 0 & 0 & 0 & \cdots & -H_{T-2} & H_{T-2} + H_{T-1} & -H_{T-1} \\ 0 & 0 & 0 & \cdots & 0 & -H_{T-1} & H_{T-1} \end{bmatrix} \tag{4.19}$$

The gradient vector of $E_{parallel}$ can be evaluated as follows:

$$\frac{\partial E_{parallel}}{\partial \mathbf{x}_{1:T}} = 2H_{parallel}\mathbf{x}_{1:T} \tag{4.20}$$

4.4.3.4 Distance Between Point Cloud and Mesh

To make the fitted 3D vehicle models close to the point cloud segments from LiDAR sensors, we define the distance between a point cloud and a vehicle mesh in three-dimensional space as follows:

$$E_{3D}(\mathbf{x}_{1:T}, \mathbf{h}_{1:T}) = \sum_{t=1}^{T} d(S_t, \mathcal{M}(\mathbf{x}_t, \mathbf{h}_t)) \tag{4.21}$$

$$d(S_t, \mathcal{M}(\mathbf{x}_t, \mathbf{h}_t)) = \frac{1}{|S_t|} \sum_{\mathbf{p} \in S_t} \|\mathbf{v}_{\mathbf{p}}(\mathbf{x}_t, \mathbf{h}_t) - \mathbf{p}\|_2^2 \tag{4.22}$$

$$\mathbf{v}_{\mathbf{p}}(\mathbf{x}_t, \mathbf{h}_t) = \underset{\mathbf{v} \in \mathcal{M}(\mathbf{x}_t, \mathbf{h}_t)}{\operatorname{argmin}} \left[1 - \frac{(\mathbf{v} - \mathbf{x}_t)^T (\mathbf{p} - \mathbf{x}_t)}{\|\mathbf{v} - \mathbf{x}_t\|_2 \|\mathbf{p} - \mathbf{x}_t\|_2} \right] \tag{4.23}$$

where $\mathcal{M}(\mathbf{x}, \mathbf{h})$ is a vehicle mesh transformed with heading $\mathbf{h}$ and translation $\mathbf{x} \in \mathbb{R}^3$, and $\mathbf{v} \in \mathcal{M}$ is a point on the mesh surface. $\mathbf{v}_{\mathbf{p}}$ returns the point on the mesh surface $\mathcal{M}$ that corresponds to the point $\mathbf{p} \in S_t$. The corresponding point on the mesh is computed in (4.23) by extending the point $\mathbf{p} \in S_t$ along the ray from $\mathbf{x}_t$, and finding the point on the mesh that intersects with the ray.

In (4.21), the three-dimensional vectors in the world coordinate frame can be computed as follows:

$$\begin{aligned} \mathbf{x}_t^W &= {}^W_G R_{*,1:2}\,\mathbf{x}_t + z\,{}^W_G R_{*,3} + \mathbb{O}_G^W \qquad (4.24) \\ \mathbf{h}_t^W &= {}^W_G R_{*,1:2}\,\mathbf{h}_t + \mathbb{O}_G^W \qquad (4.25) \end{aligned}$$

where z is the height of the vehicle center from the ground plane, which is a single constant for each template mesh model since we use a single vehicle model across a sequence, and the fitted vehicles are assumed to be on the ground plane.

4.4.3.5 Heading Smoothness Prior

Since vehicles do not rapidly change the heading orientation between frames, two neighboring heading vectors are close. We minimize the following term to enforce neighboring heading vectors to be similar:

$$E_{smooth}(\mathbf{h}) = \sum_{t=1}^{T-1} \|\mathbf{h}_{t+1} - \mathbf{h}_t\|_2^2 = \mathbf{h}^T A_{smooth}\mathbf{h} \qquad (4.26)$$

$$\text{where} \quad A_{smooth} = \begin{bmatrix} I & -I & 0 & \cdots & 0 & 0 \\ -I & 2I & -I & \cdots & 0 & 0 \\ 0 & -I & 2I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2I & -I \\ 0 & 0 & 0 & \cdots & -I & I \end{bmatrix} \in \mathbb{R}^{2T\times 2T} \qquad (4.27)$$

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_T \end{bmatrix} \in \mathbb{R}^{2T} \qquad (4.28)$$

4.4.3.6 Optimization Solver

We minimize the sum of aforementioned objective terms to find the optimal translation and heading vectors:

$$\underset{\mathbf{x}_{1:T},\mathbf{h}_{1:T-1}}{\text{minimize}} \; E\left(\mathbf{x}_{1:T}, \mathbf{h}_{1:T-1}\right) \qquad (4.29)$$

$$E = w_p E_{parallel} + w_f E_{forward} + w_r E_{raytrace} + w_s E_{smooth} \qquad (4.30)$$

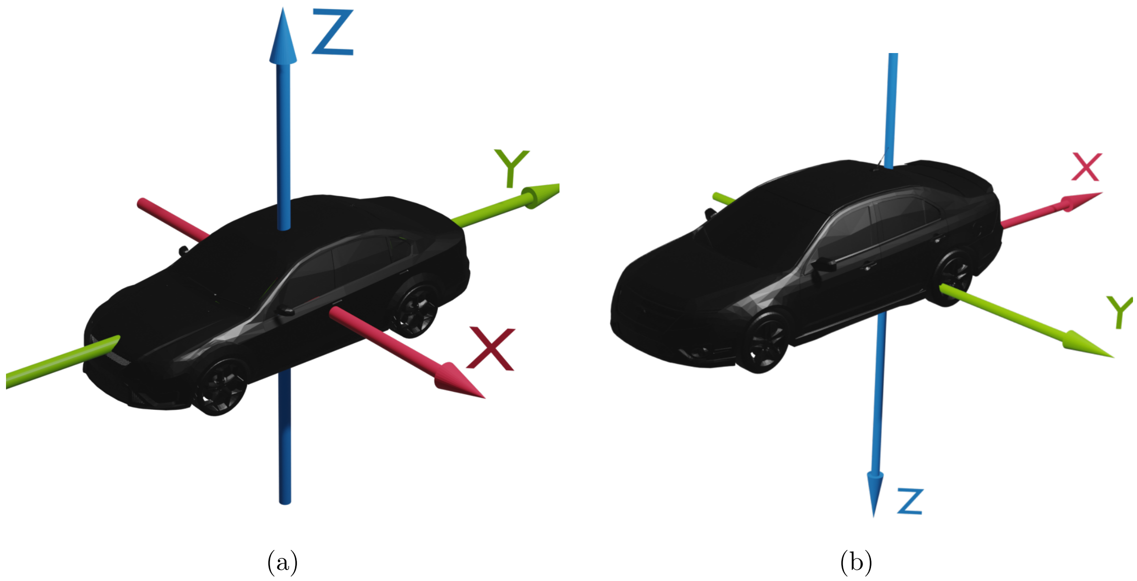


Figure 4.11: (a) An object-centered coordinate frame of the 3D vehicle models from the ObjectNet3D dataset (b) The world coordinate frame of the PedX dataset shown with our capture vehicle.

where we used $w_f = 10^0$, $w_p = 10^{-2}$, $w_r = 10^{-4}$ and $w_s = 10^{-3}$. We used larger weights for the $E_{forward}$ and $E_{parallel}$ terms than the others. We enforced the first two constraints harder because the movement looks less natural and smooth when the vehicle poses violate those terms. For the $E_{raytrace}$ term, LiDAR point clouds often contain outliers, so we use smaller weights for this term.

We use one of the first-order gradient-based solvers (BFGS algorithm) to optimize all the pose vectors in a sequence jointly. The solver converges to reasonable solutions with good initial poses from the fitted global trajectories.

4.4.3.7 3D Model Alignment

Once we compute the transition and heading orientation vectors, the final step is to align one of the template 3D mesh models according to the estimated vehicle poses. The coordinate frame of the ObjectNet3D template models are depicted in Figure 4.11a. We apply the following transformation to the vertices of template mesh models to align the model to the estimated translation and heading vectors ${}^{G}\mathbf{h}$ and

${}^{G}\mathbf{x}$:

$$A = \left[\ {}^{G}\hat{\mathbf{x}} \,\middle|\, {}^{G}\hat{\mathbf{y}} \,\middle|\, {}^{G}\hat{\mathbf{z}}\ \right] \tag{4.31}$$

$$B = \left[\ -\,{}^{G}\mathbf{h} \times {}^{G}\hat{\mathbf{z}} \,\middle|\, -\,{}^{G}\mathbf{h} \,\middle|\, {}^{G}\hat{\mathbf{z}}\ \right] \tag{4.32}$$

$${}^{O}_{G}R = A\, B^{T} \tag{4.33}$$

$${}^{O}_{G}H = \begin{bmatrix} {}^{O}_{G}R & -\,{}^{O}_{G}R\ {}^{G}\mathbf{x} \\ \mathbb{O}_{1\times 3} & 1 \end{bmatrix} \tag{4.34}$$

4.5 Experiments

4.5.1 Verification of The Vehicle Models

To validate our model-fitting pipeline, we projected the estimated 3D models onto the image planes and computed the distance between the 2D segmentation masks and the projected models. The 2D segmentation masks are obtained from the manually labeled 2D polygons. As a distance metric between the 2D segmentation masks and the projected 3D models, we used an Intersection-over-Union (IoU). Table 4.2 shows the results.

Due to the constraints enforced when fitting models, the IoU computed using 2D segmentation masks was low. The degraded IoU mainly comes from several assumptions that we made. To make the densely reconstructed 3D scene models consistent, we snapped the vehicle models onto the ground plane. This makes a slight deviation from the actual vehicle poses. Moreover, we used one of the template vehicle models for fitting, which cannot represent the actual vehicles with various shapes and sizes. As a slightly relaxed metric, we computed the IoU using 2D bounding boxes around the vehicles' visible pixels, which resulted in a higher value than the IoU with segmentation masks.

To validate the estimated vehicles poses in sequences, we compute the amount of violations for the constraints (4.10) and (4.14). Table 4.3 shows the values close to 1, which means the constraints have been mostly met.

4.5.2 Verification of The Reconstructed Scene Models

To validate the densely reconstructed 3D scene models, we simulated the LiDAR sensors to generate point clouds and compare the structure with the point clouds captured by real LiDAR sensors. The two sets of point clouds are aligned and visualized in Figure 4.12. The simulated point clouds are aligned well with the captured data

Table 4.2: IoUs computed using 2D segmentation masks and 2D bounding box labels.

IoU with segmentation masks	IoU with 2D bounding boxes
0.67	0.8

Table 4.3: The amount of violations for the constraints.

$E_{forward}$ (4.10)	$E_{parallel}$ (4.14)
0.95	0.97

from real LiDAR sensors. In contrast, the simulated data looks slightly denser and cleaner due to fewer disturbances from external noise sources such as sunlight and reflective surfaces.

4.5.3 Simulating Data From an Agent's Perspective

One of the weaknesses of the PedX dataset is that the data was captured from a static view. The LiDAR returns thereby do not cover the full 3D scene information due to the occlusions. To overcome those restrictions, we implemented a system that can generate simulated data using virtual sensors. Given our 3D scene models, we can place cameras at any arbitrary location in the real metric space. Since we do not consider textures and materials for our 3D meshes, we placed virtual depth cameras in a scene.

Figure 4.13 shows example results when we placed a virtual camera inside one of the moving vehicle models with known trajectory. Since we are given object trajectories from real scenarios, our simulated data is more realistic compared to other synthetic datasets. By rendering the 3D mesh models, we can obtain dense depth maps. Also, by rendering each distinct instance and finding the depth ordering, we can determine the semantic label images.

