Learning to Detect Pedestrian Flow in Traffic Intersections from Synthetic Data

Abhijit Baul
The University of Texas Rio Grande Valley

LEARNING TO DETECT PEDESTRIAN FLOW IN TRAFFIC INTERSECTIONS FROM SYNTHETIC DATA

A Thesis

by

ABHIJIT BAUL

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING

May 2021

Major Subject: Electrical Engineering

LEARNING TO DETECT PEDESTRIAN FLOW IN TRAFFIC INTERSECTIONS FROM SYNTHETIC DATA

A Thesis
by
ABHIJIT BAUL

COMMITTEE MEMBERS

Dr. Weidong Kuang
Chair of Committee

Dr. Heinrich Foltz
Committee Member

Dr. Jingru Zhang
Committee Member

Dr. Jia Chen
Committee Member

May 2021

ABSTRACT

Baul, Abhijit, Learning to Detect Pedestrian Flow in Traffic Intersections from Synthetic Data. Master of Science in Engineering (MSE), May, 2021, 47 pp., 6 tables, 25 figures, 48 references.

Pedestrian flow Detection at a traffic-intersection has always been a challenging task. Challenges include different weather conditions, different pedestrian density, occlusions, lack of available data and so on. Emergence of deep learning and computer vision algorithms has shown promises to deal with these problems. Most of the recent works only focus on either detecting combined pedestrian flow or counting the total number of pedestrian. In this research work, we are proposing an end-to-end Deep Learning framework to detect not only combined pedestrian flow but also pedestrian flow in different directions. For this purpose, we have created a synthetic dataset from a video game. Our GTA-Intersection dataset has 75 cross-road pedestrian scene with different pedestrian density and camera height. Also, we have collected several videos from the street to evaluate our proposed method. After that, we proposed a Pedestrian Flow Inference Model (PFIM). Our contributions are, 1) we are introducing a synthetic pedestrian dataset that we have created using a video game and a real-world dataset we have collected from the street. Our dataset has small, medium and high density pedestrians crossing a crossroad, captured from different camera height, 2) We have proposed a Pedestrian Flow Inference Model (PFIM) that is trained on the synthetic dataset first and then is tested extensively on our real-world dataset. While testing on real-world dataset, we have embraced domain adaptation to reduce the domain gap between synthetic data and real-world data. Our proposed Pedestrian Flow Inference Model (PFIM) can detect pedestrian density and flow regardless the height of the camera in three different directions - from left to right direction, from right to left direction and total. Combining all, It has successfully tackled the challenges mentioned above and achieved state-of-the-art performances.

DEDICATION

I want to dedicate my research work to Dr. Andrew Ng who inspired me to do research in deep learning and computer vision. I would also like to mention Dr. Ian Goodfellow who proposed the idea of Generative Adversarial Network. I also like to thank my parents for their support throughtout my life.

ACKNOWLEDGMENTS

I will always be grateful to Dr. Weidong Kuang, the chair of my thesis committee, for all his mentoring and guidance. I am grateful to Dr. Hongkai Yu for his idea of this research work, assessing the performance of my work every week. Many thanks go as well to my thesis committee members, Dr. Jingru Zhang, Dr. Heinrich Foltz, & Dr. Jia Chen. Their advice, and comments helped to improve my work. I would also like to thank my lab mates at the UTRGV Machine Learning and Data Science Lab who helped me to find out interesting ideas that was helpful throughout the research.

