学　士　論　文

題　目　　道路シーンにおける歩行者の再配置によるデータ拡張

指導教員　　西野 恒 教授

京都大学工学部 電気電子工学科

氏 名　　Chen Shi

令和４年２月１０日

Contents

Abstract

Road Scene Pedestrian Relocation for Data Augmentation

Chen Shi

A robust pedestrian detector is essential for every self-driving vehicle. Recent deep-learning-based pedestrian detection algorithms have reported high overall accuracy on public benchmarks. In challenging cases, however, they tend to show a significant drop in performance. A representative example is detection of pedestrians at far distances imaged as small regions, which we refer to as "*far pedestrian* detection." A self-driving vehicle must be able to detect pedestrians from a distance to keep itself from hitting them, especially when it is running fast. Pedestrians would otherwise be put into great danger. In this thesis, we aim to give a solution to the *far pedestrian* detection problem. We focus on the lack of annotations on *far pedestrians* in autonomous driving datasets, which we believe has become the bottleneck of *far pedestrian* detection. Specifically, we propose a data augmentation strategy where we cut out annotated pedestrians and relocate them farther out with proper scale and occlusion. To evaluate our data augmentation strategy quantitatively, we train Mask R-CNN either with or without *pedestrian relocation* and compare their performances in *far pedestrian* detection and instance segmentation. The experimental results show that *pedestrian relocation* is able to slightly improve the performance in both tasks. To the best of our knowledge, we are the first to boost both detection and segmentation tasks with a focus on *far pedestrians* using data augmentation.

1 Far Pedestrian Detection Problem

In this chapter, we first introduce the background of pedestrian detection for autonomous driving and then raise "*far pedestrian* detection" as a critical yet unexplored problem. Finally, we briefly describe the purpose, method, evaluation, and conclusion of this research.

1.1 Background

Despite their enormous commercial potential, self-driving vehicles are yet to show convincing safety performance in real traffic conditions. Computer vision researchers have thus been striving to develop self-driving systems that are robust enough to deal with all types of traffic scenarios. Traffic collisions can be roughly categorized into Vehicle-to-Vehicle (V2V) collisions and Vehicle-to-Pedestrian (V2P) collisions. While V2V collisions can be prevented through mutual communication between vehicles (V2V communication [1]), a self-driving car must detect pedestrians in its way in advance to avoid hitting them.

Research on road scene pedestrian detection has seen remarkable progress since the emergence of public benchmarks. Before the prevalence of deep learning, most pedestrian detection algorithms were based on the classification of low-level features. For example, Dalal et al. [2] extract Histogram of Oriented Gradients (HOG) feature from road scene images and use a linear SVM classifier to detect pedestrians. In recent years, deep-learning-based methods such as Faster R-CNN [3] and YOLO [4] have greatly improved both the accuracy and runtime performance of pedestrian detection. Public benchmarks also played a significant role in the progress of pedestrian detection. Large-scale datasets provided by these benchmarks facilitated the evaluation and comparison of different methods designed for the task. Furthermore, the various metrics customized for different failure cases help clarify the shortcomings of each algorithm.

Despite the high overall accuracy on the benchmarks, deep-learning methods usually struggle to detect far-away pedestrians who appear small in the images. Pedestrian detection benchmarks typically group the annotated pedestrians in the datasets into different groups representing different difficulties. In most cases, the difficulty levels are defined according to the extent of occlusion (being partly hidden) and scale (size of the pedestrian appearing in the image). They are broadly considered the two major contributors to the failure cases in pedestrian detection [5, 6]. In recent years, deep-neural-network models have produced impressive results in scenarios with reasonable difficulty. When the evaluation is limited to "small-scale" pedestrians, however, their performance declines dramatically. On the CityPersons Benchmark [7], Faster R-CNN achieves a log-average miss rate (LAMR) of 12.97% for "reasonable" subset (50 pixels or more in height) but has a LAMR as high as 37.24% for "reasonable, small" subset (50 to 75 pixels in height).

When self-driving vehicles cannot detect pedestrians from a distance, they may have inadequate time to steer themselves or brake before hitting people. We refer to this problem as the "*far pedestrian* detection" problem. It poses a huge threat to pedestrians especially when the cars are running at a high speed. The difficulty of detecting *far pedestrians* can be attributed to the quantization error due to low resolution [5, 6]. When the resolution is increased, however, computational costs go up and real-time detection becomes more difficult. Thus, there is a trade-off between *far pedestrian* detection accuracy and its runtime performance. Most recent works attempt to tackle the *far pedestrian* detection problem by working on the architectures of deep neural networks. For example, Qi et al. [8] proposed an "aligned multi-scale training" strategy to better adapt detectors to low-resolution objects.

Instead of modifying the detector itself, we focus on the lack of annotations on *far pedestrians* in autonomous driving datasets. Fig. 1.1 shows the distribution of distances of the pedestrians annotated with 2D bounding boxes in the KITTI dataset. The frequency roughly declines with distance (when distance $> 10m$) and only 0.38% (44 out of 11470) of annotated pedestrians are over 50 meters away. There are, however, pedestrians in this distance range (according to the depth maps provided by KITTI [9]) captured in the raw images but left unannotated. This is detrimental to *far pedestrian* detection because any prediction would be judged as a false positive (misdetection) when no ground truth label is attached to this region. Therefore, we cannot expect a detector trained on KITTI to learn to detect pedestrians that are more than 50 meters away. This is extremely dangerous because a car running at 100 km/h can cover 50 meters in 2 seconds. To solve this problem, we need to complement the datasets with the missing annotations on *far pedestrians* before training detectors to extract features from them. An augmented dataset with adequate annotations on *far pedestrians* would help unleash the potential of detection algorithms for *far pedestrian* detection.

1.2 Overview

In order to address the *far pedestrian* detection problem, we aim to augment existing autonomous driving datasets with more accurately annotated pedestrians at far positions. Specifically, we cut out annotated pedestrians from the dataset and relocate them farther out along with the labels on them with proper scale and occlusion. We call this method "*pedestrian relocation*."

We propose a static one-frame *pedestrian relocation* method and a dynamic multi-frame relocation framework built on top of it. The cut-outs are obtained with existing instance segmentation methods (e.g. Mask R-CNN [10]) for one-frame *pedestrian relocation* or with video panoptic segmentation (e.g. VPSNet [11]) for multi-frame usage. The scales of relocated pedestrians are determined by regressing a *scale factor* for each image (See Sec. 3.2). We realize photorealistic pedestrian synthesis with a novel strategy that combines Poisson Image Editing [12], color shift and edge blurring (See Sec. 3.3). We generate clear occlusion using depth maps that are refined with high-resolution depth maps [13] estimated from corresponding raw images (Also see Sec. 3.3). For *dynamic pedestrian relocation*, the camera ego-motion needs to be obtained first. It is achieved with OpenVSLAM [14], if ground truth inter-frame relative camera poses are not provided (See Sec. 3.4). After the acquisition of camera ego-motion, we estimate 3D motions of cut-out pedestrians from source image sequences (See Sec. 3.5) and reproduce them in target sequences (See Sec. 3.6).

We evaluate *pedestrian relocation* both qualitatively and quantitatively. First, we show example outputs of the *pedestrian relocation* pipeline. We demonstrate with these examples how each approach contributes to the plausibility and naturalness of the final results. Second, we verify the effectiveness of *pedestrian relocation* as a data augmentation strategy for *far pedestrian* detection and instance segmentation through comparative experiments. As a result, we found that our method is able to slightly improve the accuracy of Mask R-CNN in both tasks.

In summary, the contributions of this research are two-fold:

1. We propose a *static pedestrian relocation* method and find through experiments that it slightly improves the performance of Mask R-CNN in both detection and instance segmentation regarding "*far pedestrians*" (50 pixels or less in height).

2. We further propose a dynamic *pedestrian relocation* framework that realizes sequence-to-sequence relocation with spatio-temporal consistency across frames.

We present a novel solution to the *far pedestrian* detection problem. We believe our *pedestrian relocation* strategy will open up new possibilities for the autonomous driving industry.

2 Related Work

In this chapter, we first introduce previous works on the topic of pedestrian detection with special emphasis on those attempting to improve *far pedestrian* detection. Next, we give a brief introduction of several datasets related to this research. Finally, we present multiple existing works that perform pedestrian synthesis into traffic scenes and compare them with *pedestrian relocation*.

2.1 Far Pedestrian Detection

Pedestrian detection is a well-studied topic in the field of computer vision. Initial pedestrian detection algorithms depended on classifiers such as linear SVM [2] and Adaboost [15] to classify low-level features extracted from images such as HOG [2], color self similarity [16] and shapelet features [17]. In recent years, deep-learning-based methods including YOLOv3 [18] and Faster R-CNN [3] have been updating the state-of-the-art accuracy.

Despite high overall accuracy in pedestrian detection, special scenarios such as occluded or distant pedestrians still remain challenging. As is pointed out by [5] and [6], low resolution of *far pedestrians* contributes to a considerable part of detection failures. Dinakaran et al. [19] claim that the issue can be alleviated by increasing resolution. Consequent rise of computational cost, however, makes it impractical for self-driving application which requires real-time inference.

Several recent works have attempted to tackle the *far pedestrian* detection problem by modifying network architectures for better adaptation to different scales of people [3, 10, 8, 20, 21, 22]. These methods, however, still cannot realize their full potential without sufficient annotations on *far pedestrians*. Thus, datasets with incomplete pedestrian annotations have become a bottleneck of the *far pedestrian* detection paradigm. Therefore, we consider it more important to complement the datasets with more accurate annotations on *far pedestrians* to lay a foundation for methods based on deep neural networks.

2.2 Traffic Scene Datasets

Rapid improvement of pedestrian detection algorithms is also a result of an emergence of public pedestrian detection benchmarks. Large-scale datasets not only enabled model training with large amounts of data but also made detailed evaluations of different scenarios possible. For example, the Caltech Pedestrian Dataset [23] grouped pedestrians into three scales based on their pixel-wise heights: near (80 or more pixels), medium (30-80 pixels) and far (30 pixels or less). CityPersons [7], a pedestrian detection dataset based on Cityscapes [24], similarly categorizes those pedestrians whose pixel-wise heights are between 30 and 80 as "smaller" pedestrians. Pixel-wise height is a metric suitable for the definition of "*far pedestrians*" for its strong correlation with the difficulty of human detection and segmentation. There is not, however, a universal definition that applies to all datasets since the relationship between pixel-wise heights of pedestrians and their actual distances is highly dependent on image resolutions.

Comprehensive road scene datasets such as KITTI [9] and Cityscapes offers diverse types of annotations, which opens up new possibilities for pedestrian-related tasks. KITTI provides annotation files for tasks such as depth estimation, visual odometry, object detection and tracking, semantic segmentation, and optical flow. KITTI-STEP (Segmenting and Tracking Every Pixel) [25], a subset of KITTI, provides 20 image sequences with dense panoptic annotations and object tracking IDs. We executed *pedestrian relocation* on KITTI-STEP in our quantitative experiment (Sec. 4.2) since our method requires high-quality panoptic segmentation maps as preparation. Cityscapes provides raw images with higher resolution and thus have more distant objects annotated than KITTI [24]. This is the reason why we chose Cityscapes as the test dataset.