To simulate LiDAR point clouds, we uniformly generated vertices on the sphere within the vertical FOV range. Figure 4.14 shows the process of simulating a Velodyne HDL-32 LiDAR sensor, which consists of 32 vertical rings within a 40° vertical FOV and has a maximum range of 100 m. Among the vertices on the sphere surface, we first found the points within the camera's view frustum. Those vertices were projected onto the camera image plane. Note that not all the points reach to the image planes due to the occlusions by 3D object meshes. To simulate a 360° rotating LiDAR sensor, we repeated the same process while rotating the virtual camera around the rotating

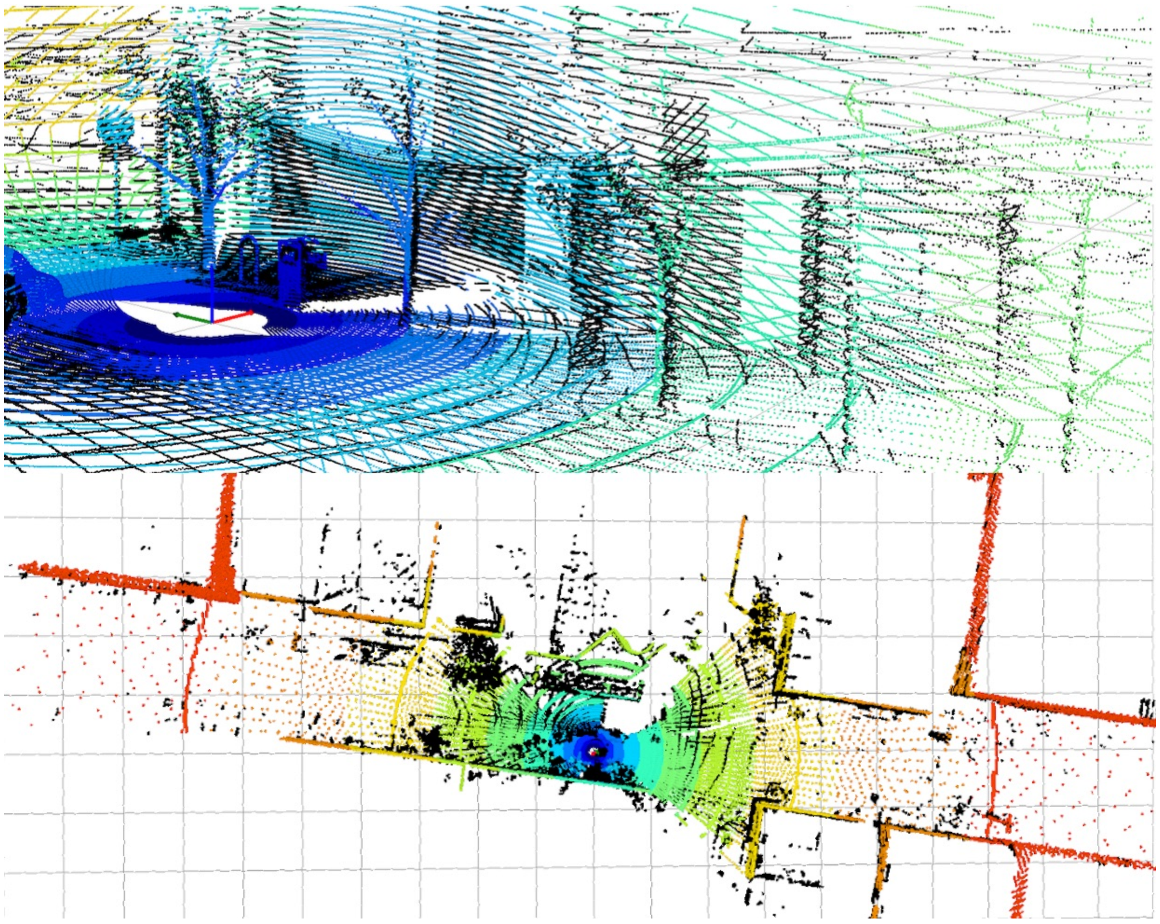


Figure 4.12: Comparison of real and simulated point clouds. Points from real LiDAR sensors are shown in black, and the simulated points are colored-coded based on the distance from the LiDAR origin.

axis.

4.5.4 Ego-Vehicle Trajectory Prediction

Predicting future trajectories of road agents is one of the critical components in many AV systems. Given the sensor inputs or perception results as inputs, future prediction can help improve safety and assist planning. However, predicting future states is challenging, especially in urban scenarios where multi-agents coexist with complex interactions.

There has been much progress in solving the trajectory prediction problem, particularly in scenes with pedestrians. Vehicle trajectory prediction has also been explored using approaches similar to the pedestrians. The underlying assumptions in the most recent work are that the trajectories of all the road agents are observable. To be

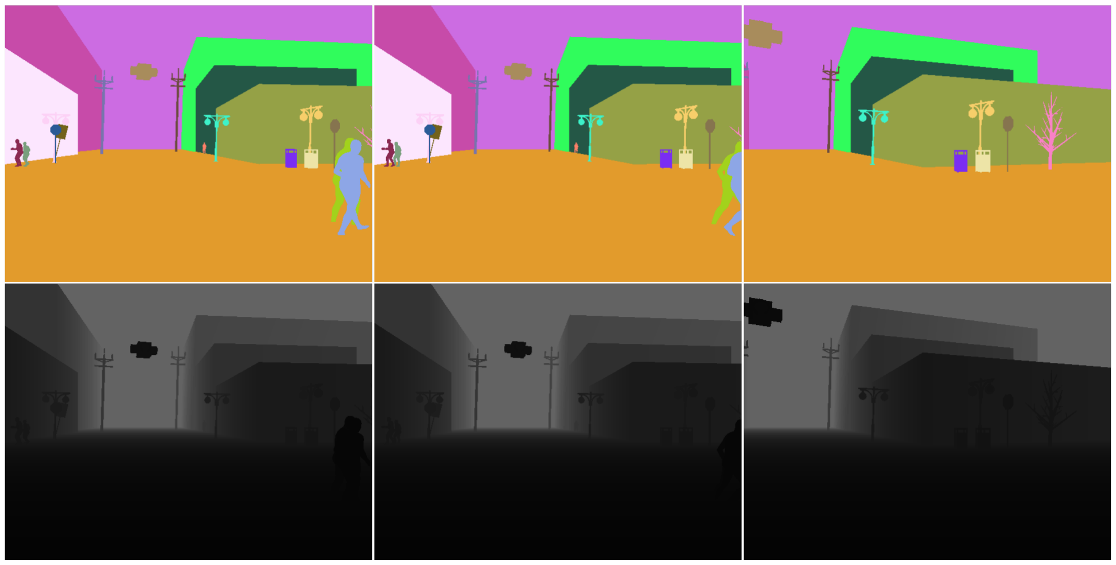


Figure 4.13: Instance-level label images and depth maps captured by a virtual camera placed at one of the moving vehicles.

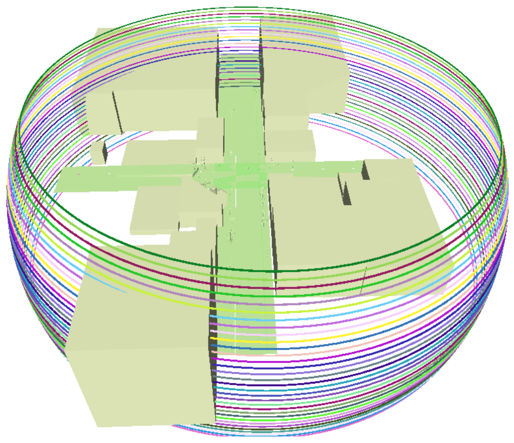


Figure 4.14: Simulating a 32-beam LiDAR sensor. Points that belong to different rings are shown in distinct colors.

specific, most trajectory prediction datasets consist of 2-dimensional spatial trajectories, BEV images of the entire scene, or top-down view HD maps. For example, some trajectory datasets were collected using drones [2, 18] to acquire trajectories with complete visibility and reduced influence by road agents. For the datasets collected using a running vehicle, the trajectories are manually labeled in LiDAR point space [3, 20].

Thinking of AVs in reality, however, BEV images are not readily available because sensors are generally equipped around the roof of vehicles. Even if we can obtain the track annotations by applying the perception algorithms to raw sensor data, the ultimate goal of future predictions might be to assist drivers or AVs in reducing potential risks while following the long-term route produced by the planning system.

This section introduces the ego-vehicle trajectory prediction problem as one potential application of our generated data. We experimented with a simple recurrent model to predict future trajectories using perspective images. Since there is no explicit modeling of the complex interactions with surrounding objects or tracking all the surrounding road agents, we suggest an approach to predict potential short-term future risks.

Figure 4.15 describes our proposed first-person trajectory prediction network. Given a sequence of past spatial coordinates and first-person view images, we aimed to predict the potential future trajectory in terms of spatial coordinates. As first-person view images, we propose using pixel-level segmentation label maps instead of RGB images due to the lack of RGB images in our ParametricX dataset. The network has an encoder-decoder structure. For the encoder to process and encode the observed past information, we used two separate LSTMs, one for encoding the observed spatial trajectory and the other for processing the observed scene images in terms of depth maps and pixel-level segmentation maps. The hidden states from the encoder were concatenated and fed into the decoder. An additional LSTM was used for the decoder to produce future predictions.

As a baseline, we also trained the Trajectron++ [88] model to solve multi-agent trajectory prediction problems. We trained the basic model by using spatial trajectories of all the road agents in a scene. Prior global maps are assumed to be not given.

The Waymo Open dataset [24] was used to train the networks because it provides a large number of ego-vehicle trajectories with semantic frame data. We sampled the data in $2Hz$ and formulated the problem to predict the future trajectories for the subsequent 6 frames given the maximum 8 past frames. Since the Trajectron++ model

Table 4.4: Comparison of the ego-vehicle trajectory prediction results with Waymo Open dataset.

Dataset	Trajectron++	Ours
ADE	0.65	1.07
FDE	1.71	1.94

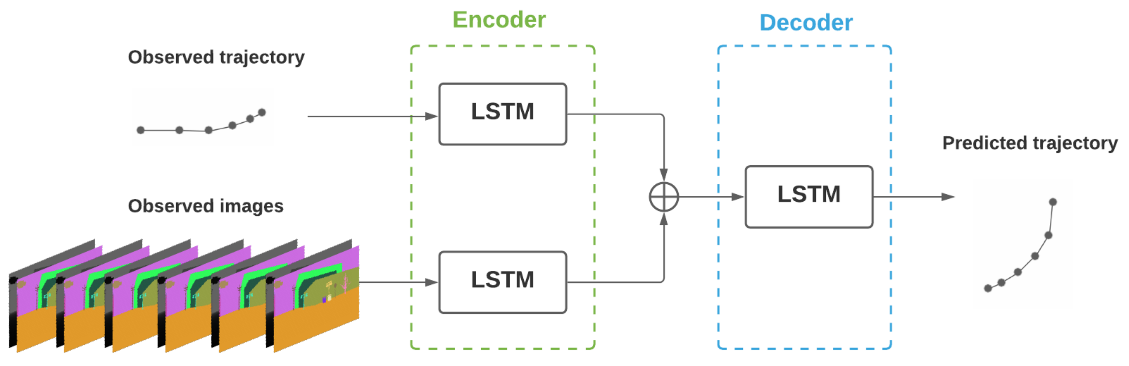


Figure 4.15: Ego vehicle trajectory prediction network overview.

outputs future trajectories for all the existing road agents, we take the predictions for the ego vehicle when evaluating.

Table 4.4 shows the results. We report two metrics as in many prior works: Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE is the average Euclidean distance between the predicted trajectory and the ground truth. FDE is the Euclidean distance between the final positions of the predicted and the ground truth trajectories.