TABLE OF CONTENTS

Page

ABSTRACT . . . iii
DEDICATION . . . iv
ACKNOWLEDGMENTS . . . v
TABLE OF CONTENTS . . . vi
LIST OF TABLES . . . viii
LIST OF FIGURES . . . ix
CHAPTER I. INTRODUCTION . . . 1
- 1.1 Pedestrian Flow . . . 1
- 1.2 Applications . . . 2
- 1.3 Overview of the Proposed Framework . . . 2
- 1.4 Outline of Thesis . . . 4

CHAPTER II. REVIEW OF LITERATURE . . . 5
- 2.1 Feature-based Detection . . . 5
- 2.2 Deep CNN-based Detection . . . 5
- 2.3 Pedestrian Datasets . . . 6
  - 2.3.1 Real-World Datasets . . . 6
  - 2.3.2 Synthetic Datasets . . . 7

CHAPTER III. GTA-INTERSECTION DATASET . . . 9
- 3.1 Creating Traffic-Intersection Scenes . . . 9
- 3.2 Examples from GTA-Intersection Dataset . . . 12
- 3.3 Real-Intersection Dataset . . . 12

CHAPTER IV. METHODOLOGY . . . 16
- 4.1 Generating Optical Flow . . . 16
- 4.2 Pedestrian Counting Net . . . 18
  - 4.2.1 Review on AlexNet . . . 19
  - 4.2.2 Review on YOLOv3 . . . 19
- 4.3 Loss Function for the training of Pedestrian Counting Net . . . 21

4.4 Pedestrian Flow Inference Model 21
CHAPTER V. DOMAIN ADAPTATION 23
5.1 Cycle GAN 23
5.2 Structural Similarity Index Measure Loss 25
5.3 Losses for Domain Adaptation 26
5.3.1 Some Examples of Structure-Aware Domain Adaptation 27
CHAPTER VI. EXPERIMENT 29
6.1 Experimental Setting for Pedestrian Counting Net 29
6.2 Experimental Setting for Cycle GAN 29
6.3 Comparison Methods 30
6.4 Experimental Results 30
CHAPTER VII. DISCUSSION 36
7.1 Summary of Findings 36
CHAPTER VIII. CONCLUSION 41
8.1 Conclusion from the results 41
8.2 Future Work 42
BIBLIOGRAPHY 43
BIOGRAPHICAL SKETCH 47

LIST OF TABLES

Page

Table 3.1: Details of the datasets used in this experiment: image numbers, pedestrain density, and image resolution. There are 75 videos in the GTA-Intersection and 3 videos in Real-Intersection. 15

Table 6.1: Performance on the Real-Intersection dataset of the proposed method without using any real-data for fine-tuning. 31

Table 6.2: Performance on the Real-Intersection dataset of the proposed method using a small portion of real-data for fine-tuning. 31

Table 6.3: Performance of our proposed method on the Subset1 @citesnum[45] using a small portion of real-data for fine-tuning. 32

Table 6.4: Performance of our proposed method on the Subset2 @citesnum[8] dataset using a small portion of real-data for fine-tuning. 32

Table 6.5: Performance of our proposed method on Subset3 (we have collected from the street) using a small portion of real-data for fine-tuning. 33

LIST OF FIGURES

Page

Figure 1.1: Proposed deep learning framework to detect the pedestrian flow parameters: firstly, the proposed Pedestrian Counting Network (PCNet) predicts the pedestrian counting numbers moving from left to right, moving from right to left, and total as N_L, N_R, N_T; secondly, based on the predicted N_L, N_R, N_T, an Pedestrian Flow Inference Model (PFIM) is designed to estimate the pedestrian flow parameters: a) pedestrian flow and density from left to right direction, b) pedestrian flow and density from right to left direction, and c) combined pedestrian flow and density in both direction. 4

Figure 2.1: An image frame from Kitti @citesnum[15] dataset 7

Figure 2.2: An image frame from Cityperson @citesnum[43] dataset 7

Figure 3.1: Script Hook V @citesnum[2] 10

Figure 3.2: Scene Director V @citesnum[3] 11

Figure 3.3: GTA-Intersection dataset: 1^{st} row: small pedestrian density (0-10), 2^{nd} row: medium pedestrian density (10-20), c) 3^{rd} row: high pedestrian density (20-40); each column corresponds to different camera height. 11

Figure 3.4: Examples from GTA-Intersection dataset. 12

Figure 3.5: Examples from GTA-Intersection dataset. 13

Figure 3.6: Examples from GTA-Intersection dataset. 13

Figure 3.7: Examples from GTA-Intersection dataset. 14

Figure 3.8: Real-Intersection dataset: 1^{st} row: frames collected from @citesnum[45], 2^{nd} row: frames collected from @citesnum[8], c) 3^{rd} row: frames collected from the street from us. 14