2.3 Pedestrian Synthesis

Object synthesis in traffic scenes has been widely studied in recent years. Chen et al. [26] enabled geometry-aware object composition into novel scenes rendered at novel poses. This work, however, focuses on rigid object insertion (e.g. vehicles), and its effectiveness on deformable human bodies is not evaluated.

As for human image synthesis, Minkesh et al. [27] extract human images from source images with Mask R-CNN [10] and simply paste them into target images. This direct "copy and paste" approach leaves their synthesis results unnatural. By contrast, our method accomplishes photorealistic results by integrating Poisson Image Editing [12] into our synthesis methodology. Naturalness of our synthesis results allows CNNs (Convolutional Neural Networks) to extract useful features for pedestrian detection. Wang et al. [28] implemented a user interface where one can manually insert a pedestrian into a scene with realistic scale and lighting. They also mentioned data augmentation as a potential application of it. While their method supports only one-frame insertion, ours can further achieve spatio-temporally consistent multi-frame pedestrian relocation. Moreover, we quantitatively evaluated the effectiveness of our method as a data augmentation strategy.

There are also other recent works that use pedestrian synthesis as a way to augment datasets. A great proportion of them use GANs (Generative Adversarial Networks) to generate new pedestrians. Zhi et al. [29] use a GAN to change poses of pedestrians in the EuroCity Person dataset [30] for better generalization. Note that it does not increase the number of pedestrians in the dataset and is thus unable to generate more *far pedestrians*. Similarly, the GAN model proposed by Vobecky et al. [31] generates synthetic pedestrians with new poses for data augmentation, but they did not show *far-pedestrian*-based results, either.

Zhang et al. [32] take a slightly different approach to data augmentation. They let GAN learn plausible object placements in traffic scenes and rearrange objects in the scenes to diversify datasets. It relies on segmentation of salient objects in the first place, which makes object placement at far positions difficult for the GAN. On the other hand, our method makes full use of geometric information to realize *pedestrian relocation* that is robust to distance variation.

To sum up, we are the first to focus on boosting *far pedestrian* detection and instance segmentation though there have been similar works previously.

3 Pedestrian Relocation

In this chapter, we first give an overview of the whole *pedestrian relocation* pipeline. Then, we go into detail about methods implemented in each module defined in the overview section.

3.1 Overview

In this thesis, we propose a *pedestrian relocation* framework for data augmentation. As shown in Fig. 3.1, the *pedestrian relocation* pipeline involves cutting out annotated pedestrians from source image sequences and synthesizing them into target scenes with correct scale, occlusion, and spatio-temporal consistency. The whole pipeline can be divided into 5 major modules: (1) a scale regression module that samples "ground" pixels from the depth map of a given scene and calculate a scale factor k that best describes the scales of pedestrians throughout the scene, (2) a photorealistic pedestrian synthesis module that naturally synthesizes a given cut-out pedestrian into a specified target image with the correct scale and occlusion, (3) a camera ego-motion estimation module to estimate inter-frame relative camera poses, (4) a pedestrian motion estimation module that calculates 3D motions of pedestrians in their source scenes, and (5) a final *dynamic pedestrian relocation* module that aggregates the 3D motions and the camera ego-motion to update the location of the pedestrian in the next target frame.

For static one-frame synthesis, only modules (1) and (2) are required. To perform temporally consistent *dynamic pedestrian relocation* between two i-frame sequences, all five modules are needed and the pipeline is cycled i times. For both static and dynamic usage, depth maps and panoptic segmentation[33] of both source and target images are prepared in advance. It is desirable to have ground truth depth maps that provide correct scales since *pedestrian relocation* relies heavily on depth information to calculate scales and realize robust 3D transform. Panoptic segmentation can be achieved by UPSNet[34] for single images or VPSNet[11] for image sequences. For *dynamic pedestrian relocation* only, the pipeline further requires relative camera poses to enable camera coordinate transform. It is ideal to use ground truth camera poses, but since few datasets provide them, we will introduce an alternative solution with visual SLAM in Sec. 3.4. In addition, the scale regression module uses keypoint information to filter out invalid (occluded, truncated, too small, etc.) pedestrians. Again, we typically use an existing keypoint detector (e.g. OpenPose[35]) to fulfill this purpose since most mainstream autonomous driving datasets do not contain human pose information.

3.2 Scale Regression

Since correct scales are required for plausible pedestrian relocation, we need to find the size to which a pedestrian cut-out should be rescaled for any specified position in a target scene. Inspired by Wang et al. [28, 36], we use linear regression to calculate a scale factor k for every image. The k describes the relationship between pixel-wise heights of pedestrians and their depths (distances from camera) in an image. We base scale regression on two assumptions: (1) the ground can be well-approximated by a plane; (2) all pedestrians are the same height in the 3D world coordinate. Scale regression is the same as what Wang et al. proposed in [28] except when real heights of pedestrians are known (as is the case with the KITTI object tracking dataset). In this case, we leave out assumption (2) and obtain a more accurate k by normalizing all pedestrian heights to a preset standard height.

According to assumption (1), all ground points (X, Y, Z) in a 3D world coordinate should follow a plane equation:

$$AX + BY + CZ = 1\,. \tag{3.1}$$

Let H be the standard height of pedestrians in world coordinates, then we have under perspective projection that

$$x = X \cdot \frac{f}{Z}, y = Y \cdot \frac{f}{Z}, h = H \cdot \frac{f}{Z}\,, \tag{3.2}$$

where (x, y) is the bottom middle point (middle point of the lower edge of the bounding box) of any pedestrian in the image coordinate, h stands for their pixel-wise height and f denotes the focal length of camera. Eliminating X and Y in Eq.

3.1 with Eq. 3.2, we have

$$\frac{A}{f}x + \frac{B}{f}y + C = \frac{1}{Z}\,. \tag{3.3}$$

Letting $a = \frac{A}{f}$, $b = \frac{B}{f}$, $c = C$ and $k = fH$, finally we have

$$ax + by + c = \frac{1}{Z}\,, \tag{3.4}$$

$$h = k \cdot \frac{1}{Z}\,, \tag{3.5}$$

where k is the scale factor. Note that Eq. 3.4 applies to all pixels belonging to the ground. Therefore, we can sample image coordinates and their corresponding ground truth depths (x, y, Z) from "ground" pixels to regress plane parameters (a, b, c). Then, pixel-wise heights of all pedestrians in an image can be used to calculate k with another linear regression.

Fig. 3.2 shows an outline of the scale regression module. First, we generate a binary mask indicating "ground" regions from the semantic segmentation mask. Next, we randomly get a certain number of (x, y, Z) samples (Z can be obtained from depth maps) within the binary mask and use linear regression based on RANSAC algorithm [37] to solve for plane parameters (a, b, c). Instead of totally random sampling, we use a sampling weight that is proportional to Z^3 to acquire unbiased samples with respect to depth. In parallel, keypoint detection is performed to filter out invalid pedestrians whose heights are not fully visible. Usually, we use OpenPose [35] as the keypoint detector. This step can be replaced by a usage of occlusion labels, if available (as is the case with the KITTI object tracking dataset). Then, we collect the bottom middle points (x, y) and pixel-wise heights h of all valid pedestrians in the image. Finally, we calculate disparities (inverse depths) $1/Z$ through Eq. 3.4 and compute the scale factor k using linear regression. Namely, we solve for the k in

$$k(ax + by + c) = h\,. \tag{3.6}$$

Note that k is set to a default value when no valid pedestrian is found throughout the scene.

3.3 Photorealistic Pedestrian Synthesis

Plausible pedestrian synthesis should achieve correct scale and occlusion. First, this module takes as input a specified bottom middle point (x, y) at which to put a pedestrian and get the corresponding target depth Z. If it is the first frame to place the pedestrian, the module randomly selects an (x, y) from ground pixels and gets the Z from the ground truth depth map. Otherwise, (x, y, Z) is updated by the dynamic *pedestrian relocation* module. Next, it rescales the given cut-out bounding box by aligning its height to the h obtained through Eq. 3.5 using k and Z. Then, it queries the depth map to judge which parts of the synthesized pedestrian should be occluded. For most datasets, ground truth depth maps either have unclear edges (e.g. Cityscapes [24]) or are overly sparse (e.g. KITTI [9]). As shown in Fig. 3.3, we estimate high-resolution depth maps with MergeNet [13] for all frames and align their scales with their ground truth counterparts. We make up for the incompleteness of ground truth depth maps in this way. The alignment is achieved through RANSAC linear regression. Note that initialization of pedestrian position and depth map alignment adopt the same weighted sampling methodology as that of plane parameter regression (See Sec. 3.2).

As shown in Fig. 3.4, we further adjust colors and blurriness of pedestrians to fit them in the target backgrounds naturally. In terms of color adjustment, we take a mixed strategy of Poisson Image Editing (PIE) [12] and conventional "copy and paste." The general idea of PIE is to seamlessly embed a foreground image into a background image by retaining the gradients of the source image while keeping the continuity at the composition boundary. It works relatively well when the background is generally the same color. As shown in Fig. 3.5, however, pedestrians synthesized with PIE are prone to be eroded by complicated road scene backgrounds with sharp color change and thus look transparent. Nonetheless, pedestrians synthesized in this way are usually globally inconspicuous. We take advantage of this by computing the average color of the pedestrian synthesized with PIE to guide a color shift process. Specifically, we first compute the difference of average RGB colors within the segmentation mask of the pedestrian between the PIE result and the source image. We then let the RGB color of the source image shift by a certain proportion of that difference. Namely, the color-shifted pedestrian cut-out image I'_c should be

$$I'_c(x, y) = I_c(x, y) + \alpha \sum_{(x,y)\in mask} (J_c(x, y) - I_c(x, y)) \, , \tag{3.7}$$

where I_c is the rescaled pedestrian cut-out image, J_c is the PIE result, $c \in \{R, G, B\}$ is a color channel, *mask* here is a binary mask indicating which pixels belong to the pedestrian and is derived from the panoptic segmentation mask, and α is a color shift coefficient (typically we set $\alpha = 0.2$). Then we paste the color-shifted pedestrian cut-out into the target image. This step often causes jagged edge artifacts (Fig. 3.4(c)) due to aliasing. As shown in Fig. 3.4(a), we fix this issue by blurring the edges with a Gaussian filter.

We assign the tracking ID to the synthesized pedestrian if specified by the *dynamic pedestrian relocation* module (See Sec. 3.6). A new instance ID is given to the synthesized pedestrian otherwise. The corresponding region in the panoptic segmentation map is updated accordingly. The same region in the refined depth map is also updated with the target depth Z when the current frame is not the last in a sequence. In addition, we apply random horizontal flip to further diversify appearances of synthesized pedestrians.

3.4 Camera Ego-Motion Estimation

Static pedestrian relocation can be achieved with the two modules described above. When it comes to sequence-to-sequence dynamic relocation, however, an existence of camera ego-motion (motion of the camera itself across frames) would hinder both pedestrian motion estimation from source sequences and their relocation to target sequences. Therefore, we need to estimate the camera ego-motions of both sequences first to remove the hindrance.