4.6 Conclusion

We have presented an urban intersection dataset with dense 3D mesh models fitted to the entire scene. The data was captured at real unsignalized urban intersections using LiDAR and camera sensors from a static perspective. To overcome the challenges of gathering large-scale data at complex urban intersections, we propose ways to augment the raw capture data from reconstructed synthetic 3D scene models. Sensor data, such as LiDAR points, depth images, and semantic maps, can be generated by placing sensors at arbitrary locations within an intersection area.

While many urban traffic datasets for AVs have been recently released, not all the datasets are usable since configurations and specifications of sensors vary on different

AV platforms. The realistically simulated data that was generated based on real scene geometries and trajectories can be one initial step to build cross-platform datasets.

We also show a simple application to predict future trajectories of an ego vehicle. By predicting short-term future trajectories of a vehicle itself, the ego-vehicle trajectory prediction problem can aid human drivers or self-driving cars by detecting any abnormal short-term movement of the vehicles. Similar approaches can be applied in other domains if data becomes available. For example, it can be extended to navigate short-term paths for blind individuals with holding sensors.

CHAPTER V

Towards Realistic 3D Scene Models

5.1 Introduction

In Chapter IV, we performed the dense 3D reconstruction of urban intersections by fitting mesh models. The reconstructed scene models can generate simulated data to augment the dataset and overcome the cost of capturing and labeling real data. However, the mesh models used were texture-less with no materials applied. LiDAR points and labels can be easily simulated, but RGB cameras cannot be mocked using incomplete mesh models. To explore any solution to overcome this limitation, we looked into understanding the intrinsic properties of a scene. Many factors are required to achieve realistic and natural rendering from 3D models, from lighting and illumination to texture and material. This chapter can be a preliminary step towards a more realistic reconstruction of 3D scene models.

Understanding the intrinsic properties of a scene, such as illumination, lighting, material, texture, and reflectance, is critical for many vision-based robotic applications. The ability to estimate these properties is of great benefit for navigation, grasping, segmentation, and classification. The inability to disambiguate these properties may hamper robot performance in feature matching, object recognition, and 3D reconstruction. However, inferring the underlying intrinsic properties of a scene from an image is an ill-posed problem because many possible combinations of such properties can explain an image. Moreover, it is challenging to gather ground truth because measuring reflectance and shading from real-world data requires carefully calibrated laboratory equipment.

Prior work has mostly explored this problem from a single image using generative learning frameworks [4, 5]. They typically address the problem as an energy minimization task with constraints on reflectance and shading. The approaches in the literature are not real-time, taking upwards of one minute per frame. There have been attempts to estimate intrinsic image sequences from video [114, 115] in which the authors consider additional temporal constraints to improve consistency between frames. These methods are not suitable for robotic deployment as the processing requires forward and backward traversal through the video stream instead of the frame by frame processing. Additionally, the computation time is too high for practical use in a robotic system. In general, little prior research has directly addressed intrinsic image decomposition for robots.

This chapter aims to recover the intrinsic properties from a stream of RGB-D images that could come from a mobile robot or another moving platform. We propose a novel pipeline to tackle the intrinsic image decomposition problem by formulating it as an estimation problem. Rather than attempting to solve complete image decomposition at every frame, our proposed framework accumulates and propagates evidence on intrinsic properties over time to make use of the knowledge from past frames. We integrate state-of-the-art features into a Bayesian estimation pipeline to produce competitive results in a fraction of the time of a single-shot intrinsic image estimation technique. The contributions of this chapter are as follows: 1) we propose a novel architecture for recursive intrinsic estimation; 2) we reduce the running time of RGB-D stream image decomposition by order of magnitude from the state-of-the-art; 3) using a novel quiver of features to measure the similarity of the intrinsic of different image regions, we enforce much greater temporal consistency across frames; and 4) we provide a quantitative and qualitative evaluation of the proposed technique on real data and simulated data with ground truth.

This chapter is organized as follows: In Section 5.3, we first describe the proposed system. In Section 5.4, we discuss details about how we formulate the intrinsic image decomposition as an estimation problem. We present our resulting images and compare the results with other baseline methods in Section 5.5. Section 5.6 discusses the conclusions drawn and addresses future work.

5.2 Related Work

5.2.1 Intrinsic image decomposition

The intrinsic image decomposition problem aims to separate an image into shading and reflectance layers. The reflectance map shows how light is reflected by the object's surface, irrespective of viewpoint, geometry, and illumination. The shading map encodes the shading effects due to geometry, shadows, inter-reflections, etc. The most common intrinsic image model states that an image I is the element-wise product of the reflectance map R and the shading map S as follows:

$$I_i = R_i \times S_i \quad \text{for all pixels } i \tag{5.1}$$

While the intrinsic image decomposition problem has long been studied, it still remains one of the most challenging problems in the field. Most related papers are based on a generative learning framework, formulated as a Conditional Random Field (CRF), with priors on the piecewise-constancy of the reflectance map, the smoothly varying nature of the shading map [4], and relative reflectance prior learned through a data-driven approach [5]. Narihira *et al.* [116] recently presented a novel framework to perform the pixel-wise prediction of intrinsic images by relying on a fully convolutional neural network trained with synthetic dataset. However, due to the lack of sufficient non-synthetic training data, it does not show competitive results with real images.

5.2.2 Decomposition from RGB-D image or video

To tackle the intrinsic image decomposition task, several papers have attempted to consider additional cues such as 3D scene geometry [117–120] and temporal constraints [114, 115, 121] for processing RGB video data. They leverage these additional cues as means of further constraining the ill-posed image decomposition problem. Noisy depth maps have demonstrated utility for joint estimation of many intrinsic factors [117, 118]. Chen *et al.* [119] incorporate depth cues by separating the shading term further into direct irradiance, indirect irradiance, and illumination color maps. Surface normal has been also shown to improve the quality of resulting decompositions as incorporated in the shading prior. [120, 122]. Lee *et al.* [114] suggest novel constraints for estimating intrinsic image sequences from RGB-D video which take advantage of 3D geometry and temporal cues from RGB-D video [115]. An intrinsic video method [121] has been proposed to extract temporally coherent albedo and shading by exploiting optical flow. However, all these methods require an order of minutes making them inapplicable for real-time robotic systems.

5.2.3 Real-time Vision Processing

Several prior works have used a probabilistic framework to solve vision tasks in real-time. We drew on the rich body of literature in this domain to build our approach. Held *et al.* [123] propose a real-time point cloud segmentation method using a probabilistic framework. They partition the entire point clouds into smaller segments and make successive decisions whether to split each segment or merge multiple segments

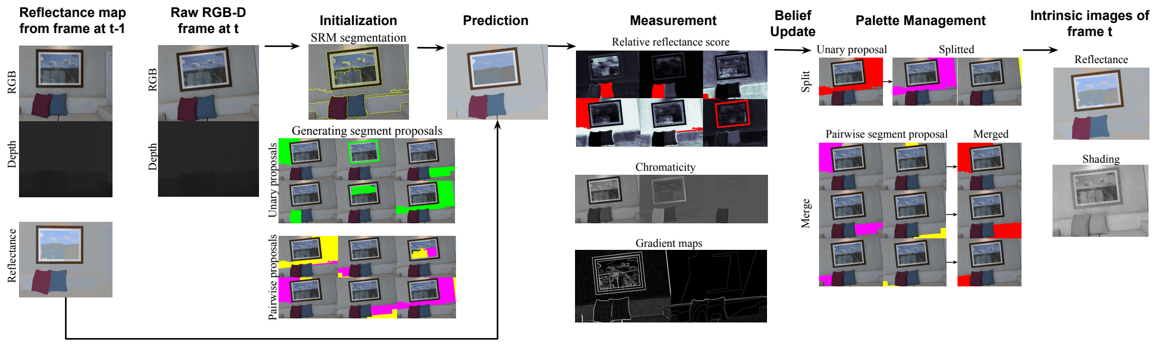


Figure 5.1: The pipeline of our proposed framework to estimate intrinsic images from RGB-D streams. The diagram summarizes each step with sample intermediate outputs during the interval $[t, t+1)$.

into a larger one. While our task is different from Held *et al.*, the formulation for our framework is motivated by this work. Miksik *et al.* [124] made pixel-wise predictions from streaming data for semantic scene understanding as opposed to the image decomposition problem being addressed here. They propagate redundant information across frames to increase both temporal consistency of the pixel-wise prediction and the efficiency of the proposed approach.

5.3 System Overview

This section presents an overview of the proposed pipeline. Figure 5.1 summarizes a single iteration of the proposed approach that generates intrinsic images from a stream of RGB-D frames. We follow a similar process to the Bayes filter [125] algorithm where the belief about a state is recursively calculated over time using measurement and control data.

At the start of the iteration during $[t, t+1)$, we are given the resulting intrinsic images from the previous frame $t-1$. Given a raw RGB-D frame at time t, we first initialize the framework by partitioning pixels into disjoint segments. Next we generate segment proposals that consist of either a single reflectance or a pair of distinct reflectance values. For each segment proposal, beliefs are being tracked over time. In the prediction step, we assign proper reflectance values to each segment by using the resulting reflectance map from the previous iteration. The beliefs are also transferred from previous segments to the corresponding segments at a current frame.

The next step in the pipeline obtains measurements for each segment proposal. The measurement is a new observation for each segment at time t, which will be used to update the belief in the subsequent step. We extract several features for both unary and pairwise segment proposals (see Section 5.4.6). Using computed measurement likelihoods, we finally update the beliefs for each segment proposal and determine whether to perform splitting and merging in a step we refer to as palette management (see Section 5.4.5). After repeating merging operations, we obtain the final reflectance map. As we process the image on the segment level instead of pixel by pixel, the reflectance image loses details, such as texture, and the shading map may contain severe discontinuities around segment boundaries. After passing through a simple post-processing step at the end of an iteration, we obtain the final intrinsic decomposition of an image. In the following sections, details of each step will be discussed.

5.4 Probabilistic Framework

The details about the problem formulation and the probabilistic framework are discussed in this section. The images presented in this section are taken from the living room dataset used in the ICL-NUIM RGB-D benchmark [126], which provides a sequence of RGB-D frames with the ground-truth camera trajectory.

5.4.1 State representation

At each frame, we observe an RGB image with the corresponding depth map. We group the image pixels into n disjoint subsets of distinct reflectance values. Let S_t denote a set of disjoint subsets that compose the frame at time t such that:

$$S_t = \{s_t^1, s_t^2, \dots s_t^n\} \tag{5.2}$$

where s_t^i represents the i^{th} reflectance segment in the frame at t. Note that pixels within a segment share the common reflectance. For each segment s_t^i, we define the state variable x_t^i and the measurement variable z_t^i. The state is a binary indicator where $x_t^i = 1$ if the pixels in s_t^i correspond to a single reflectance, and $x_t^i = 0$ if the pixels belong to more than one reflectance values. The measurement variable z_t^i is computed by extracting some features from the segment s_t^i.

5.4.2 Initialization

To obtain S_t we use the Statistical Region Merging (SRM) algorithm [127] which efficiently partitions an image into disjoint segments while preserving object bound-

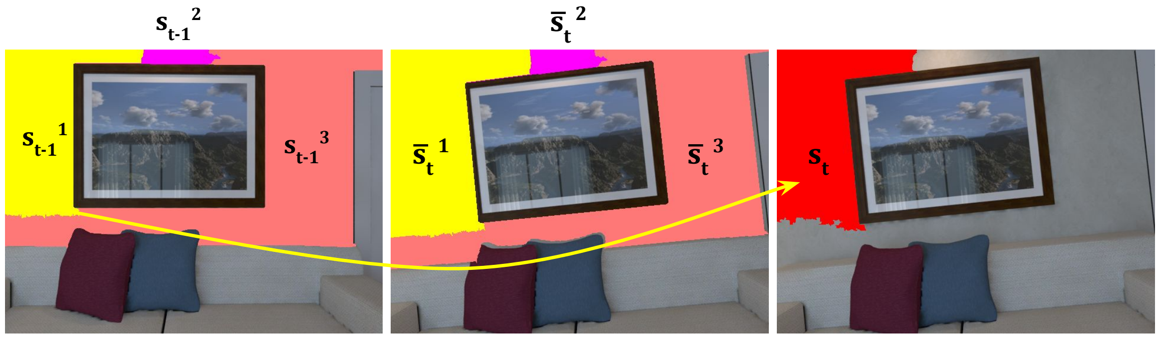


Figure 5.2: Temporal associations for s_t between two consecutive frames are found by computing intersection between s_t and each of previous segments $\bar{s}_t^1$, $\bar{s}_t^2$, and $\bar{s}_t^3$ in the warped frame.

aries. The obtained initial segments are refined through splitting and merging operations in later steps. For those operations, we generate segment proposals, each of which consists of either one or two initial segments. We named them unary and pairwise segment proposals. Note that splitting is a unary operation performed over a single segment, while the merging is a binary operator that requires at least two distinct segments.