Figure 4.1: Optical flow generation 16

Figure 4.2: Optical flow generation (Collected from Internet) 17

Figure 4.3: Detailed Two-Branch CNN framework 18

Figure 4.4: AlexNet framework. Image credit @citesnum[21]. 19

Figure 4.5: DarkNet framework. Image credit @citesnum[27] 20

Figure 5.1: CycleGAN @citesnum[48] framework 24

Figure 5.2: Structural Similarity Index Measure (SSIM) between Synthetic Images and Reconstructed Synthetic Images from CycleGAN 26

Figure 5.3: Domain adaptation example 27

Figure 5.4: Domain adaptation example 28

Figure 6.1: Pedestrian counting results by our proposed PCNet framework: (a) Detection (N_L: 33, N_R: 34, N_T: 68), Ground Truth (N_L: 35 , N_R: 37, N_T: 72), (b) Detection (N_L: 34, N_R: 36, N_T: 71), Ground Truth (N_L: 36, N_R: 38, N_T: 74), (c) Detection (N_L: 5, N_R: 4, N_T: 10), Ground Truth (N_L: 6, N_R: 5, N_T: 11), (d) Detection (N_L: 6, N_R: 8, N_T: 15), Ground Truth (N_L: 7, N_R: 9, N_T: 16), (e)Detection (N_L: 19, N_R: 5, N_T: 24), Ground Truth (N_L: 18, N_R: 5, N_T: 23), (f) Detection (N_L: 20, N_R: 5, N_T: 25), Ground Truth (N_L: 20, N_R: 6, N_T: 26). 34

Figure 6.2: Detected pedestrian flow and density on different directions by our proposed PFIM framework : a) flow and density from left to right, b) flow and density from right to left, and c) combined pedestrian flow and density in both directions. 35

Figure 7.1: Performance of PCNet 37

Figure 7.2: Performance of PCNet 38

Figure 7.3: Performance of PCNet 39

CHAPTER I

INTRODUCTION

Pedestrian flow detection has been a real-world challenge since 1960 or before that. In the beginning, there were many mathematical models that tried to estimate the pedestrian volume and density from previously collected data. With the advances of technology, there are many camera-based surveillance system available that try to detect pedestrian flow in a traffic intersection. But the challenge of detecting pedestrian flow by an automatic system is still unresolved. There are some algorithms that extracted features from an input image frame and tried to detect pedestrian flow without human-supervision. With the recent success of Deep learning and Computer Vision in solving real-world problems, this research work tried to find out how much this new approach can contribute to automatic pedestrian flow detection.

1.1 Pedestrian Flow

The term Pedestrian Flow can be defined as the number of pedestrian passing through a certain area, in a certain amount of time. More elaborately [14],

$$V = \frac{S}{M}. \tag{1.1}$$

Where V is the volume of flow, S is the average speed of the pedestrians and M is pedestrian area module, $M = \frac{1}{Density}$. In real-world condition, we can easily estimate the pedestrian area. However, estimating the pedestrian density is a challenging task. It changes with daytime, weather and many other factors.

1.2 Applications

Detecting Pedestrian Flow has many applications including crowd analysis [34, 41], video surveillance, traffic flow monitoring, planning for future infrastructure development near roads and so on. Crowd analysis tries to interpret data on the natural movement of groups or object, more elaborately, human movements. Video Surveillance tries to detect aberrant behavior in a certain place. All these can be benefited from detecting pedestrian flow accurately. Traffic flow monitoring is the base case where pedestrian flow detection can play the most important role. Also, developing an infrastructure near roads require analysis of pedestrian flow near that area.