The camera ego-motion of a sequence of images can be described by a series of relative camera poses. Every relative camera pose between two frames is composed of a 3×3 rotation matrix R and a 3×1 translation vector $\boldsymbol{t}$. The rotation and translation represent a change of camera orientations and camera positions, respectively.

If ground truth relative camera poses are not contained in the dataset, we use OpenVSLAM [14] to estimate them from raw image sequences. OpenVSLAM is an open-source visual SLAM (Simultaneous Localization and Mapping) toolbox. It can output camera poses of all frames with respect to the first frame with an image sequence as input. Notably, monocular visual SLAM algorithms tend to estimate inaccurate translations especially when the camera is moving slowly or rotating fast. There are two workarounds to this issue: (1) If vehicle speed information is available (as is the case with Cityscapes), we rescale the Euclidean norm of every output translation to the corresponding vehicle speed; (2) If stereo images are available, the stereo option of OpenVSLAM can be used for better estimation of translations.

3.5 Pedestrian Motion Estimation

With known relative camera poses, we can use them to counteract the camera ego-motion and find out what the 3D motion of each pedestrian would be in a fixed camera coordinate system. As shown in Fig. 3.6, the goal of this module is to calculate the 3D pedestrian motion in the camera coordinate system of the first frame of a sequence.

We show relationships among different coordinate systems in Fig. 3.7. First, for each frame i where a pedestrian appears, we back-project the image coordinate of their bottom middle point $\boldsymbol{x}_I^i = (x^i, y^i)$ to the current camera coordinate $\boldsymbol{X}_{C_i}^i$:

$$\boldsymbol{X}_{C_i}^i = \begin{bmatrix} X_{C_i}^i \\ Y_{C_i}^i \\ Z_{C_i}^i \end{bmatrix} = Z_{C_i}^i \begin{bmatrix} \boldsymbol{x}_N^i \\ 1 \end{bmatrix} = Z_{C_i}^i K^{-1} \begin{bmatrix} \boldsymbol{x}_I^i \\ 1 \end{bmatrix}, \; K = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.8}$$

where $\boldsymbol{x}_N^i = (\frac{X_{C_i}^i}{Z_{C_i}^i}, \frac{Y_{C_i}^i}{Z_{C_i}^i})$ is the normalized camera coordinate and K is the intrinsic matrix of camera (the focal lengths f_x, f_y and the optical center (x_0, y_0) are usually available as camera calibration parameters). In addition, $Z_{C_i}^i$ is derived from the ground truth depth map.

Second, we transform all 3D pedestrian positions into the camera coordinate of the first frame. As we have the relative camera pose of frame i with respect to the first frame ($R_{C_1}^{C_i}$ and $\boldsymbol{t}_{C_1}^{C_i}$), camera coordinates $\boldsymbol{X}_{C_i}^i$ and $\boldsymbol{X}_{C_1}^i$ should satisfy

$$\boldsymbol{X}_{C_i}^i = R_{C_1}^{C_i} \boldsymbol{X}_{C_1}^i + \boldsymbol{t}_{C_1}^{C_i} \,. \tag{3.9}$$

We invert this process to calculate $\boldsymbol{X}_{C_1}^i$:

$$\boldsymbol{X}_{C_1}^i = {R_{C_1}^{C_i}}^{-1} \boldsymbol{X}_{C_i}^i - {R_{C_1}^{C_i}}^{-1} \boldsymbol{t}_{C_1}^{C_i} = R_{C_i}^{C_1} \boldsymbol{X}_{C_i}^i + \boldsymbol{t}_{C_i}^{C_1} \,, \tag{3.10}$$

where $R_{C_i}^{C_1} = {R_{C_1}^{C_i}}^{-1}$ and $\boldsymbol{t}_{C_i}^{C_1} = -{R_{C_1}^{C_i}}^{-1} \boldsymbol{t}_{C_1}^{C_i}$.

After the unification of all 3D positions into the camera coordinate system of the first frame, we finally get camera-invariant 3D motions of the pedestrian that are ready for transfer to other sequences:

$$\Delta \boldsymbol{X}^i = \boldsymbol{X}_{C_1}^{i+1} - \boldsymbol{X}_{C_1}^i,\ i = 1, 2, 3, ..., L-1 \,, \tag{3.11}$$

where L is the length of the image sequence where the pedestrian appears.

3.6 Dynamic Pedestrian Relocation

As shown in Fig. 3.7, we solve an inverse problem of pedestrian motion estimation to achieve spatio-temporally consistent pedestrian relocation. Given the 3D motion of a pedestrian, their position in the previous frame, and relative camera pose, we need to find out where they should appear in the next frame. In this way, we can update the bottom middle point and target depth indicating where the pedestrian should be in the next frame and start the next cycle.

Fig. 3.8 shows an outline of the *dynamic pedestrian relocation* module. First, the module takes as input the current bottom middle point (x^i, y^i) and target depth $Z_{C_i}^i$ and obtain the 3D pedestrian position in the current camera coordinate system $\boldsymbol{X}_{C_i}^i$ through backward projection (Eq. 3.8). Second, it updates the 3D position by adding the motion estimated from the source sequence:

$$\boldsymbol{X}_{C_i}^{i+1} = \boldsymbol{X}_{C_i}^i + \Delta \boldsymbol{X}^i \,. \tag{3.12}$$

Note that the updated position is still in the previous camera coordinate system. Since we do not always have the relative camera pose between frame i and $i+1$, we transform $\boldsymbol{X}_{C_i}^{i+1}$ to $\boldsymbol{X}_{C_1}^{i+1}$ first and then $\boldsymbol{X}_{C_1}^{i+1}$ to $\boldsymbol{X}_{C_{i+1}}^{i+1}$:

$$\boldsymbol{X}_{C_1}^{i+1} = {R_{C_1}^{C_i}}^{-1} \boldsymbol{X}_{C_i}^{i+1} - {R_{C_1}^{C_i}}^{-1} \boldsymbol{t}_{C_1}^{C_i} = R_{C_i}^{C_1} \boldsymbol{X}_{C_i}^{i+1} + \boldsymbol{t}_{C_i}^{C_1} \,, \tag{3.13}$$

$$\boldsymbol{X}_{C_{i+1}}^{i+1} = R_{C_1}^{C_{i+1}} \boldsymbol{X}_{C_1}^{i+1} + \boldsymbol{t}_{C_1}^{C_{i+1}}. \tag{3.14}$$

Finally, we project $\boldsymbol{X}_{C_{i+1}}^{i+1}$ to the image plane

$$\begin{bmatrix} \boldsymbol{x}_I^{i+1} \\ 1 \end{bmatrix} = K \begin{bmatrix} \boldsymbol{x}_N^{i+1} \\ 1 \end{bmatrix} = K \begin{bmatrix} X_{C_{i+1}}^{i+1} / Z_{C_{i+1}}^{i+1} \\ Y_{C_{i+1}}^{i+1} / Z_{C_{i+1}}^{i+1} \\ 1 \end{bmatrix} = \frac{1}{Z_{C_{i+1}}^{i+1}} K \boldsymbol{X}_{C_{i+1}}^{i+1} \tag{3.15}$$

and update the bottom middle point to $\boldsymbol{x}_I^{i+1} = (x^{i+1}, y^{i+1})$ and the target depth to $Z_{C_{i+1}}^{i+1}$ for the next cycle. The tracking ID is also propagated as it is to the next cycle.

4 Experimental Results

In this chapter, we evaluate our method both qualitatively and quantitatively. First, we show both success and failure cases of the *pedestrian relocation* results. We demonstrate with the successful ones how the approaches applied in the framework contribute to plausibility of final results. With the failure cases, we try to point out the parts of *pedestrian relocation* that still need improvement. Next, we quantitatively evaluate the effectiveness of *pedestrian relocation* as a data augmentation strategy for *far pedestrian* detection and instance segmentation.

4.1 Qualitative Evaluation of Pedestrian Relocation

First, we show example results of photorealistic pedestrian synthesis in Fig. 4.1. In the success cases ((a)-(e)), we can see the colors of the synthesized pedestrians are well adjusted to fit into the target environment. In the failure cases ((f)-(j)), however, we find the synthesized pedestrians too bright compared to the dark backgrounds. It shows that our synthesis method struggles to handle a large difference in brightness between source scenes and target scenes.

Second, we display an example where the refined depth map generated correct occlusion. We can see in Fig. 4.2 that the synthesized pedestrian is occluded by a black car. This example image is cropped from an output image of *dynamic pedestrian relocation.* This type of heavy occlusion, however, is rarely generated in *static pedestrian relocation* since we initialize the bottom middle point of a synthesized pedestrian with a visible "ground" pixel. This bias may cause models to struggle with occluded pedestrians.

Finally, we show example results of the *dynamic pedestrian relocation* pipeline in Fig. 4.3. In the success case (Fig. 4.3(a)), the walking motion of the synthesized pedestrian is plausible and is not influenced by camera ego-motion. In the failure case (Fig. 4.3(b)), the walking trajectory of the synthesized pedestrian is unnaturally zigzag. This type of failure cases is scene-dependent according to our observation. We consider it a consequence of inaccurate estimation of relative camera poses either in the source or target sequence.

4.2 Overview: Quantitative Evaluation of Data Augmentation

The purpose of our data augmentation strategy is to enable deep neural networks to detect pedestrians at farther distances. As shown in Fig. 4.4, the Cityscapes dataset contains more annotated "*far pedestrians*" (defined later in this section) than the KITTI dataset. Based on this observation, we train the same model on KITTI either with or without our data augmentation method implemented and compare their performances regarding *far pedestrians* on Cityscapes. The tasks we tested are single-image pedestrian detection and instance segmentation. We train a Mask R-CNN to perform these two tasks jointly.

We design our experiment as below: (1) We augment the KITTI-STEP dataset with *static pedestrian relocation*; (2) We train Mask R-CNN on KITTI-STEP with and without data augmentation, respectively; (3) We fine-tune both trained models to the Cityscapes training set; (4) We evaluate and compare the performances of the two fine-tuned models on the Cityscapes validation set.

To show the effectiveness of our data augmentation strategy clearly, we give a new definition of "*far pedestrians*" and make evaluation metrics strictly about them. According to the statistics about KITTI-STEP shown in Fig. 4.4, the frequency of pedestrians starts to decline sharply when the pixel-wise height falls below 50, and only 9.4% (686 out of 7257) of annotated pedestrians are in this range. Therefore, we define a pedestrian whose pixel-wise height is 50 or less as a "*far pedestrian.*" During the *pedestrian relocation* phase, we only add pedestrians at positions where they can be rescaled to 50 pixels tall or less. We also only count "*far pedestrians*" in the final evaluation.