Suppose the initial frame consists of n disjoint segments. All n segments are potential candidates for splitting, and all $O(2^n)$ combinations of those segments are candidates for merging. However, instead of passing through all combinatorial possibilities, which involves prohibitive computation, we only consider pairs of segments nearby as potential candidates. Additionally, we track the beliefs for all potential segment candidates to perform splitting and merging in later steps.

5.4.3 Prediction

The prediction step accumulates evidence from the past frames. With the Markov assumption, we propagate evidence at time t not from all the past frames but only from frame $t-1$. Given the frame at $t-1$ and the camera movement u_t during the interval $[t-1, t)$, we warp an image using a homography between the two frames. By similarly applying a homography to the resulting reflectance map from the previous frame, we obtain a reflectance map for the current frame. We denote the segments defined over the warped image as $\bar{s}_t$.

Let a_t indicate the temporal association of the segment s_t with all the segments in the previous frame. We assign $a_t^j = 1$ if the j^{th} segment of frame $t-1$ corresponds to the same reflectance region with the segment s_t, and $a_t^j = 0$ otherwise. In our framework, we simply compute the overlap between two segments s_t and $\overline{s}_t^j$ and determine the association a_t^j by thresholding it. Figure 5.2 describes how the temporal association vector is defined between two consecutive frames. The first image shows the segments in the previous frame at $t-1$, and the second image describes the predicted segments in the warped frame with respect to the camera pose at t. The third image shows one of the segments obtained from the initialization step at current frame t. Since s_t has sufficient overlap with $\overline{s}_t^1$ alone, we determined the first three components of the temporal association vector as $(a_t^1,\ a_t^2,\ a_t^3) = (1,\ 0,\ 0)$.

For a segment proposal s_t, the probability that s_t consists of a single reflectance region and is associated with the previous frame in terms of a_t is:

$$\begin{aligned}\overline{bel}(x_t, a_t) &= p(x_t, a_t | Z_{1:t-1}, u_{1:t}) && (5.3)\\ &= \textstyle\sum_{\forall x_{t-1}} p(x_t | a_t, x_{t-1}, Z_{1:t-1}, u_{1:t}) \\ &\quad\times\ p(a_t | x_{t-1}, Z_{1:t-1}, u_{1:t})\, p(x_{t-1} | Z_{1:t-1}, u_{1:t-1}) && (5.4)\\ &= \textstyle\sum_{\forall x_{t-1}} p(x_t | a_t, x_{t-1}, u_t)\, p(a_t)\, bel(x_{t-1}) && (5.5)\end{aligned}$$

where u_t represents camera movement, and Z_t is the set of measurements for all segments at frame t. We can interpret $\overline{bel}(x_t, a_t)$ as evidence transferred to s_t from the previous segments specified by a_t. The first term in (5.5) corresponds to the state transition probability, which maintains the temporal consistency of a segment across frames. In our implementation, we model this transition probability with a constant value for each of $x_{t-1} = 1$ and $x_{t-1} = 0$.

When a segment proposal s_t is judged to have a single reflectance value in the previous frame, it is highly likely that the segment consists of a single reflectance value in the current frame as well. So, we use a large constant as a transition probability for $x_t = 1$. On the other hand, when a segment is determined to have multiple reflectance values in the past frame, there will be a little chance of having a single reflectance value in the current frame. Therefore, we model a small transition probability for $x_t = 0$.

The remaining terms in (5.5) correspond to the prior beliefs about the data association and the previous segments. a_t does not depend on the knowledge from the previous frame, and we use a constant value for $p(a_t)$. The belief $bel(x_{t-1})$ indicates how likely it is for the segments to consist of either a single reflectance or more than one reflectance components. Beliefs are recursively estimated through the iterations which we discuss in detail in the following section.

5.4.4 Update

The update step aims to compute the belief on the number of distinct reflectance values. The belief is factorized as:

$$bel(x_t) = p(x_t | Z_{1:t}, u_{1:t}) = p(x_t | z_t, Z_{1:t-1}, u_{1:t}) \tag{5.6}$$

$$= \eta \sum_{\forall a_t} p(z_t | x_t, a_t, Z_{1:t-1}, u_{1:t}) \, \overline{bel}(x_t, a_t) \tag{5.7}$$

where the first term in the summation of (5.7) corresponds to the measurement model which is discussed in Section 5.4.6, and the second term is computed in the prediction step as in (5.5). When transitioning from (5.6) to (5.7), the data association a_t is introduced which relates the past frame with the current segment proposal. Since a segment proposal s_t is not associated with all segments in the previous frame but only connects a few, introducing a_t simplifies the computation. We sum over all 2^n associations where the previous frame contains n final segments. Fortunately, segment proposals are associated with at most a few or none of the previous segments. In other words, $p(x_t, a_t | Z_{1:t-1}, u_{1:t})$ for most associations is close to zero so that it can be ignored to efficiently compute the belief.

5.4.5 Palette Management

Given the updated beliefs, we can predict the state of each segment proposal: $x_t = 1$ if $bel(x_t) > 0.5$ and $x_t = 0$ otherwise. Based on those determined states for all segment proposals, splitting or merging operations are applied to refine segments and finally obtain intrinsic images. When $x_t = 1$, we do nothing if s_t consists of a single reflectance segment, or if s_t has multiple segments we merge them. For the segment proposal where $x_t = 0$, we further split such segments if s_t consists of a single reflectance segment, or do nothing if s_t already consists of a pair of distinct reflectance segments. At the end of the palette management step, we go through post-processing where the resulting reflectance image has its detail restored and the shading image is smoothed to satisfy the shading continuity constraint. The reflectance values in the palette are also updated when necessary.

5.4.5.1 Splitting

The splitting is performed on segments where $x_t = 0$, which implies the pixels in the segment are more likely to belong to more than one reflectance values. Figure 5.3 describes the way splitting works on a unary segment proposal. The two left images

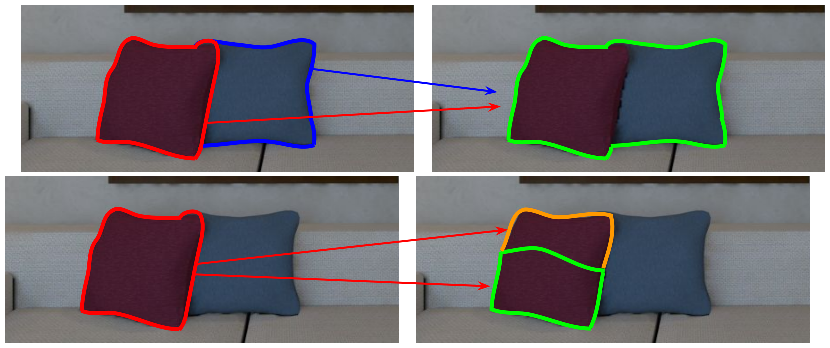


Figure 5.3: Splitting and merging. Two consecutive frames are connected with temporal associations.

correspond to the frame at $t-1$ and t. Two cushions have distinct colors but suppose they are assigned to be one segment s_t during the initialization step. The belief of the temporally associated segments s^r_{t-1} and s^b_{t-1} in the frame $t-1$ are propagated to s_t. If $bel(x_t) < 0.5$, we split s_t into two disjoint segments. When splitting the segment s_t, we project the matching segments s^r_{t-1} and s^b_{t-1} in the previous frame onto the current frame, which we denote as $\overline{s}^r_t$ and $\overline{s}^b_t$. The overlapping region between the projected segment and s_t defines a new segment.

5.4.5.2 Merging

In the merging step, we find segments that actually share the same reflectance value and merge these segments so that the total number of segments in the frame does not grow too much. Figure 5.3 shows how the merging operation is applied to the pairwise segment proposal. The right two images correspond to the frame at $t-1$ and t for each. Suppose pixels within the red cushion in the frame t are grouped into two disjoint segments during the initialization step. This is clearly not the desirable result. To decide whether to merge these two segments, say s^1_t and s^2_t, we consider the temporal associations with the previous frame. Red arrows indicate that both segments s^1_t and s^2_t share the common association with the segment s_{t-1} in the previous frame, and the belief on s_{t-1} is transferred to the pairwise segment proposal $s^{1,2}_t$ in the current frame. In the given example, it is highly likely that $p(x^{1,2}_t|Z_{1:t-1}, u_{1:t-1}) > 0.5$. That means the segment proposal $s^{1,2}_t$ which consists of two distinct segments s^1_t and s^2_t has high chance of being composed of a single reflectance value. In such a case, we merge the two segments. We repeat this merging operation over all the pairwise segment proposals until there are no more proposals to be merged.

5.4.5.3 Post-processing of shading map

As described in (5.1), we can obtain a shading map by dividing each pixel of an RGB image into the corresponding predicted reflectance. Here, we assume that the shading is grayscale.

All the operations in our framework are performed at the segment-level, so the resulting reflectance may lose much of its high frequency detail, including the texture of object surfaces. The shading map may also contain unwanted discontinuities with artifacts around the segment boundaries that violates the underlying assumption that shading maps should vary smoothly [4]. To make our shading map satisfy this constraint, we perform basic smoothing on the shading map, particularly for the pixels around the segment boundaries.

5.4.5.4 Palette adjustment

The palette is the set of distinct reflectance values where we store all the reflectance values seen in the reflectance image. We also maintain a mapping between each reflectance and its corresponding RGB pixels so the palette can function as a lookup table for initializing new segments. For example, when a segment proposal s_t has a temporal association with no previous segments, we find the element from the palette that has the closest chromaticity to that of s_t and use its reflectance as our initial estimate.

However, there still exists a possibility that a new object will emerge in a scene with a completely new reflectance never seen in the palette. In such a case, incorrect reflectance values could be being propagated in future frames. To avoid this, we threshold the distance of new observations and the palette RGB colors. For values exceeding the threshold, new reflectance values are added to the palette.

5.4.6 Measurement Model

To update the belief in (5.7), we compute the measurement likelihood $p(z_t|x_t, a_t, Z_{1:t-1})$ for each segment proposal. The following are the assumptions that motivated our measurement model:

- Piecewise-constancy of a reflectance map [128].
- Nearby pixels which have similar chromaticity and intensity being more likely to have similar reflectance [4].
- Relative reflectance prior learned in a data-driven way [129].

Our measurement model is defined based on the above properties.

5.4.6.1 Chromaticity prior

We consider the variance of chromaticity values within a segment. The probability for a segment with state x_t to have z_t as a measurement is defined as:

$$p(z_t|x_t) = \frac{1}{\lambda} \exp\left(-\frac{1}{\lambda} \textstyle\sum_c \sigma_c^2/3 \right) \tag{5.8}$$

where σ_c^2 is a variance of the chromaticity channel $c \in \{red, green, intensity\}$, and the parameter $\lambda \in \{\lambda_1, \lambda_0\}$ defines the distribution for each state of x_t. For segment proposals that consist of multiple reflectance regions, the variance of chromaticity would be large. On the other hand, it is highly probable that the variance is small for the segment that consists of a single reflectance region.