1.3 Overview of the Proposed Framework

One of the main challenges in pedestrian flow detection is the lack of a high-resolution dataset. To resolve that issue, at first, we tried to create a synthetic dataset. For this purpose, we selected the GTA V (Grand Theft Auto V) game. The motivation behind selecting this game are, 1) The game is open source and available for research, 2) There are some game plug-ins and libraries available that can be used for creating an ideal crossroad-pedestrian scenario. These libraries also allows to capture videos from different height and angle and create different weather pattern. For simplicity, we have only considered day time and clear environment in this research. However, it can be extended for different day times and weather conditions. 3) As the number of pedestrians in each scene is preset by us, we are fully aware of the number of pedestrian in each scene, hence there is no chance of annotation error. In addition to that, the resolution is very high which is not a general case for most of the available datasets mentioned above. Detailed explanation is provided in chapter VI.

After creating the synthetic dataset, we generated optical flow from each pair of consecutive video frames. Optical flow [17] is a state-of-the-art algorithm for analysing different types of motion and velocity in a specific image frame. Our research has shown that the addition of optical flow with video frames combined improved the accuracy for detecting the flow.

Our proposed Pedestrian Flow Inference Model (PFIM) consists of a deep CNN architecture, Pedestrian Counting Net (PCNet). It is a two-branch deep CNN architecture where a pretrained AlexNet [21] was used for feature extraction. Initially, we divided our synthetic dataset into train and validation set and trained the PCNet which has shown tremendous success in counting pedestrian numbers in each frame. The output from PCNet goes into PFIM that determines the pedestrian flow and density. Then, we began our evaluation on real-world datasets. But unfortunately, we have only found two real-world dataset that is similar to our synthetic dataset - [8, 45]. We have selected some video frames that are close to our synthetic dataset but still the resolution was not good and there was severe occlusion especially the frames adopted from [45]. Hence, we decided to collect some real world cross-road scene from the street. We have also included the description of the real-world dataset with our synthetic dataset.

In order to reduce the gap between virtual domain and real-world domain, we embraced domain adaptation methods like Cycle GAN [48] and measured the quality of the reconstructed images using Structuaral Similarity Index Measure (SSIM) [47]. Both of them have been used by some recent works [39, 23, 37] for crowd analysis and counting and has shown promising performance in reducing the domain gap. We have evaluated the performance of our proposed two-branch Pedestrian Counting Net (PCNet) in two ways- without and with fine-tuning using real-world data. Our goal was to observe the introduction of synthetic data - whether it can successfully resemble the real-world scenario or not. To accomplish that, initially we observed PCNet performance only trained on 1) Synthetic data or trained on 2) Synthetic data and then fine-tuned on cross-domain frames achieved by Cycle GAN. We then evaluated the performance of both 1) and 2) method by directly detecting the pedestrian flows on real-world datasets. In addition to that, we choose a small percentage of real-world dataset to fine-tune both 1) and 2) method and then measured the performance. Our proposed framework achieved state-of-the-art performances in detecting the pedestrian flow. More detailed evaluation is available in section VI.

Fig. 1.1 demonstrates our complete framework. Briefly, this research's contributions are:

1) We have created a synthetic annotated dataset named GTA-Intersection dataset using the GTA V video game.