4.3 Results

We start with a detailed description of experimental setups. We added 26911 synthetic pedestrians in total (from 7257 to 34169). The color shift coefficient was $\alpha = 0.2$. We use Mask R-CNN with a ResNet-50 backbone pretrained on ImageNet [38]. During training on KITTI-STEP, batch size is set to 16 and we train 90000 iterations in total. During fine-tuning on Cityscapes training dataset, we freeze weights in the backbone (i.e. only train prediction heads) to retain its capability of feature extraction trained on KITTI-STEP. We still set the batch size to 16 and train 10000 iterations during this stage. We adopt stochastic gradient descent (SGD) as the optimizer. We convert all KITTI-STEP and Cityscapes annotations into the COCO format [39] for training, validation and training.

We show the quantitative results in Table 4.1. The metrics we adopted are AP (average precision), AP_{50} (AP with IoU threshold at 0.5) and AP_{75} (AP with IoU threshold at 0.75), all of which are official COCO metrics. We can see that the model trained with *pedestrian relocation* as data augmentation has a slight advantage over the one trained without data augmentation in every metric (except for AP_{75} for instance segmentation where the difference is as small as 0.001). Since all other conditions are kept the same, we can conclude that our data augmentation strategy is effective for both *far pedestrian* detection and instance segmentation.

As shown in Fig. 4.5, the model trained with augmented data is able to detect *far pedestrians* with higher confidence than the one trained without data augmentation. It shows that our data augmentation strategy helped the model detect and segment farther pedestrians.

We show in Fig. 4.6 an example of misdetections made only by the model trained with data augmentation. We attribute the emergence of these false positives to low-quality pedestrian synthesis. There exist relocated pedestrians in the outputs who can hardly be recognized as humans. There are two possible reasons for this. First, shape features of these pedestrians are lost due to erroneous panoptic masks. Second, improper down-sampling strategy introduced noise or artifacts to the appearance of these pedestrians. If we can fix these issues in the future, we expect that frequencies of false positives will go down and AP can be further improved.

5 Conclusion

In this research, we proposed a *pedestrian relocation* framework to augment existing autonomous driving datasets with annotated *far pedestrians*. We quantitatively evaluated the effectiveness of our method for *far pedestrian* detection and instance segmentation. We found through comparison that the Mask R-CNN trained on the augmented dataset performed slightly better than the one trained on the original dataset. We believe that our data augmentation strategy will lay a solid foundation for future improvement of *far pedestrian* detection.

There are a few directions in which we can possibly improve the *pedestrian relocation* method. First, appearances of synthesized pedestrians can be further diversified by a proper pose transfer process. Pose transfer is a task that requires a target person to change their pose in accordance with the pose of a source person. There already exist several pose transfer algorithms [40, 41] that can be used. We expect that integration of pose transfer into the *pedestrian relocation* framework would make models trained on datasets augmented by our method more robust to domain shifts. Second, we can transform our framework into a GAN structure. We can still keep the rescaling process of pedestrians deterministic since it has already shown satisfactory robustness. The synthesis results, however, still look unnatural in some special cases. As already shown in Sec. 4.1, the color shift approach we adopted cannot handle well pedestrian synthesis into dark backgrounds. To promote naturalness, we can design a discriminator that tries to tell whether a pedestrian in a generated image is synthesized or not. If naturalness of pedestrian synthesis is improved, we can expect the trained model to produce fewer false positives during tests.

In the future, our data augmentation strategy can also be tested for other computer vision tasks. First, we can find out whether *pedestrian relocation* is also effective for human pose estimation through similar experiments. For one-frame human pose estimation models such as OpenPose [35], *static pedestrian relocation* may be helpful to capture human keypoints from a farther distance. For models that achieve temporally consistent multi-frame pose estimation such as OpenPifPaf [42], we can apply *dynamic pedestrian relocation*. If our method can also boost human pose estimation, it will help autonomous driving systems better understand behaviors of *far pedestrians*. For example, an estimation of the head pose of a far-away pedestrian can help a self-driving car judge if the pedestrian is aware of the car approaching. We believe such applications would further improve safety performance of self-driving vehicles. Apart from safety improvements, there are also other potential applications that can facilitate the lives of people. For example, it can assist self-driving taxis to detect hand-waving gestures from farther distances. Second, we can verify the effectiveness of the *dynamic pedestrian relocation* method for pedestrian tracking and re-identification. If a self-driving vehicle is able to track and re-identify pedestrians from farther distances, it can, for example, predict trajectories of far-away pedestrians and thus have more time to plan its moves.

Acknowledgements

I would first like to express my deepest appreciation to my advisors Professor Ko Nishino and Associate Professor Shohei Nobuhara. This research would not have been possible without their help. I would also like to extend my gratitude to the lab seniors. Among them, I am particularly grateful to Yuzheng Xu who kept providing valuable advice for me during this research. I must also thank Chengbo Shi for sharing his knowledge and experience about the KITTI dataset and Mask R-CNN, which was really helpful. I also had great pleasure of working with Katsuya Masuno from Yazaki Corporation in the first few months of this research. Last but not least, I would like to give a special thank to my family and my girlfriend who always supported me. I cannot wait to see them after my graduation.

References

[1] Albert Demba and Dietmar PF Möller, Vehicle-to-vehicle communication technology, *2018 IEEE International Conference on Electro/Information Technology (EIT)*, (IEEE, 2018), pp. 0459–0464.

[2] Navneet Dalal and Bill Triggs, Histograms of oriented gradients for human detection, *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 1, (Ieee, 2005), pp. 886–893.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Faster r-cnn: towards real-time object detection with region proposal networks, *IEEE transactions on pattern analysis and machine intelligence*, Vol. 39, No. 6, (2016), pp. 1137–1149.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, You only look once: Unified, real-time object detection, *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2016), pp. 779–788.

[5] Bojan Pepik, Rodrigo Benenson, Tobias Ritschel, and Bernt Schiele, What is holding back convnets for detection?, *German conference on pattern recognition*, (Springer, 2015), pp. 517–528.

[6] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai, Diagnosing error in object detectors, *European conference on computer vision*, (Springer, 2012), pp. 340–353.

[7] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele, Citypersons: A diverse dataset for pedestrian detection, *CVPR*, (2017).

[8] Lu Qi, Jason Kuen, Jiuxiang Gu, Zhe Lin, Yi Wang, Yukang Chen, Yanwei Li, and Jiaya Jia, Multi-scale aligned distillation for low-resolution detection, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2021), pp. 14443–14453.

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun, Vision meets robotics: The kitti dataset, *The International Journal of Robotics Research*, Vol. 32, No. 11, (2013), pp. 1231–1237.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, Mask r-cnn, *Proceedings of the IEEE international conference on computer vision*, (2017), pp. 2961–2969.

[11] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon, Video panoptic segmentation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2020), pp. 9859–9868.

[12] P. Pérez, M. Gangnet, and A. Blake, Poisson image editing, *ACM Transactions on Graphics (SIGGRAPH'03)*, Vol. 22, No. 3, (2003), pp. 313–318.

[13] S Mahdi H Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yagiz Aksoy, Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2021), pp. 9685–9694.

[14] Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada, Openvslam: a versatile visual slam framework, *Proceedings of the 27th ACM International Conference on Multimedia*, (2019), pp. 2292–2295.

[15] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona, Fast feature pyramids for object detection, *IEEE transactions on pattern analysis and machine intelligence*, Vol. 36, No. 8, (2014), pp. 1532–1545.

[16] Stefan Walk, Nikodem Majer, Konrad Schindler, and Bernt Schiele, New features and insights for pedestrian detection, *2010 IEEE Computer society conference on computer vision and pattern recognition*, (IEEE, 2010), pp. 1030–1037.

[17] Payam Sabzmeydani and Greg Mori, Detecting pedestrians by learning shapelet features, *2007 IEEE Conference on Computer Vision and Pattern Recognition*, (IEEE, 2007), pp. 1–8.

[18] Joseph Redmon and Ali Farhadi, Yolov3: An incremental improvement, *arXiv preprint arXiv:1804.02767*, (2018).

[19] Ranjith K Dinakaran, Philip Easom, Ahmed Bouridane, Li Zhang, Richard Jiang, Fozia Mehboob, and Abdul Rauf, Deep learning based pedestrian detection at distance in smart cities, *Proceedings of SAI Intelligent Systems Conference*, (Springer, 2019), pp. 588–593.

[20] Yichao Yan, Jinpeng Li, Jie Qin, Song Bai, Shengcai Liao, Li Liu, Fan Zhu, and Ling Shao, Anchor-free person search, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2021), pp. 7690–7699.

[21] Rudy Bunel, Franck Davoine, and Philippe Xu, Detection of pedestrians at far distance, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, (IEEE, 2016), pp. 2326–2331.

[22] Xiaowei Zhang, Li Cheng, Bo Li, and Hai-Miao Hu, Too far to see? not really! —pedestrian detection with scale-aware localization policy, *IEEE transactions on image processing*, Vol. 27, No. 8, (2018), pp. 3703–3715.

[23] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona, Pedestrian detection: A benchmark, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, (IEEE, 2009), pp. 304–311.

[24] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, The cityscapes dataset for semantic urban scene understanding, *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2016), pp. 3213–3223.

[25] Mark Weber, Jun Xie, Maxwell Collins, Yukun Zhu, Paul Voigtlaender, Hartwig Adam, Bradley Green, Andreas Geiger, Bastian Leibe, Daniel Cremers, et al., Step: Segmenting and tracking every pixel, *arXiv preprint arXiv:2102.11859*, (2021).

[26] Yun Chen, Frieda Rong, Shivam Duggal, Shenlong Wang, Xinchen Yan, Sivabalan Manivasagam, Shangjie Xue, Ersin Yumer, and Raquel Urtasun, Geosim:

Realistic video simulation via geometry-aware composition for self-driving, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2021), pp. 7230–7240.

[27] Asati Minkesh, Kraisittipong Worranitta, and Miyachi Taizo, Extract and merge: Merging extracted humans from different images utilizing mask r-cnn, *arXiv preprint arXiv:1908.00398*, (2019).

[28] Yifan Wang, Andrew Liu, Richard Tucker, Jiajun Wu, Brian L Curless, Steven M Seitz, and Noah Snavely, Repopulating street scenes, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2021), pp. 5110–5119.

[29] Rong Zhi, Zijie Guo, Wuqiang Zhang, Baofeng Wang, Vitali Kaiser, Julian Wiederer, and Fabian B Flohr, Pose-guided person image synthesis for data augmentation in pedestrian detection, *2021 IEEE Intelligent Vehicles Symposium (IV)*, (IEEE, 2021), pp. 1493–1500.

[30] Markus Braun, Sebastian Krebs, Fabian B. Flohr, and Dariu M. Gavrila, Eurocity persons: A novel benchmark for person detection in traffic scenes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2019), pp. 1–1.

[31] Antonin Vobecky, Michal Uricár, David Hurych, and Radoslav Skoviera, Advanced pedestrian dataset augmentation for autonomous driving, *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, (2019), pp. 0–0.

[32] Lingzhi Zhang, Tarmily Wen, Jie Min, Jiancong Wang, David Han, and Jianbo Shi, Learning object placement by inpainting for compositional data augmentation, *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, (Springer, 2020), pp. 566–581.

[33] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár, Panoptic segmentation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2019), pp. 9404–9413.