5.4.6.2 Gradient prior

We obtain the gradient maps from the grayscaled input image and the depth image, and then sum the gradients over the pixels within the segment region. A single reflectance region usually has a small sum of intensity gradients and have no strong edges across the segment. A large sum of gradients will indicate that the segment consists of multiple reflectance regions. We modeled the probability of a segment with state x_t having z_t as a measurement as:

$$p(z_t|x_t) = \frac{1}{\lambda} \exp\left(-\frac{1}{\lambda} \sum_{p \in s_t} |\nabla I(p)| \,/\, |s_t| \right) \tag{5.9}$$

where $\lambda \in \{\lambda_1, \lambda_0\}$ is a parameter that defines the distribution for each state of x_t, $|s_t|$ is the total number of pixel in the segment proposal s_t, and $|\nabla I(p)|$ is the magnitude of the image gradient at pixel p. Increasing λ makes the distribution decay slow.

5.4.6.3 Relative reflectance score

Zhou *et al.* [5] trained a relative reflectance classifier using a large-scale human-annotated data set [4]. Their proposed two-stage network extracts both local and context features from the image and returns relative reflectance scores for every pixel from a certain query point. The extracted score quantifies the reflectance similarity between two pixel points. Figure 5.4 shows the resulting scoremaps obtained at various query points.

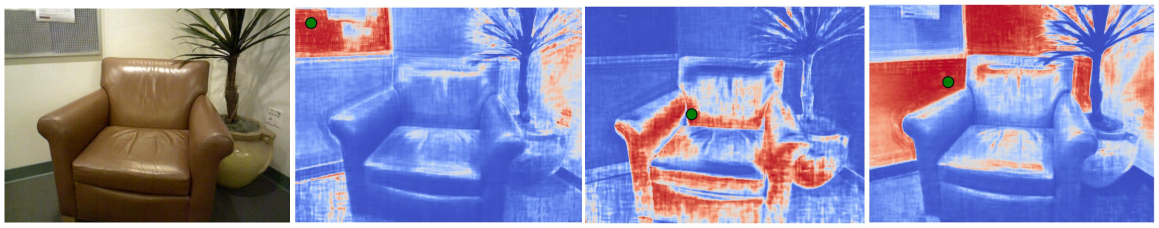


Figure 5.4: Scoremaps visualize pixels that likely have same reflectance as a certain query point. Query points are marked as green circles. Red pixels indicate high scores while the blue corresponds to the pixels with low scores.

For a pairwise segment proposal x_t, we use this relative reflectance score as a measurement z_t. We define the measurement likelihood as:

$$p(z_t|x_t) = \eta \ \exp\left(-\frac{(z_t-\mu)^2}{2\sigma^2}\right) \tag{5.10}$$

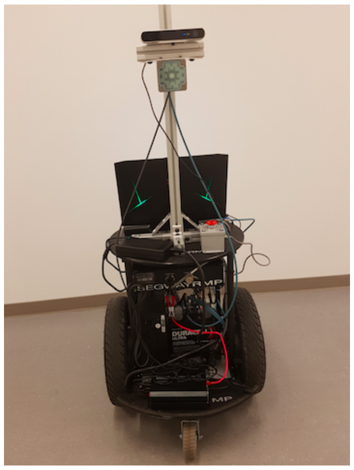
where η is a normalization constant and μ and σ are the statistics of the weighted score values computed for $x_t = 1$ and $x_t = 0$ using the large-scale relative reflectance dataset [4]. For a unary segment proposal, we find the relative scores between two pixels within the segment and use the lowest score in (5.10) to compute measurement likelihoods.

5.4.7 Processing the Initial Frame

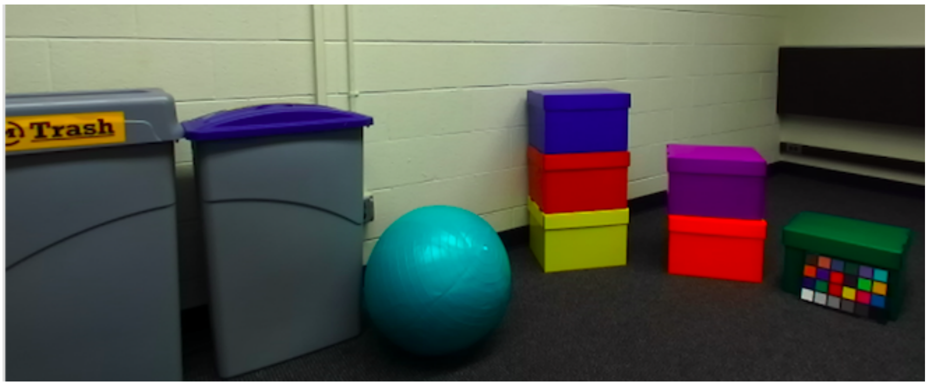
When propagating the beliefs across frames, the underlying assumption is that the past frames have been processed correctly, so that the evidence accumulated over time is reliable. We have experimented with different initial frames to see how the quality of the initial frame affects the prediction results. The results are presented in Section 5.5.

5.5 Experiments

In this section, we present the details of our results in comparison to the literature methods. First, we describe the real-world experiments with our framework. Then we report the quantitative evaluation of the resulting intrinsic images, and discuss temporal consistency and the efficiency of our system. At the end, we show potential of our framework to be applied for VR applications with 3D intrinsic model reconstruction.



(a)



(b)

Figure 5.5: (a) The Segway platform equipped with sensors and (b) the objects with distinctive colors.

5.5.1 Experiments with Real-world Scene

5.5.1.1 Platform

To verify the capability of our system to process real-world scene images, we configured the scene of simple objects with distinctive colors under normal indoor lighting. Such scenes with a wide variety of colors provide abundant options during the palette management step. Image sequences are captured by using a Segway platform which has cameras attached and provides stable and controlled motion during the data acquisition. For the experiments, we used a monocular RGB-D camera to acquire images. The objects and the platform can be seen in Figure 5.5.

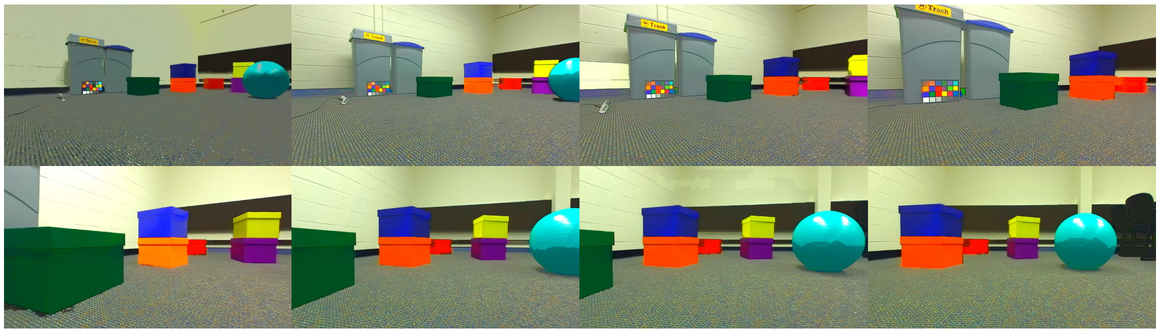


Figure 5.6: Resulting reflectance images.

5.5.1.2 Results

To extend our pipeline as a complete real-world platform, we integrated our framework into ElasticFusion [130], which estimates camera pose and reconstructs a global 3D model. The estimated camera pose by ElasticFusion is fed into our framework. Figure 5.6 shows the decomposed reflectance images.

The resulting images look temporally consistent, and the reflectance values seem desirable given the known objects. However, the reflective material of the object surface caused specular reflections, and the direct light source located at a close distance hampered obtaining the ideal reflectance images.

5.5.2 Quantitative Evaluation

There is no standard measure to evaluate the performance of the resulting intrinsic images from an image sequence due to the lack of ground truth benchmark datasets. Fortunately, the ICL-NUIM dataset provides the 3D model of a livingroom scene with rendering details. After properly controlling the properties of the scene, we generated the ground truth reflectance images by rendering the model. (see Figure 5.11)

In Table 5.1, we report three different measures used by Chen *et al.* [119]: the standard mean-squared error (MSE), the local mean-squared error (LMSE) [131], and the structural dissimilarity index (DSSIM) [132]. We used images from the subset of ICL-NUIM lr_kt1 dataset, which consists of 965 RGB-D frames. For the evaluation, we excluded the frames from index 233 to 576 because they include fully-transparent surfaces, which are outside the scope of any of the proposed techniques in the literature. Compared to the baseline methods, our framework produces the smallest error across all three measures. The reason for better performance of our framework may be related to the piecewise-constancy prior of the reflectance image.

Table 5.1: Per-frame quantitative evaluation of the reflectance maps predicted by different methods on the subset of ICL-NUIM dataset.

	MSE	LMSE	DSSIM
Bell *et al.* [4]	.01597	.006927	.08270
Zhou *et al.* [5]	.01942	.009483	.09271
Ours (init by ground truth)	**.009792**	**.006148**	**.04780**
Ours (init by Bell et al.)	.01168	.006325	.04864
Ours (w/o initial reflectance map)	.01163	.006405	.04900

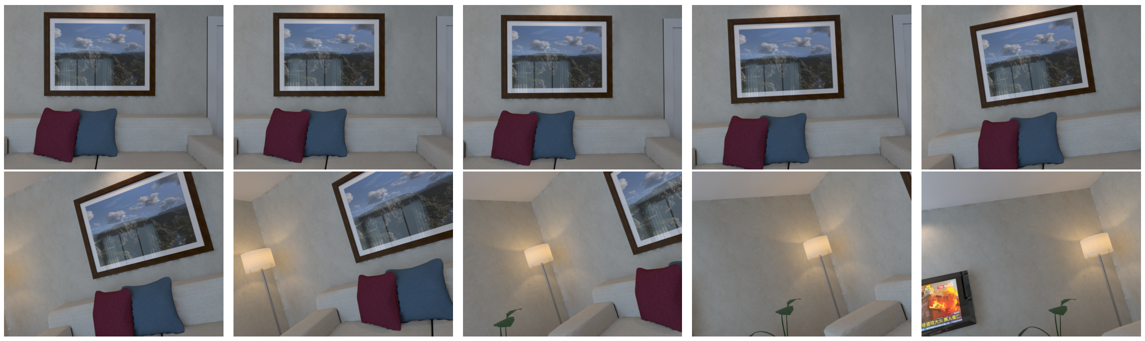


Figure 5.7: RGB images from ICL-NUIM dataset.

As we predict reflectance images at a segment level, our framework may better satisfy the piecewise-constancy constraint described by Barron *et al.* [128] especially when compared to pixel-wise prediction methods.

5.5.3 Temporal Consistency

Figures 5.8, 5.9, and 5.10 show the resulting reflectance images obtained using different methods. While the frame-based methods show inconsistent reflectance images over time, our framework demonstrates great consistency across frames. This is true particularly in Figure 5.9, where the approach Zhou *et al.* [5] proposes relies mostly on the relative reflectance prior between each pair of pixels in an image and as such the absolute values of reflectance shows a significant fluctuations over time.