[34] Hengshuang Zhao Rui Hu Min Bai Ersin Yumer Raquel Urtasun Yuwen Xiong, Renjie Liao, Upsnet: A unified panoptic segmentation network, (2019).

[35] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh, Openpose: realtime multi-person 2d pose estimation using part affinity fields, *IEEE transactions on pattern analysis and machine intelligence*, Vol. 43, No. 1, (2019), pp. 172–186.

[36] Yifan Wang, Brian L Curless, and Steven M Seitz, People as scene probes, *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, (Springer, 2020), pp. 438–454.

[37] Martin A Fischler and Robert C Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, Vol. 24, No. 6, (1981), pp. 381–395.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, Imagenet: A large-scale hierarchical image database, *2009 IEEE conference on computer vision and pattern recognition*, (Ieee, 2009), pp. 248–255.

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, Microsoft coco: Common objects in context, *European conference on computer vision*, (Springer, 2014), pp. 740–755.

[40] Zhen Zhu, Tengteng Huang, Baoguang Shi, Miao Yu, Bofei Wang, and Xiang Bai, Progressive pose attention transfer for person image generation, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2019), pp. 2347–2356.

[41] Oran Gafni, Oron Ashual, and Lior Wolf, Single-shot freestyle dance reenactment, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2021), pp. 882–891.

[42] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi, Openpifpaf: Composite fields for semantic keypoint detection and spatio-temporal association, *arXiv preprint arXiv:2103.02440*, (2021).

[43] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun, Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, *arXiv preprint arXiv:1907.01341*, (2019).

List of Figures

List of Tables

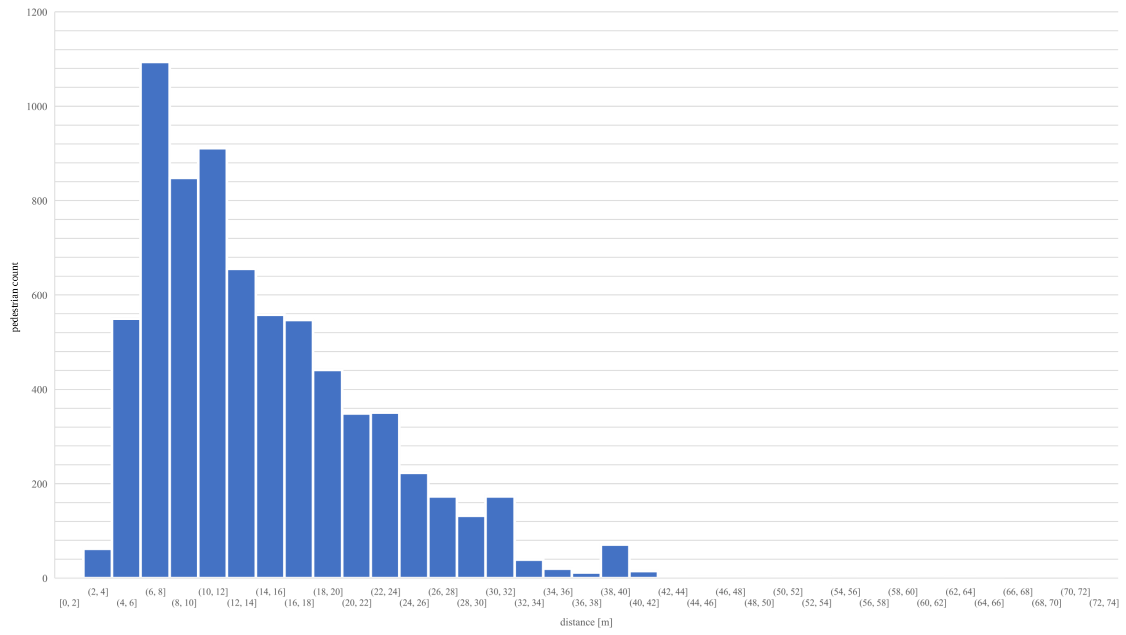


Figure 1.1: Distribution of annotated pedestrian distances in KITTI object tracking benchmark. The data of pedestrians' real distance from camera in meter are collected from the ground truth annotations of KITTI object tracking dataset. Only 0.38% (44 out of 11470) of the annotated pedestrians are over 50 meters away from the camera.

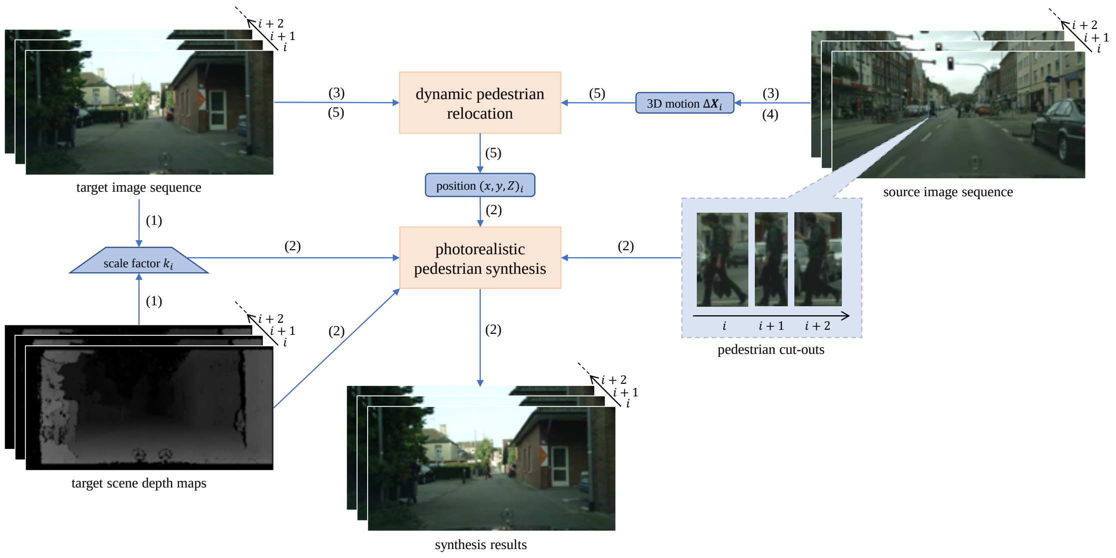


Figure 3.1: An overview of the *pedestrian relocation* pipeline. (1) The scale regression module samples "ground" pixels from a depth map and calculate a scale factor k that best describes the scales of pedestrians throughout the scene. (2) The photorealistic pedestrian synthesis module synthesizes a given cut-out pedestrian into a specified target image with correct scale and occlusion. (3) The camera ego-motion estimation module estimates inter-frame relative camera poses. (4) The pedestrian motion estimation module calculates 3D motions of pedestrians in their source scenes. (5) The final *dynamic pedestrian relocation* module aggregates the 3D motions and the camera ego-motion to update the location of the pedestrian.

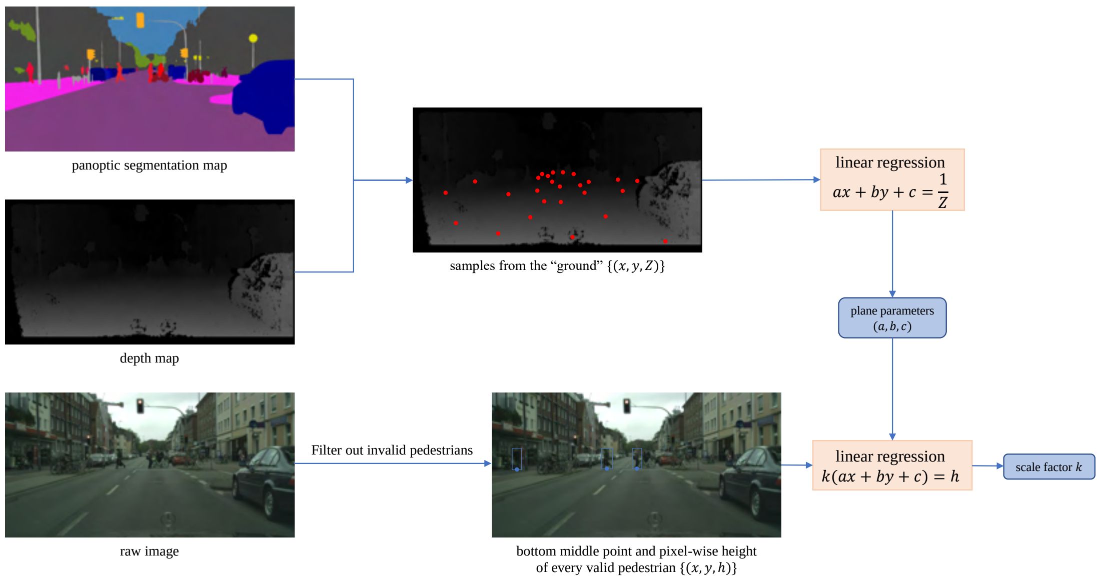

panoptic segmentation map
depth map
samples from the "ground" $\{(x, y, Z)\}$
linear regression
$ax + by + c = \frac{1}{Z}$
plane parameters
(a, b, c)
raw image
Filter out invalid pedestrians
bottom middle point and pixel-wise height of every valid pedestrian $\{(x, y, h)\}$
linear regression
$k(ax + by + c) = h$
scale factor k

Figure 3.2: An outline of the scale regression module. (1) A binary mask indicating "ground" regions is generated from the semantic segmentation mask. (2) Inside the masked region, a certain number of (x, y, Z) samples are obtained from the depth map. (3) The ground parameters (a, b, c) is obtained through linear regression of Eq. 3.4. In parallel, (4) keypoint detection is performed to filter out invalid pedestrians for scale regression, and (5) the bottom middle point (x, y) and pixel-wise height h are collected from each valid pedestrian. Finally, (6) the scale factor k is calculated through linear regression of Eq. 3.6