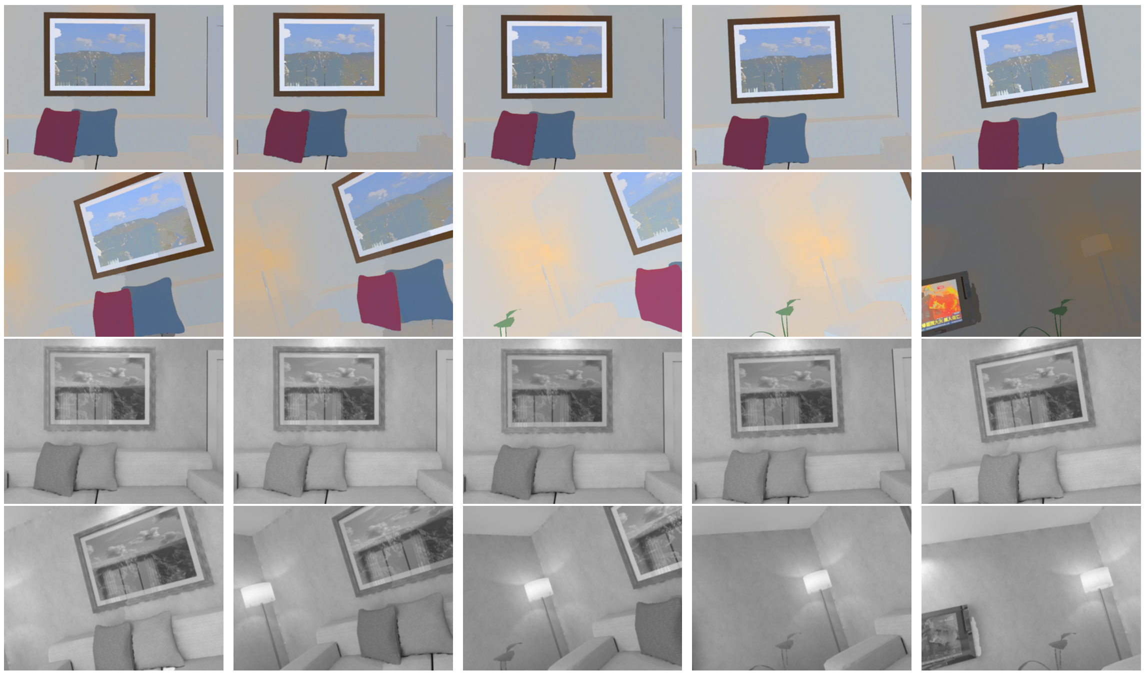


Figure 5.8: Predicted reflectance and shading images using Bell *et al.* [4].

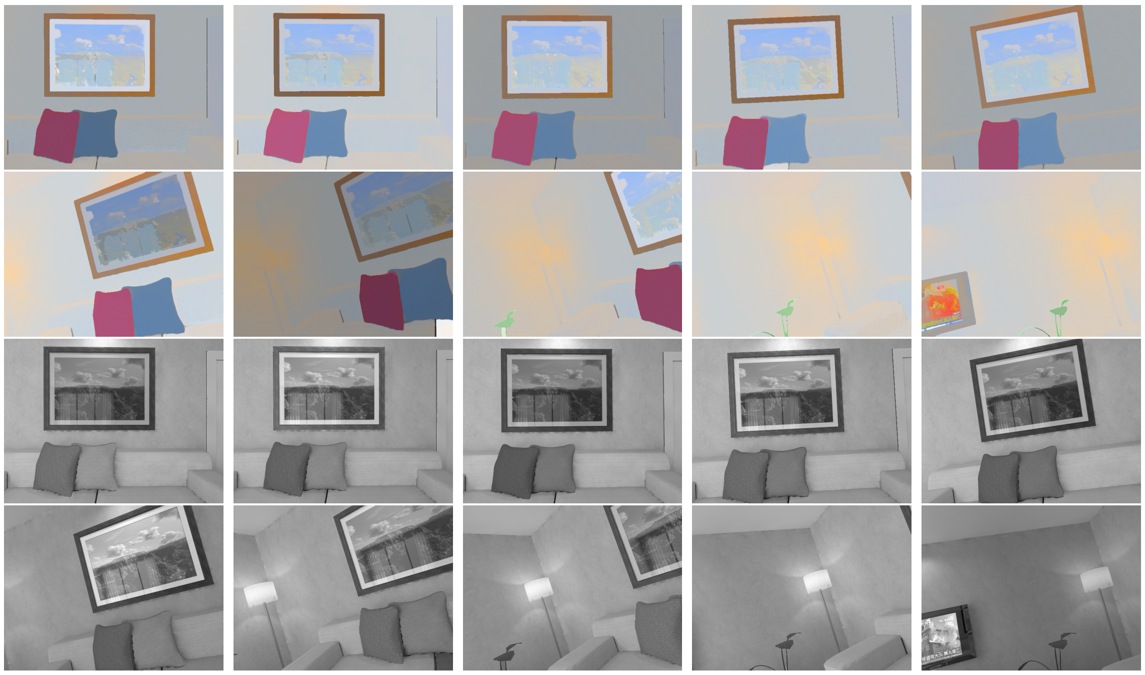


Figure 5.9: Predicted reflectance and shading images using Zhou *et al.* [5].

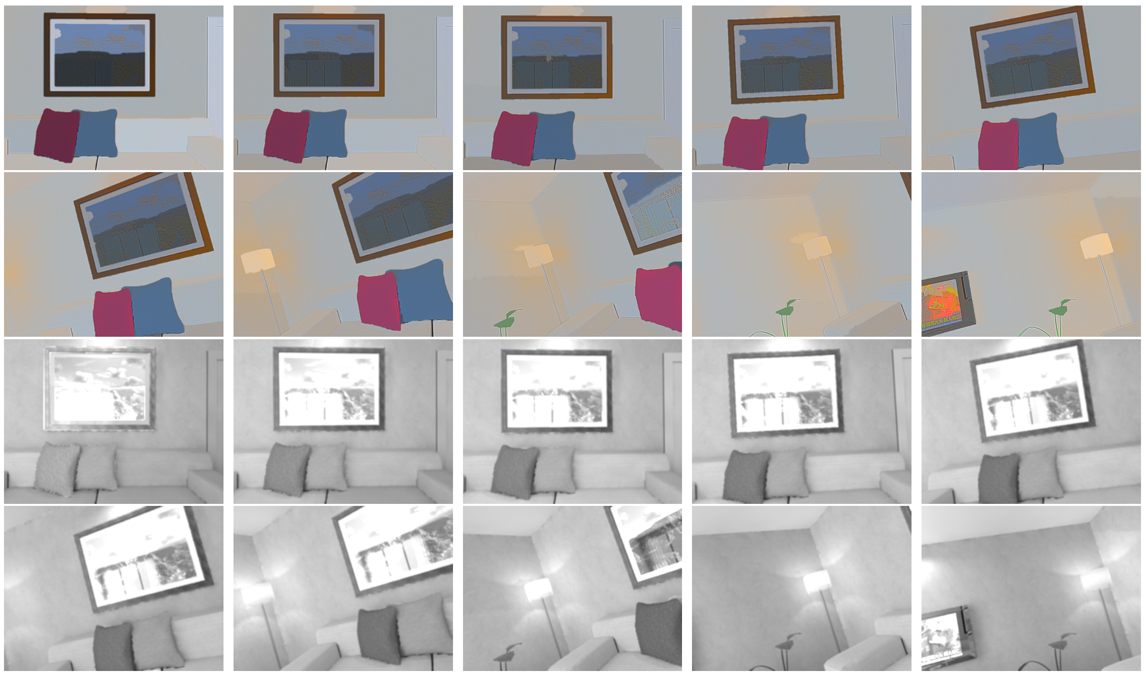


Figure 5.10: Predicted reflectance and shading images using our framework.

Table 5.2: Running time for processing a single frame.

	Bell *et al.*	Zhou *et al.*	Ours
Time (s)	89.47	13.88	1.5-3.5

5.5.4 Efficiency

None of the baseline methods are designed to work with streaming data. To compare the running time of different approaches, we measure the average running time it takes to process a single frame. The experiments were performed using a GeForce GTX TITAN X GPU and an Intel i7-4790K CPU. We are currently using a GPU for extracting high-dimensional features from an image and computing the relative reflectance prior. No other parallelization or optimization have been implemented yet. Table 5.2 shows the elapsed time for different methods to process a single frame in VGA resolution.

5.5.5 3D Intrinsic Models

Based on the predicted reflectance images with corresponding depth maps, we reconstructed a 3D reflectance model using the ElasticFusion [130] dense SLAM system.

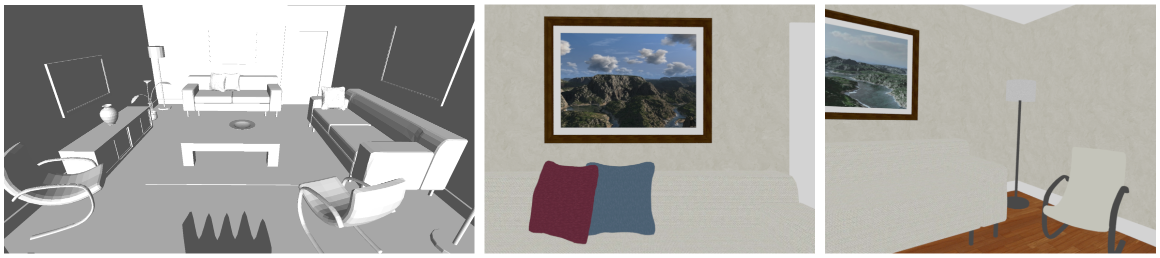


Figure 5.11: The ground truth reflectance images obtained by rendering the 3D scene model with known reflectance values.

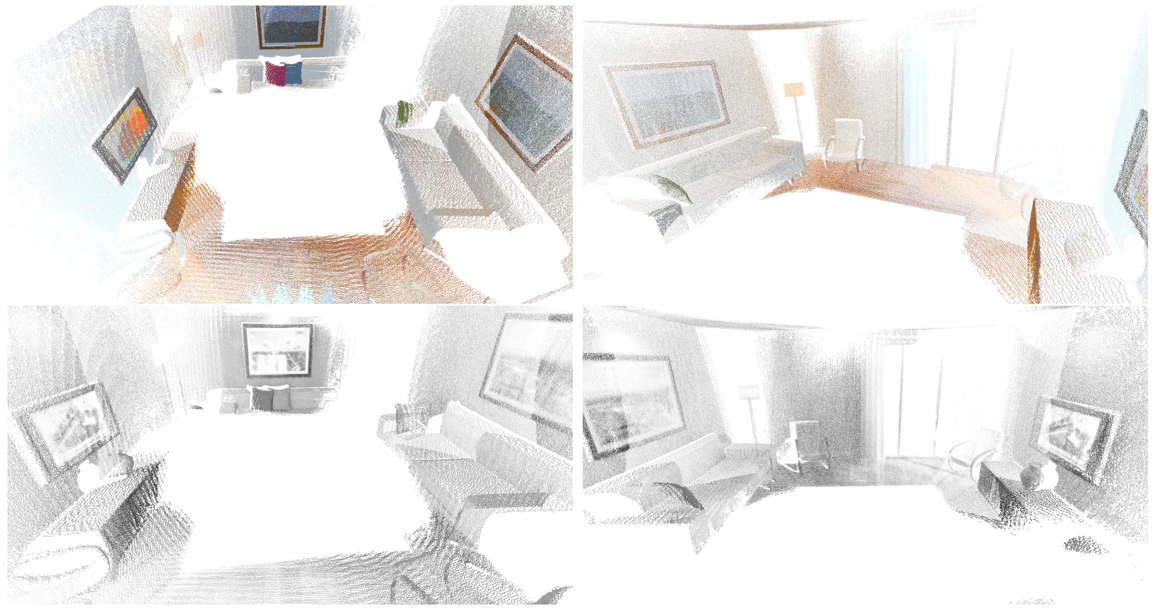


Figure 5.12: The reconstructed 3D reflectance and shading models of the living room scene from ICL-NUIM benchmark.