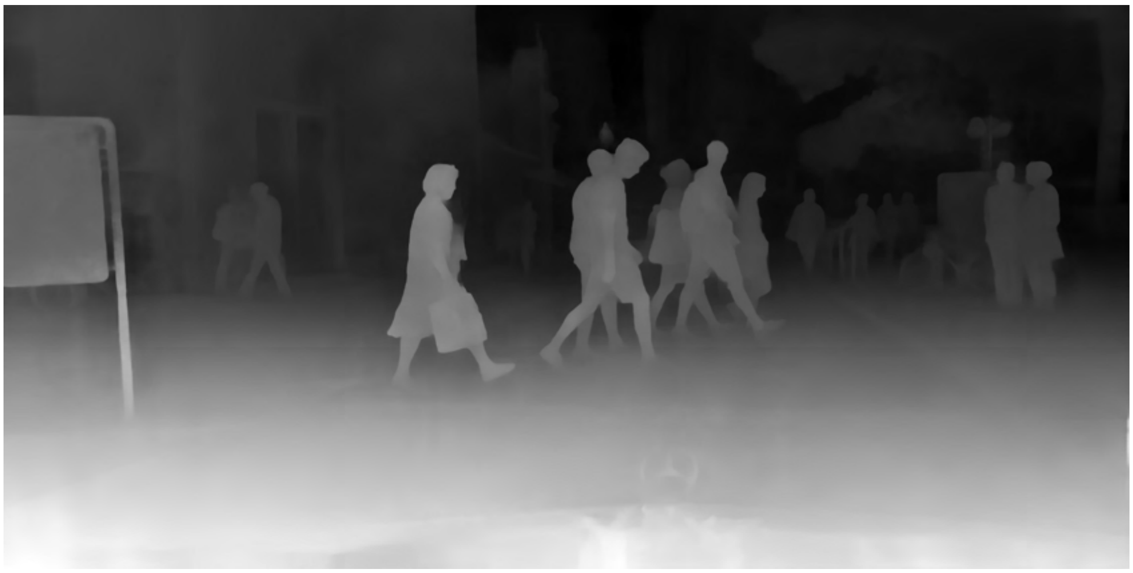
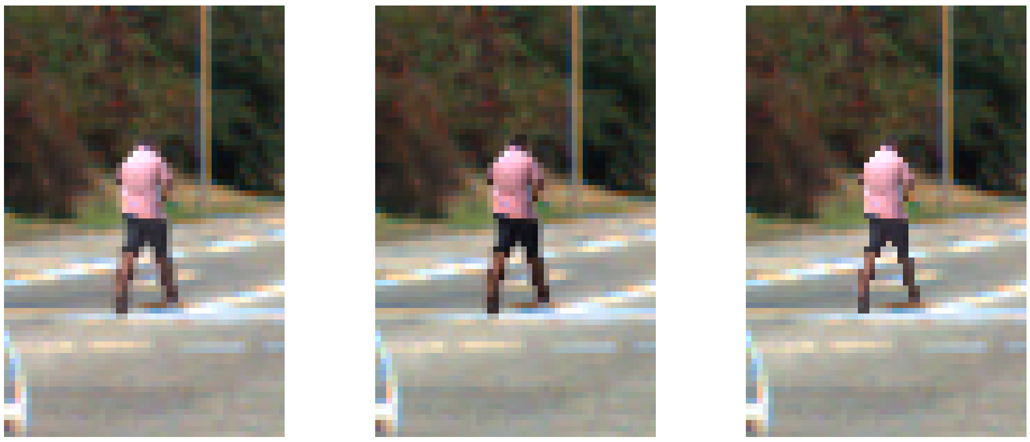


Figure 3.3: An example of high-resolution depth maps estimated by MergeNet [13]. We use MiDaS [43] as the backbone of MergeNet.



(a) Color shift+edge blurring (b) Only edge blurring (c) Only color shift

Figure 3.4: Example of photorealistic pedestrian synthesis

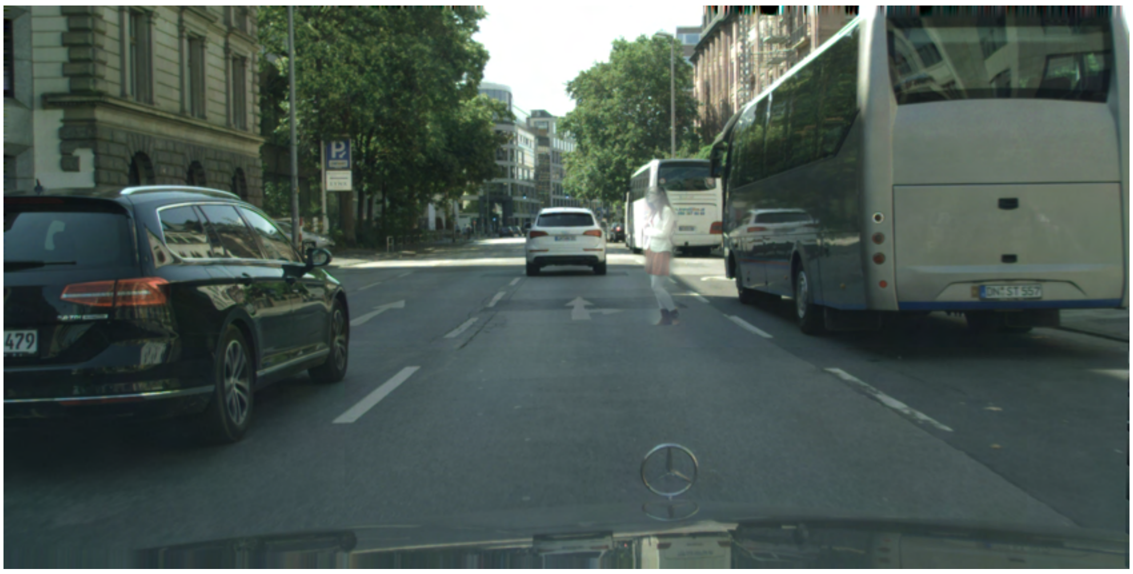
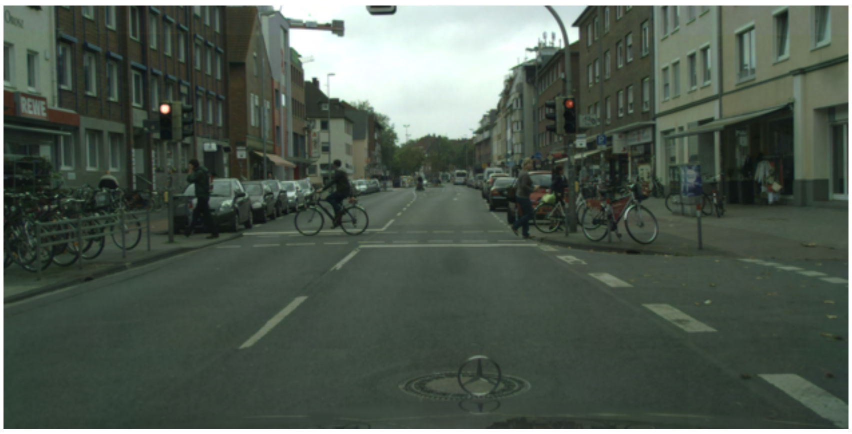


Figure 3.5: An example of Poisson Image Editing results. The upper half of the synthesized pedestrian is eroded by the white background (bus), while the lower half is eroded by the gray background (road).

(a) The first frame of a 6-frame sequence from Cityscapes [24]


(b) The final frame of the same sequence

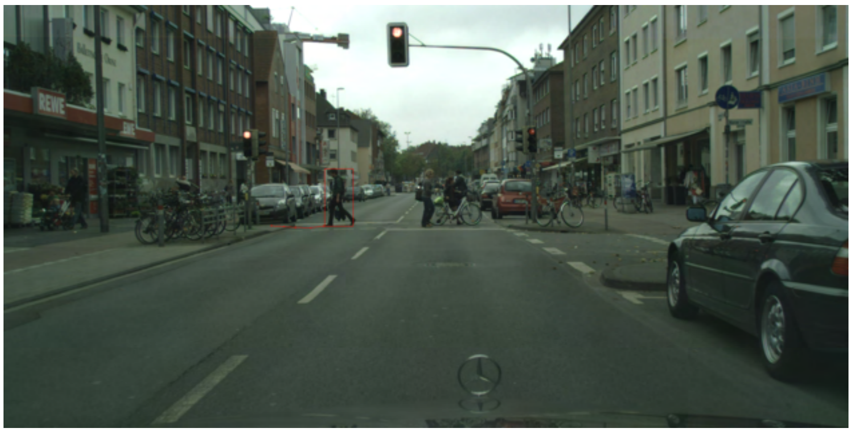

(c) Trajectory of a pedestrian

Figure 3.6: A visualization of pedestrian motion. To counteract influence of camera ego-motion, we unify all 3D positions of a pedestrian across frames into the camera coordinate system of the first frame. As shown in (c), we can get a trajectory of the pedestrian by connecting these transformed 3D coordinates.

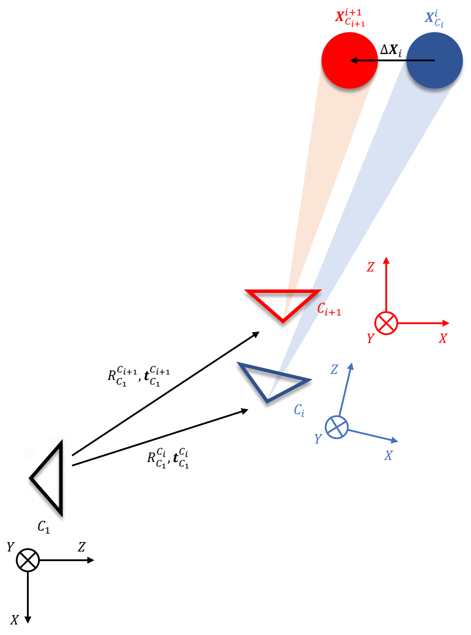


Figure 3.7: Camera coordinate systems. We show an imaginary overhead view to illustrate relationships among different camera coordinate systems. The triangles are cameras. The circles denote the positions of a pedestrian in two consecutive frames i and $i+1$. Different colors stand for different camera coordinates.

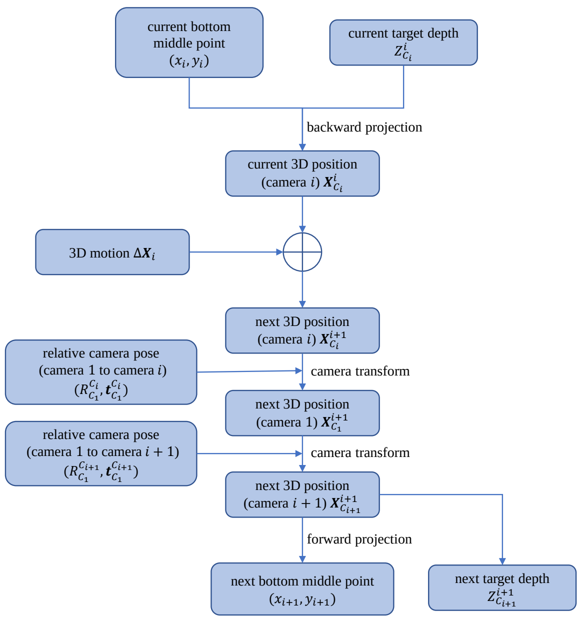

current bottom middle point (x_i, y_i)
current target depth $Z_{C_i}^{i}$
backward projection
current 3D position (camera i) $X_{C_i}^{i}$
3D motion ΔX_i
next 3D position (camera i) $X_{C_i}^{i+1}$
relative camera pose (camera 1 to camera i) $(R_{C_1}^{C_i}, t_{C_1}^{C_i})$
camera transform
next 3D position (camera 1) $X_{C_1}^{i+1}$
relative camera pose (camera 1 to camera $i+1$) $(R_{C_1}^{C_{i+1}}, t_{C_1}^{C_{i+1}})$
camera transform
next 3D position (camera $i+1$) $X_{C_{i+1}}^{i+1}$
forward projection
next bottom middle point (x_{i+1}, y_{i+1})
next target depth $Z_{C_{i+1}}^{i+1}$

Figure 3.8: Outline of the *dynamic pedestrian relocation* module.