These models contain the lighting and shading information for full 3D scenes. Figure 5.12 shows the reconstructed 3D reflectance and shading models. To demonstrate the power of this kind of data we show an example of adding a virtual 3D object to generate the novel 3D scene. The point cloud is first converted to the triangular mesh using the Poisson surface reconstruction. Then to relight the reconstructed scene, a synthetic light is placed into the scene with varying intensities. Figure 5.14 shows the rendering results. We see this as one of the major applications of this work. Augment reality models can be given much higher degrees of realism using models with matching lighting and shading and the ability to map the lighting and material properties of a 3D scene has application for segmentation and object classification.

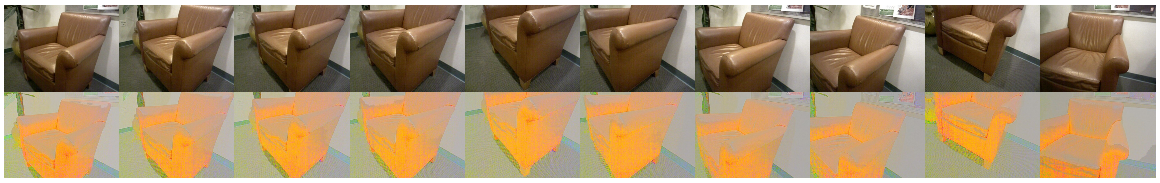


Figure 5.13: Resulting reflectance images (bottom) from using our framework with the Scene3D dataset (top).

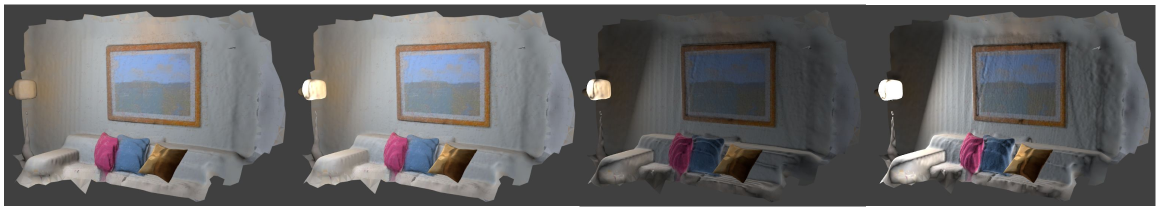


Figure 5.14: Rendered images of a scene with a virtual object added. The scene is relighted by synthetic shading effects under different lighting conditions.

5.6 Conclusion

This chapter presented an intrinsic image prediction system from an RGB-D stream using a probabilistic framework. The significant advantage of our framework is that it significantly reduces the amount of computation and running time to produce an image decomposition by taking advantage of information already known from past frames. It also maintains temporal consistency over time, which is not the case for frame-based methods. We reduce the running time of the state-of-the-art by an order of magnitude. Finally, our framework to predict intrinsic images was integrated into a real-time SLAM system to reconstruct 3D intrinsic models of a scene, which has many exciting applications for object classification, augmented reality, 3D rendering, and scene understanding.

This work suggests a direction for improving the densely reconstructed scene models in Chapter IV. From the decomposed intrinsic images, we can infer the objects' intrinsic properties in a scene. Based on the 3D intrinsic models, we can obtain realistically textured reconstruction of a scene, and color images can be rendered to generate simulated camera images.

Nevertheless, many challenges still exist. First, urban intersections are outdoor scenes, so we might need different formulation of the intrinsic image decomposition for outdoor images. From rigid objects, such as vehicles, traffic signs, lampposts, and buildings, to non-rigid objects, such as pedestrians and trees, many different types of complex objects exist in urban intersections at different scales. Many other intrinsic properties need to be considered in addition to reflectance and shading due to the hard shading, natural lighting, and effects of reflections on specular surfaces.

CHAPTER VI

Conclusion and Future Directions

6.1 Conclusions

This dissertation addressed key challenges relevant to the perception of autonomous vehicles in urban traffic environments and contributed to several components, from data collection and labeling to 3D reconstruction. We have answered several research questions with the following contributions:

1. **How can we obtain meaningful data from real urban traffic scenes?**
   In the first part of the dissertation (Chapter II), we explored the importance and challenges of acquiring real-world urban traffic data and explained our sensor system to capture data with meaningful events. The data collected at unsignalized urban intersections using the AV sensor system contains natural motions and behaviors of pedestrians and vehicles with multi-modal sensor data.

2. **How can we efficiently process the raw data to produce meaningful labels for the dynamic road agents?**
   The second part of the dissertation (Chapter III) addressed the efficient labeling of pedestrians based on 3D model fitting. The model-fitting-based pedestrian labeling method provides a rich set of annotations and suggests a pipeline to automatically label a large amount of multi-modal data.

3. **How can we augment the data to entire traffic scenes?**
   The third part of the dissertation (Chapter IV) presented the dense 3D reconstruction of the entire urban traffic scenes, including various types of road agents and static objects. The model-based dense 3D reconstruction of entire traffic scenes, including pedestrians, vehicles, and static scene components, can

be utilized to generate a large set of AV simulated realistic data with free labels. Experiments, such as simulating virtual sensors and predicting ego-vehicle trajectories, validate the quality of the simulated data and suggest interesting applications.

4. **What factors can make the reconstruction more realistic?**
   In the last part of the dissertation (Chapter V), we explored one of the factors to achieve photo-realistic reconstruction. The intrinsic image decomposition problem was tackled by taking advantage of the probabilistic framework.

6.2 Future Work

The overall subject of this dissertation is to understand the AV perception pipeline and explore data preparations at urban traffic scenes, which is the key prerequisite to solving various other problems. Future work should focus on further augmenting the data and developing diverse applications using the prepared data and labels.

6.2.1 Trajectory Augmentation

Using the reconstructed 3D scene models from Chapter III and Chapter IV, we can generate as much data as possible by configuring arbitrary specs of cameras at different locations. While the simulated data helps supplement the real captured data when there is an insufficient diversity of sensor perspectives and specs, it is not trivial to simulate trajectories of new road agents that have not been observed before. The number of agents and their trajectories cannot be easily augmented due to the complex interactions between agents. The key to trajectory augmentation is adding new trajectories consistent with all the other existing trajectories in a scene. A trajectory prediction model can be learned at the intersections to augment the road agents' trajectory data. The typical trajectory prediction problem aims to predict future trajectories given past observations. In order to augment the realistic trajectories, the problem can be formulated in a slightly different setting. To realistically add a new road agent to the scene, all the available information – the future trajectories and the past trajectories – can be fed into the recurrent networks. Developing this recurrent model that produces new trajectories consistent with all the other observed trajectories can be one potential research topic.

6.2.2 Textured Reconstruction

As a way to overcome the limitation of the texture-less scene models reconstructed in Chapter IV, we discussed in Chapter V to infer the objects' intrinsic properties and apply them to simulate camera data more realistically. However, estimating intrinsic properties from RGB images is very challenging, particularly for complex urban scenes. As an alternative for using the objects' actual intrinsic values, assigning arbitrary intrinsic values to each object might be sufficient to generate simulated data. Once intrinsic values are configured for all the objects in a scene, we can generate the more realistic simulated data, which we propose as a potential future direction.

To be specific, the scene models of urban intersections consist of rigid objects such as vehicles, traffic signs, lampposts, and buildings and non-rigid objects such as pedestrians and trees. It is simple to assign arbitrary reflectance values to the rigid objects in the intersections because few distinct values typically comprise the objects. Whereas for the non-rigid objects, especially pedestrians, realistically assigning reflectance values is difficult due to the variations and complexity in appearance. Moreover, the pedestrian mesh models currently being used in Chapter III and Chapter IV are not dressed or minimally clothed, which will produce somewhat unrealistic rendering results. Based on recent progress to generate clothed human mesh models [133], future research should further explore to use of realistic mesh models, particularly for non-rigid objects.

6.2.3 Occlusion-Aware Perception

AV pipelines generally involve various perception algorithms, from object detection and scene understanding to tracking and future prediction. In urban driving scenarios, occlusions frequently occur in many parts of a scene due to the scene clutter and the driver's limited field of view, making perception algorithms fail or degrade the performance. Occlusions can also cause safety issues when pedestrians are severely occluded by surroundings or utterly invisible to drivers. Handling occlusions is one of the critical factors to advance perception algorithms and enhance AVs' safety.

Some prior work addressed the importance of handling occlusions. The concept of vislet was introduced to consider limited view frustum of actual pedestrians, and the mixture of tracklets and vislets were used to learn recurrent models to predict future pedestrian trajectories [95]. Handling occlusions in terms of vislets resulted in better performance in predicting future trajectories. First-person view images were used along with observed trajectories to perform the trajectory prediction [96].

Although the synthetically rendered images were used, visual information from first-person views was shown to be effective in improving the performance of pedestrian trajectory prediction.

One noticeable feature of our data is that it was captured from a driver's perspective. This first-person perspective data realistically mimics actual drivers' views, including diverse occlusion patterns. Partially occluded LiDAR points and the captured RGB images containing object appearances in first-person views can be used to extract visual features for occlusion-aware perception algorithms. While our raw captured data involves occlusions, the augmented data generated from the reconstructed 3D models provides full annotations. Therefore, our data can be also used to recover or infer the actual 3D scene information from partial observations.

BIBLIOGRAPHY

[1] J. Halkia and J. Colyar, "Interstate 80 freeway dataset," *Federal Highway Administration, US Department of Transportation*, 2006.

[2] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1929–1934, IEEE, 2019.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

[4] S. Bell, K. Bala, and N. Snavely, "Intrinsic images in the wild," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 159, 2014.

[5] T. Zhou, P. Krähenbühl, and A. A. Efros, "Learning data-driven reflectance priors for intrinsic image decomposition," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3469–3477, 2015.

[6] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[7] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz, *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, p. 113816, 2020.

[8] R. Gopalan, T. Hong, M. Shneier, and R. Chellappa, "A learning approach towards detection and tracking of lane markings," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1088–1098, 2012.

[9] J. Kim and M. Lee, "Robust lane detection based on convolutional neural network and random sample consensus," in *International conference on neural information processing*, pp. 454–461, Springer, 2014.

[10] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 286–291, IEEE, 2018.

[11] R. V. Carneiro, R. C. Nascimento, R. Guidolini, V. B. Cardoso, T. Oliveira-Santos, C. Badue, and A. F. De Souza, “Mapping road lanes using laser remission and deep neural networks,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[12] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, “Vision for looking at traffic lights: Issues, survey, and perspectives,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1800–1815, 2016.

[13] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, “Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012.

[14] A. Gudigar, S. Chokkadi, and U. Raghavendra, “A review on automatic detection and recognition of traffic sign,” *Multimedia Tools and Applications*, vol. 75, no. 1, pp. 333–364, 2016.

[15] A. Breuer, J.-A. Termöhlen, S. Homoceanu, and T. Fingscheidt, “opendd: A large-scale roundabout drone dataset,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2020.

[16] A. Zyner, S. Worrall, and E. M. Nebot, “Acfr five roundabouts dataset: Naturalistic driving at unsignalized intersections,” *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 8–18, 2019.

[17] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, “INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps,” *arXiv:1910.03088 [cs, eess]*, 2019.

[18] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, “Learning social etiquette: Human trajectory prediction in crowded scenes,” in *European Conference on Computer Vision (ECCV)*, 2020.

[19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[20] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, “The apolloscape dataset for autonomous driving,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[21] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 Year, 1000km: The Oxford RobotCar Dataset,” *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[22] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes," in *International Conference on Robotics and Automation*, 2019.

[24] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. 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*, pp. 2446–2454, 2020.

[25] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 746–753, 2017.

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[27] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[28] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.

[29] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[30] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[31] J.-Y. Bouguet, "Matlab camera calibration toolbox," *Caltech Technical Report*, 2000.

[32] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 257–266, 2002.

[33] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2017.