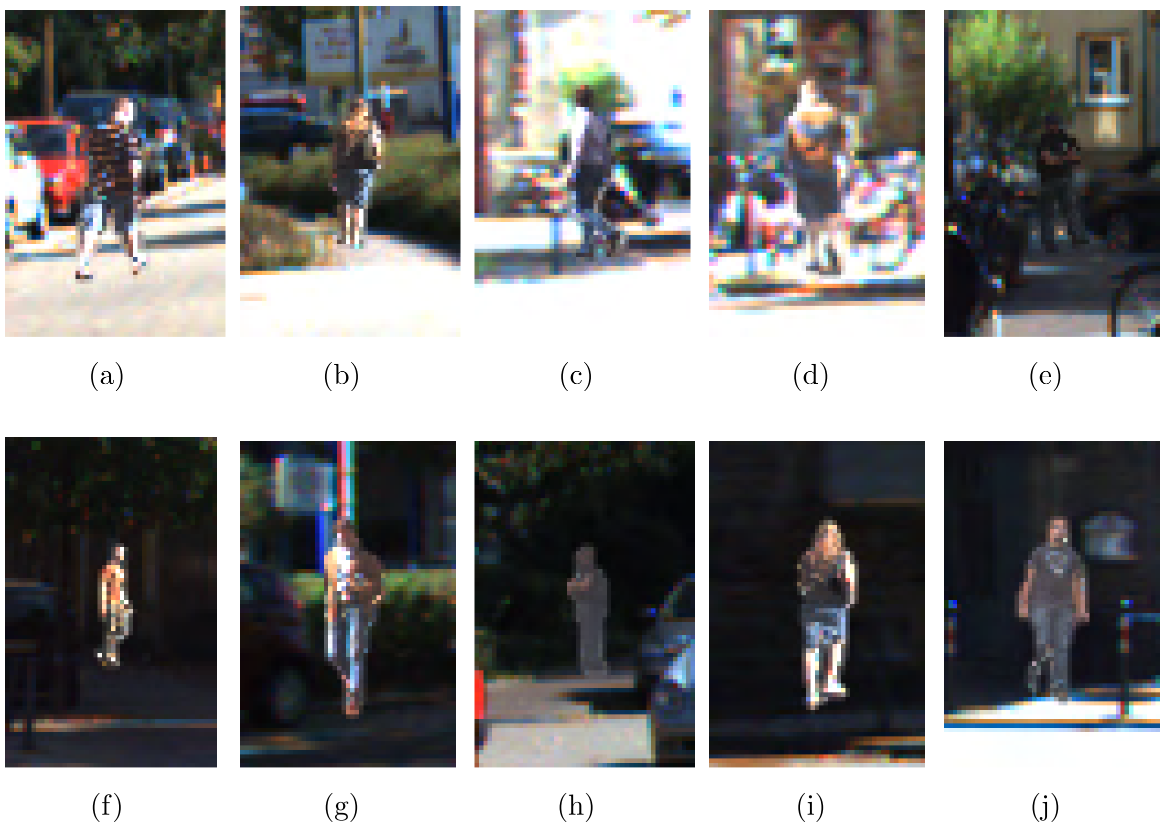


Figure 4.1: Success and failure cases of photorealistic pedestrian synthesis. (a)-(e) are success cases and (f)-(j) are failure cases.

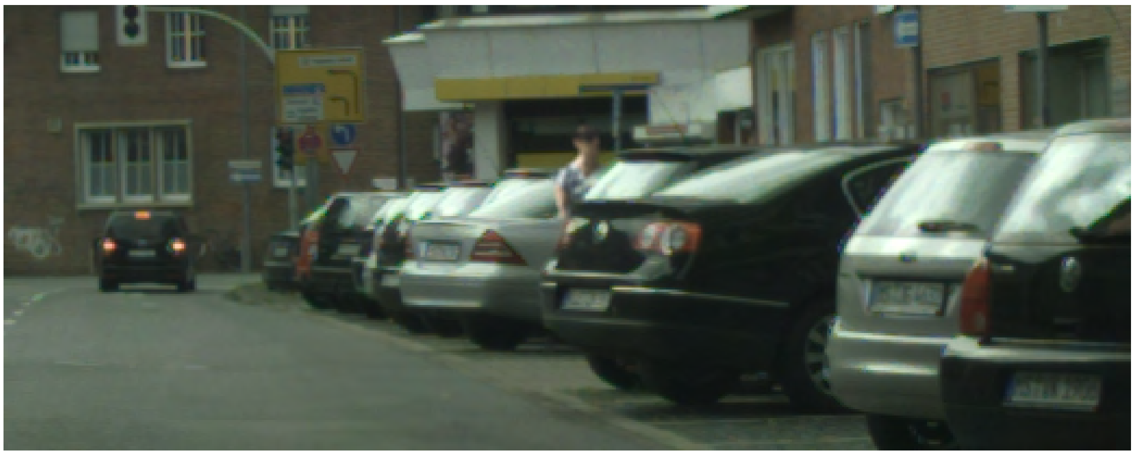
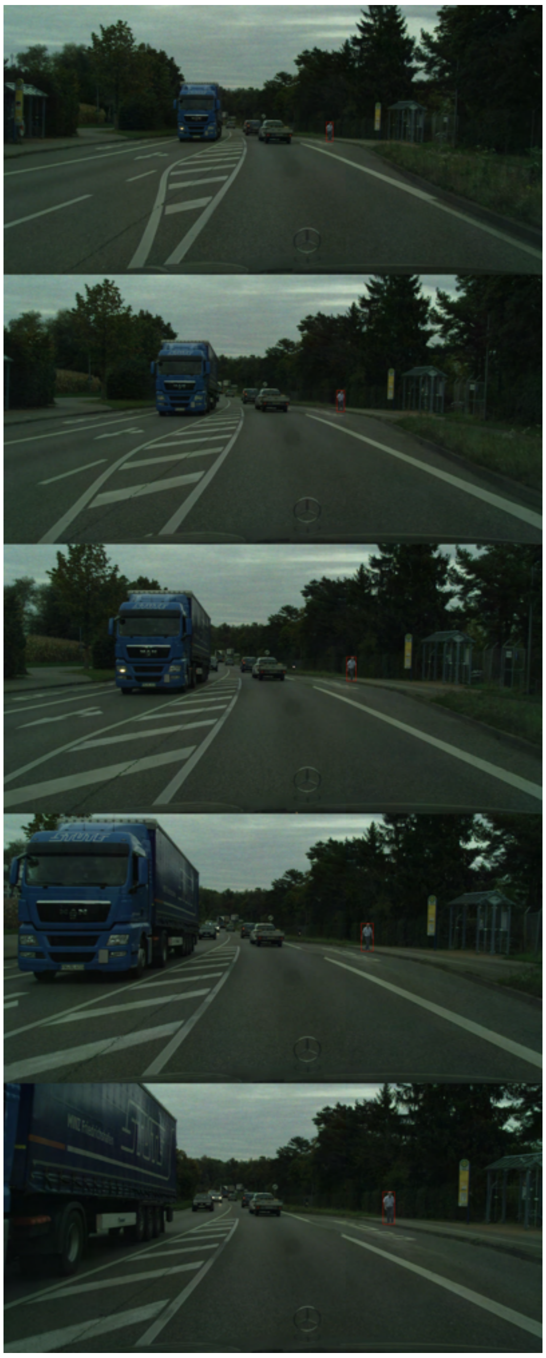


Figure 4.2: Example of generated occlusion


(a) Success case

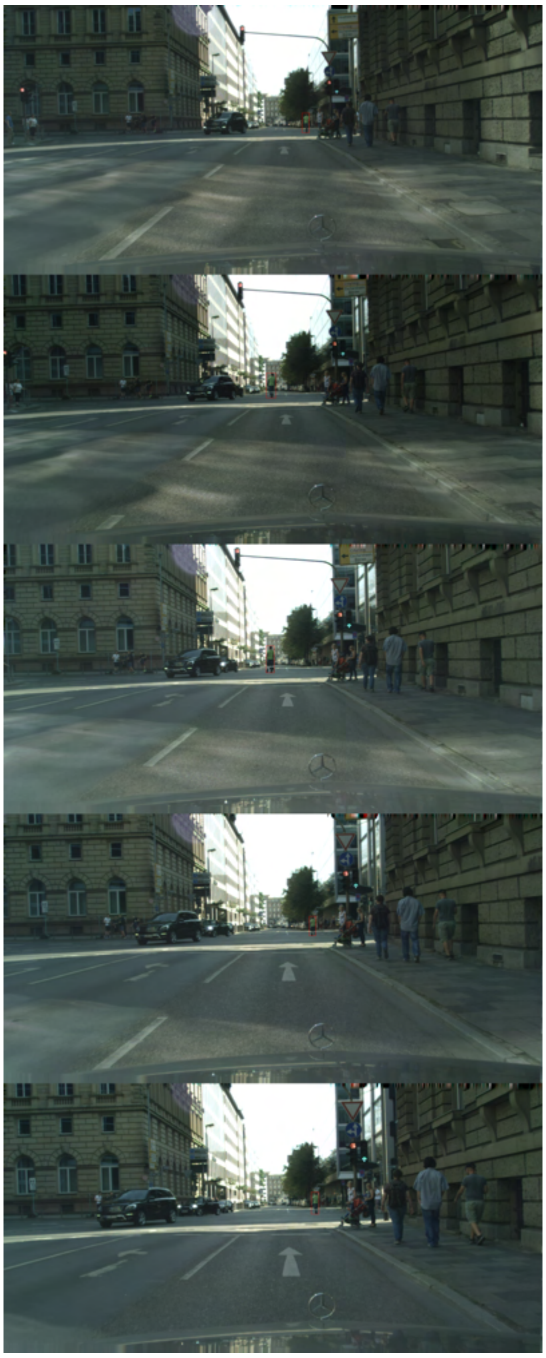
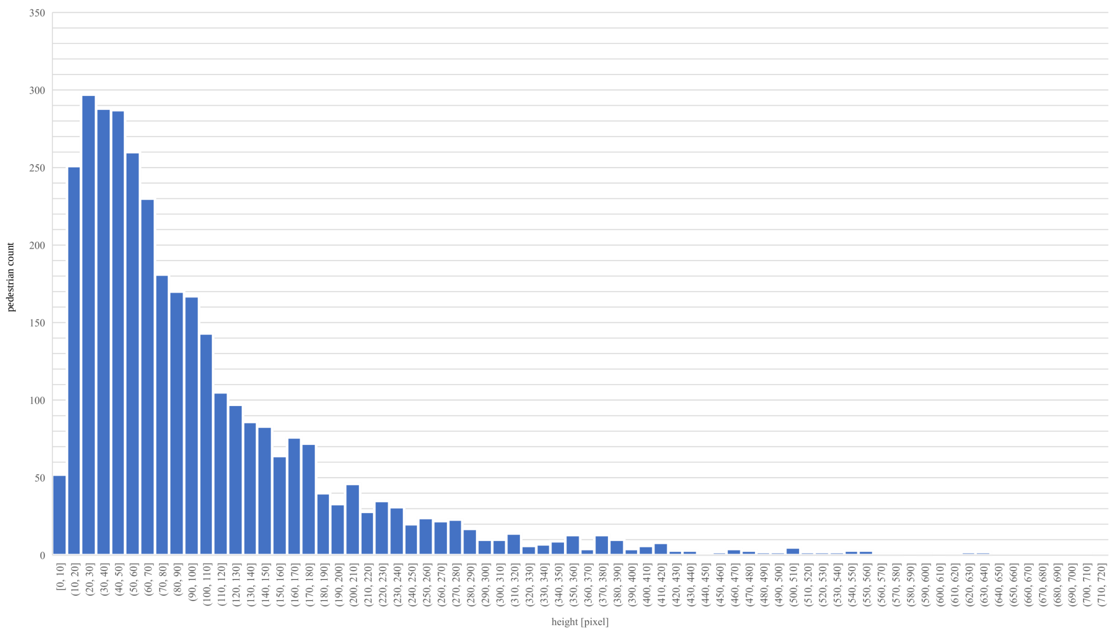
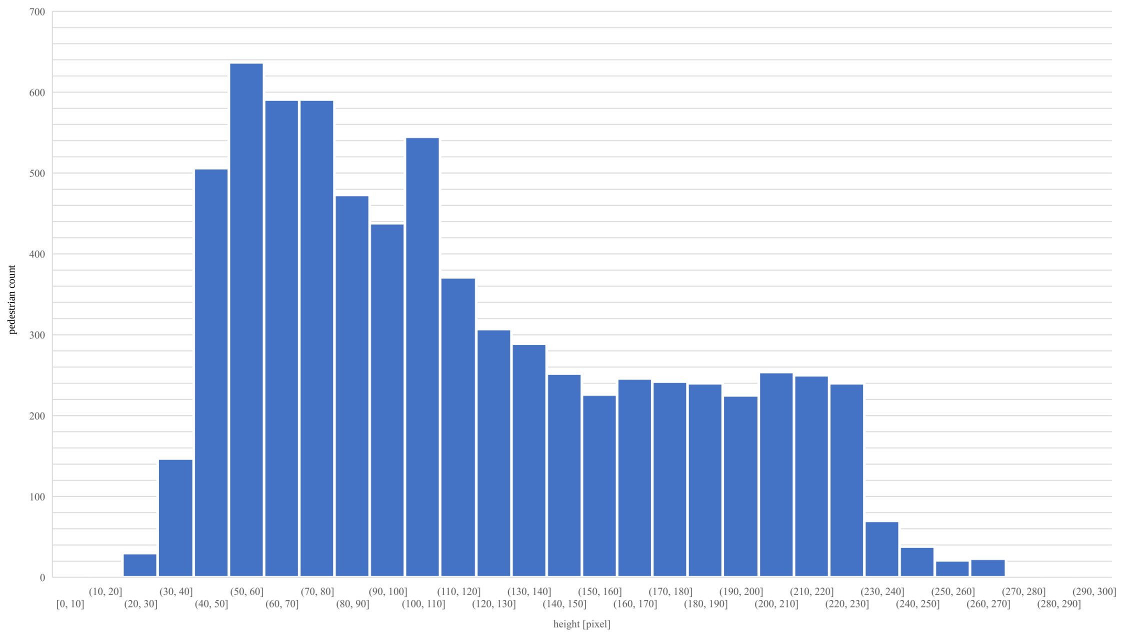