[34] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," *ACM Transactions on Graphics*, vol. 36, no. 4, p. 44, 2017.

[35] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3d human pose estimation in the wild: a weakly-supervised approach," in *International Conference on Computer Vision*, IEEE, 2017.

[36] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis, "Sparseness meets deepness: 3d human pose estimation from monocular video," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2016.

[37] C.-H. Chen and D. Ramanan, "3d human pose estimation= 2d pose estimation+ matching," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2017.

[38] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in *International Conference on Computer Vision*, IEEE, 2017.

[39] L. Sigal, A. O. Balan, and M. J. Black, "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International Journal of Computer Vision*, vol. 87, no. 1, pp. 4–27, 2010.

[40] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[41] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *European Conference on Computer Vision*, IEEE, 2016.

[42] A.-I. Popa, M. Zanfir, and C. Sminchisescu, "Deep multitask architecture for integrated 2d and 3d human sensing," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2017.

[43] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *International Conference on 3D Vision*, IEEE, 2017.

[44] D. Tome, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3d pose estimation from a single image," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2017.

[45] V. Ramakrishna, T. Kanade, and Y. Sheikh, "Reconstructing 3d human pose from 2d image landmarks," in *European Conference on Computer Vision*, Springer, 2012.

[46] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image," in *European Conference on Computer Vision*, Springer, 2016.

[47] I. Akhter and M. J. Black, "Pose-conditioned joint angle limits for 3d human pose reconstruction," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2015.

[48] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM Transactions on Graphics*, vol. 34, no. 6, p. 248, 2015.

[49] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. J. Black, and P. V. Gehler, "Unite the people: Closing the loop between 3d and 2d human representations," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2017.

[50] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2018.

[51] R. Alp Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2018.

[52] K. Rematas, I. Kemelmacher-Shlizerman, B. Curless, and S. Seitz, "Soccer on your tabletop," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2018.

[53] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, "Single-shot multi-person 3d pose estimation from monocular rgb," in *International Conference on 3D Vision*, IEEE, 2018.

[54] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Berkeley mhad: A comprehensive multimodal human action database," in *Workshop on Applications of Computer Vision*, IEEE, 2013.

[55] "Cmu mocap database." [Online; accessed 17-March-2021].

[56] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *International Conference on 3D Vision*, IEEE, 2017.

[57] "The captury." [Online; accessed 17-March-2021].

[58] T. von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll, "Sparse inertial poser: Automatic 3d human pose estimation from sparse imus," *Computer Graphics Forum*, vol. 36, no. 2, pp. 349–360, 2017.

[59] T. von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3d human pose in the wild using imus and a moving camera," in *European Conference on Computer Vision*, Springer, 2018.

[60] M. F. Ghezelghieh, R. Kasturi, and S. Sarkar, "Learning camera viewpoint using cnn to improve 3d body pose estimation," in *International Conference on 3D Vision*, IEEE, 2016.

[61] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, "Synthesizing training images for boosting human 3d pose estimation," in *International Conference on 3D Vision*, IEEE, 2016.

[62] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2017.

[63] G. Rogez and C. Schmid, "Mocap-guided data augmentation for 3d pose estimation in the wild," in *Advances in Neural Information Processing Systems*, pp. 3108–3116, 2016.

[64] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Harvesting multiple views for marker-less 3d human pose annotations," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2017.

[65] Y. Huang, F. Bogo, C. Classner, A. Kanazawa, P. V. Gehler, I. Akhter, and M. J. Black, "Towards accurate markerless human shape and pose estimation over time," in *International Conference on 3D Vision*, IEEE, 2017.

[66] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[67] "Bvh conversions of the 2500-motion carnegie-mellon motion capture dataset." [Online; accessed 17-March-2021].

[68] W. contributors, "Rodrigues' rotation formula — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Rodrigues\%27_rotation_formula&oldid=1008666891`, 2021. [Online; accessed 17-March-2021].

[69] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *International Conference on Computer Vision*, IEEE, 2017.

[70] D. Yang, L. Li, K. Redmill, and Ü. Özgüner, "Top-view trajectories: A pedestrian dataset of vehicle-crowd interaction from controlled experiments and crowded campus," *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 899–904, 2019.

[71] A. Breuer, J.-A. Termöhlen, S. Homoceanu, and T. Fingscheidt, "opendd: A large-scale roundabout drone dataset," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2020.

[72] A. Zyner, S. Worrall, and E. M. Nebot, "Acfr five roundabouts dataset: Naturalistic driving at unsignalized intersections," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 8–18, 2019.

[73] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps," *arXiv:1910.03088 [cs, eess]*, 2019.

[74] W. Kim, M. S. Ramanagopal, C. Barto, M.-Y. Yu, K. Rosaen, N. Goumas, R. Vasudevan, and M. Johnson-Roberson, "Pedx: Benchmark dataset for metric 3-d pose estimation of pedestrians in complex urban intersections," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1940–1947, 2019.

[75] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtualworlds as proxy for multi-object tracking analysis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4340–4349, 2016.

[76] J. Zolfaghari Bengar, A. Gonzalez-Garcia, G. Villalonga, B. Raducanu, H. H. Aghdam, M. Mozerov, A. M. Lopez, and J. van de Weijer, "Temporal coherence for active learning in videos," in *The IEEE International Conference in Computer Vision, Workshops (ICCV Workshops)*, 2019.

[77] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[78] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[79] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset," in *CVPR Workshop on The Future of Datasets in Vision*, 2015.

[80] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 961–971, 2016.

[81] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2255–2264, 2018.

[82] H. Xue, D. Q. Huynh, and M. Reynolds, "Ss-lstm: a hierarchical lstm model for pedestrian trajectory prediction," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1186–1194, IEEE, 2018.

[83] S. Haddad, M. Wu, H. Wei, and S. K. Lam, "Situation-aware pedestrian trajectory prediction with spatio-temporal attention model," *arXiv preprint arXiv:1902.05437*, 2019.

[84] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, IEEE, 2018.

[85] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6120–6127, 2019.

[86] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," in *Advances in Neural Information Processing Systems*, pp. 137–146, 2019.

[87] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2375–2384, 2019.

[88] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," *arXiv preprint arXiv:2001.03093*, 2020.

[89] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14424–14432, 2020.

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

[91] H. Manh and G. Alaghband, "Scene-lstm: A model for human trajectory prediction," *arXiv preprint arXiv:1808.04018*, 2018.

[92] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. Nian Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12126–12134, 2019.

[93] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei, "Peeking into the future: Predicting future person activities and locations in videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5725–5734, 2019.

[94] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2090–2096, IEEE, 2019.

[95] I. Hasan, F. Setti, T. Tsesmelis, A. Del Bue, F. Galasso, and M. Cristani, "Mx-lstm: mixing tracklets and vislets to jointly forecast trajectories and head poses," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6067–6076, 2018.

[96] H. Bi, R. Zhang, T. Mao, Z. Deng, and Z. Wang, "How can i see my future? fv-traj: Using first-person view for pedestrian trajectory prediction," in *European Conference on Computer Vision*, pp. 576–593, Springer, 2020.

[97] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, "Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation," *arXiv preprint arXiv:2007.14558*, 2020.

[98] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," *arXiv preprint arXiv:2004.02025*, 2020.

[99] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1468–1476, 2018.

[100] K. Messaoud, N. Deo, M. M. Trivedi, and F. Nashashibi, "Trajectory Prediction for Autonomous Driving based on Multi-Head Attention with Joint Agent-Map Representation," Sept. 2020. https://arxiv.org/abs/2005.02545.

[101] D. Lee, Y. Gu, J. Hoang, and M. Marchetti-Bowick, "Joint interaction and trajectory prediction for autonomous driving using graph neural networks," *arXiv preprint arXiv:1912.07882*, 2019.

[102] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1179–1184, IEEE, 2018.

[103] X. Li, X. Ying, and M. C. Chuah, "Grip: Graph-based interaction-aware trajectory prediction," in *2019 IEEE INTELLIGENT TRANSPORTATION SYSTEMS CONFERENCE (ITSC)*, IEEE, 2019.

[104] X. Li, X. Ying, and M. C. Chuah, "Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving," *arXiv preprint arXiv:1907.07792*, 2020.

[105] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "Pip: Planning-informed trajectory prediction for autonomous driving," *arXiv preprint arXiv:2003.11476*, 2020.

[106] H. Bi, Z. Fang, T. Mao, Z. Wang, and Z. Deng, "Joint prediction for kinematic trajectories in vehicle-pedestrian-mixed scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 10383–10392, 2019.

[107] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, 2018.

[108] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," *arXiv preprint arXiv:1809.10732*, 2018.

[109] K. Gillmeier, F. Diederichs, and D. Spath, "Prediction of ego vehicle trajectories based on driver intention and environmental context," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 963–968, IEEE, 2019.

[110] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, "Trajnet: Towards a benchmark for human trajectory prediction," *arXiv preprint*, 2018.

[111] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference on Computer Vision*, pp. 160–176, Springer, 2016.

[112] B. O. Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[113] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[114] K. J. Lee, Q. Zhao, X. Tong, M. Gong, S. Izadi, S. U. Lee, P. Tan, and S. Lin, "Estimation of intrinsic image sequences from image+ depth video," in *European Conference on Computer Vision*, pp. 327–340, Springer, 2012.

[115] G. Ye, E. Garces, Y. Liu, Q. Dai, and D. Gutierrez, "Intrinsic video and applications," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 80, 2014.

[116] T. Narihira, M. Maire, and S. X. Yu, "Direct intrinsics: Learning albedo-shading decomposition by convolutional regression," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2992–2992, 2015.

[117] K. Kim, A. Torii, and M. Okutomi, "Joint estimation of depth, reflectance and illumination for depth refinement," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015.

[118] J. T. Barron and J. Malik, "Intrinsic scene properties from a single rgb-d image," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 17–24, IEEE, 2013.

[119] Q. Chen and V. Koltun, "A simple model for intrinsic image decomposition with depth cues," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 241–248, IEEE, 2013.

[120] J. Jeon, S. Cho, X. Tong, and S. Lee, "Intrinsic image decomposition using structure-texture separation and surface normals," in *Computer Vision–ECCV 2014*, pp. 218–233, Springer, 2014.

[121] N. Kong, P. V. Gehler, and M. J. Black, "Intrinsic video," in *Computer Vision–ECCV 2014*, pp. 360–375, Springer, 2014.

[122] M. Hachama, B. Ghanem, and P. Wonka, "Intrinsic scene decomposition from rgb-d images," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 810–818, 2015.

[123] D. Held, D. Guillory, B. Rebsamen, S. Thrun, and S. Savarese, "A probabilistic framework for real-time 3d segmentation using spatial, temporal, and semantic cues.," in *Robotics: Science and Systems*, vol. 12, 2016.

[124] O. Miksik, D. Munoz, J. A. Bagnell, and M. Hebert, "Efficient temporal consistency for streaming video scene analysis," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 133–139, IEEE, 2013.

[125] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.* MIT press, 2005.

[126] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, (Hong Kong, China), May 2014.

[127] R. Nock and F. Nielsen, "Statistical region merging," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 26, no. 11, pp. 1452–1458, 2004.

[128] J. T. Barron and J. Malik, "Shape, illumination, and reflectance from shading," *TPAMI*, 2015.

[129] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 112, 2013.

[130] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph," *Proc. Robotics: Science and Systems, Rome, Italy*, 2015.

[131] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman, "Ground truth dataset and baseline evaluations for intrinsic image algorithms," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 2335–2342, IEEE, 2009.

[132] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[133] Q. Ma, J. Yang, A. Ranjan, S. Pujades, G. Pons-Moll, S. Tang, and M. J. Black, "Learning to dress 3d people in generative clothing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6469–6478, 2020.