(b) Failure case

Figure 4.3: Success and failure cases of *dynamic pedestrian relocation*. The relocated pedestrians are marked by red bounding boxes.

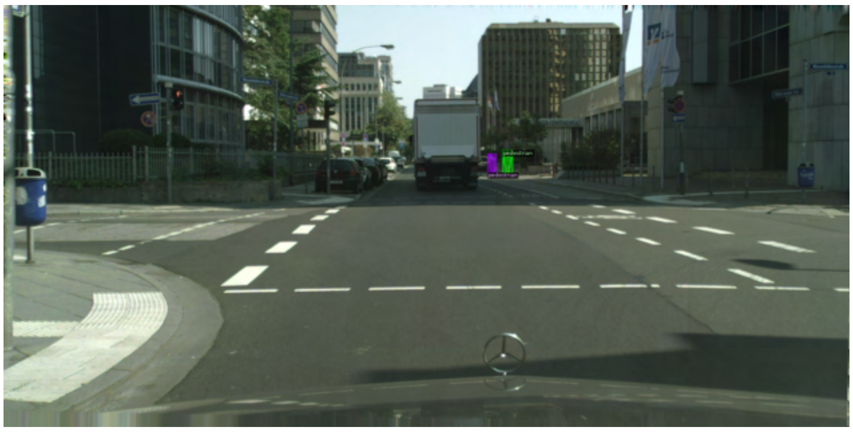


(a) Cityscapes validation dataset [24]

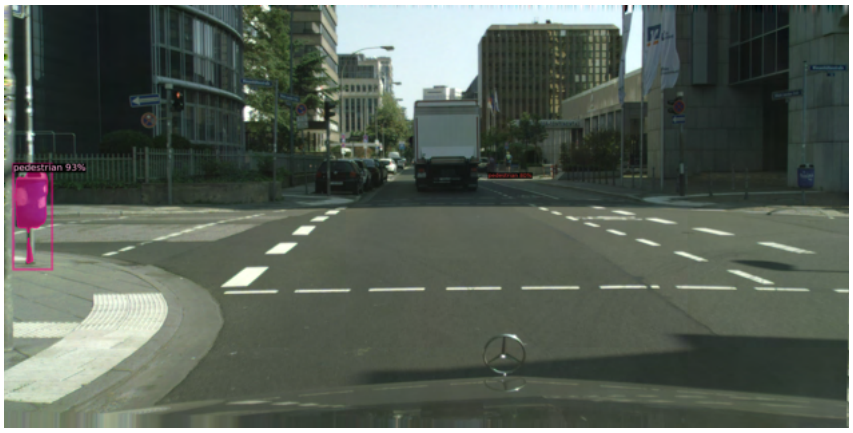


(b) KITTI-STEP [25]

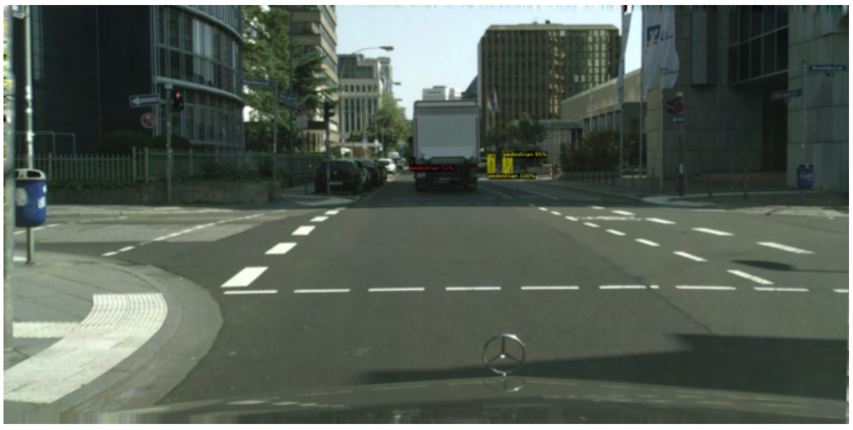
Figure 4.4: Histograms of pedestrian heights in pixel



(a) Ground truth



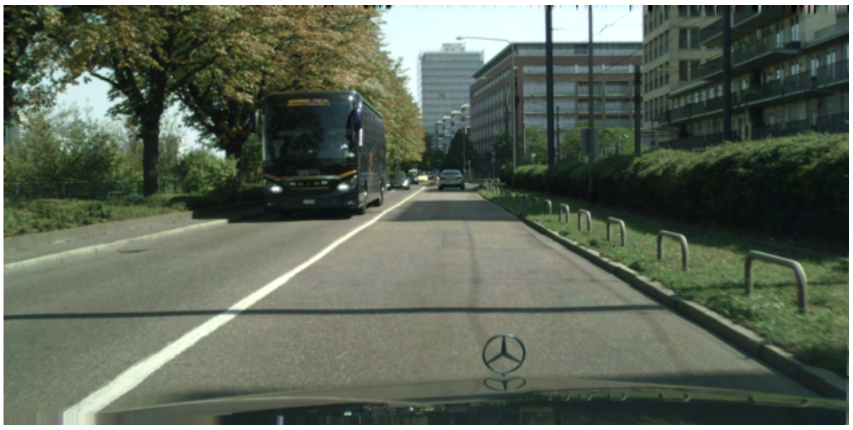
(b) Prediction made by the model trained without data augmentation

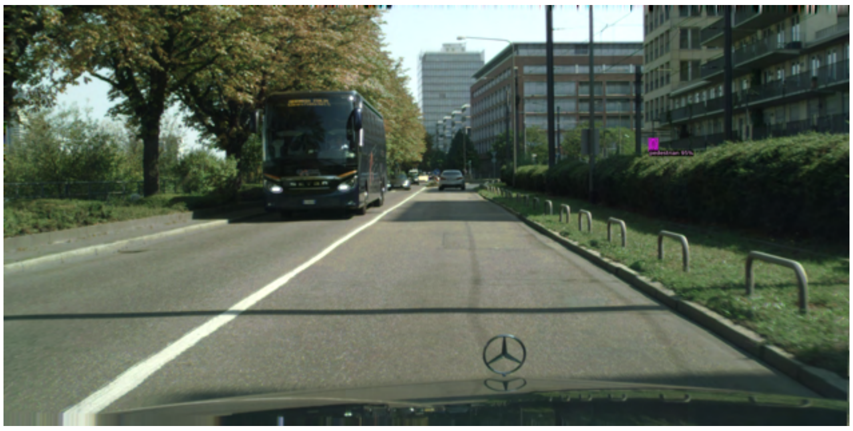


(c) Prediction made by the model trained with data augmentation

Figure 4.5: A comparison of a pair of predictions made by the two models. There are two pedestrians who are almost equally far from the camera. The model trained with data augmentation is able to detect both of them with higher confidence, while the model trained without data augmentation detects only one of them with lower confidence.

(a) Ground truth


(b) Prediction made by the model trained without data augmentation



(c) Prediction made by the model trained with data augmentation

Figure 4.6: An example of misdetections made only by the model trained with data augmentation. It mistakes a traffic sign for a pedestrian with high confidence.

Task	Detection			Instance Segmentation		
Metric	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
w/o *Pedestrian Relocation*	4.800	15.214	1.961	3.671	12.291	0.657
w/ *Pedestrian Relocation*	5.917	16.640	2.452	3.931	13.379	0.658
Improvement	1.117	1.426	0.491	0.260	1.088	0.001

Table 4.1: Quantitative results on Cityscapes validation dataset [24]. We train Mask R-CNN [10] on KITTI-STEP [25] either with or without pedestrian relocation as data augmentation. Then, we fine-tune both models on Cityscapes training dataset. Finally, we evaluate and compare their performance for *far pedestrian* detection and instance segmentation on the validation set. The evaluation is based on official COCO metrics [39].