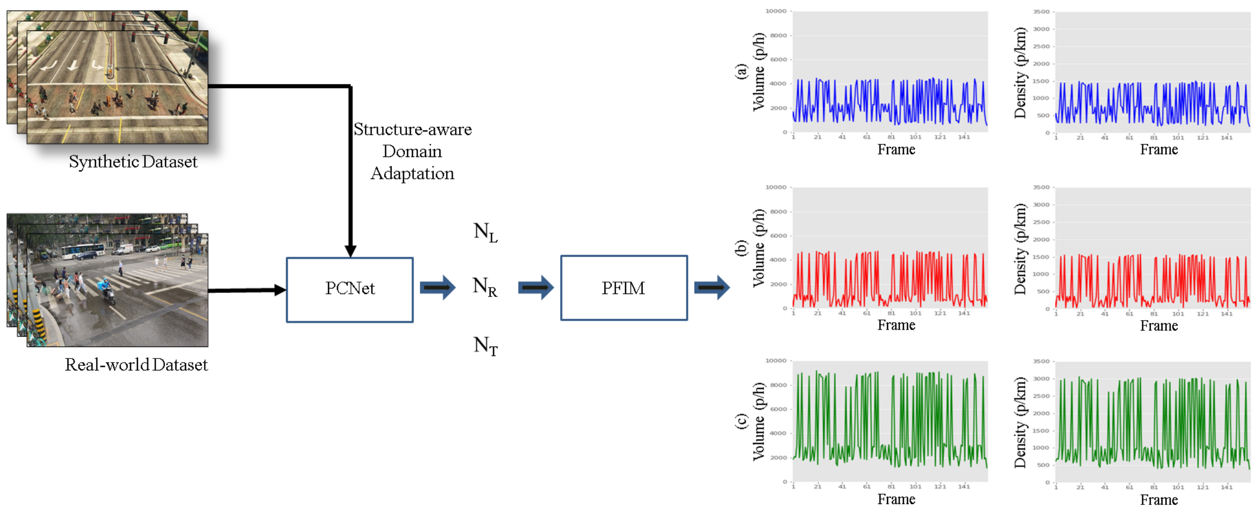


Figure 1.1: Proposed deep learning framework to detect the pedestrian flow parameters: firstly, the proposed Pedestrian Counting Network (PCNet) predicts the pedestrian counting numbers moving from left to right, moving from right to left, and total as N_L, N_R, N_T; secondly, based on the predicted N_L, N_R, N_T, an Pedestrian Flow Inference Model (PFIM) is designed to estimate the pedestrian flow parameters: a) pedestrian flow and density from left to right direction, b) pedestrian flow and density from right to left direction, and c) combined pedestrian flow and density in both direction.

2) We have proposed a Pedestrian Flow Inference Model (PFIM) that is able to detect pedestrian flow and density in three different directions - from left to right, from right to left and the total flow and density.

3) We have embraced domain adaptation techniques like Cycle GAN and image reconstruction metric Structural Similarity Index Measure(SSIM) and achieved state-of-the-art performances using our PFIM framework.

1.4 Outline of Thesis

In chapter II, there is discussion about the previous and relevant research works in pedestrian flow detection. The chapter III contains the description of GTA-Intersection dataset. Chapter IV describes our proposed Pedestrian Counting Net (PCNet) and Pedestrian Flow Inference Model (PFIM). In chapter V, we discussed domain adaptation. We discussed the experiments and results in section VI.

CHAPTER II

REVIEW OF LITERATURE

The literature review regarding pedestrian flow detection consists of three sections - The first one is about the algorithms that require hand-crafted features and the second section is about algorithms that used Deep Convolutional Neural Network framework. Since, one of the purposes of this research work focused on the utilization of synthetic dataset in pedestrian flow detection, the third section discusses related work to that.

2.1 Feature-based Detection

In the beginning, the researchers focused on extracting hand-crafted features from the images and train a classifier to detect pedestrian flow. The prominent research work in this genre is [11] which extracts integral channel features with respect to oriented gradient (HOG), color features (LUV) and gradient magnitude. Then it uses boosted decision forest as classifier. Due to the success of this algorithm, later it was extended by many research [10, 5, 40]. Also, different variants of [11] are available. Beside the feature-based methods, There are some sensor-based pedestrian flow detection method ([18, 33]. These types of methods use WiFi and Bluetooth to gather essential information about surrounding pedestrians and try to estimate the pedestrian flow.

2.2 Deep CNN-based Detection

Due to the success of deep learning and computer vision algorithms in various imaging task, recent research in pedestrian flow detection mainly focused on deep CNN-based algorithms. Initially, some researchers modeled pedestrian flow detection similar to object detection task. Since, Faster R-CNN ([28] is a hugely successful algorithm, it was widely used for pedestrian detection. Later, some research work [22, 6] extended fast and faster R-CNN with multi-scale network to deal with scale variations. Scale variation means, the size and positions of pedestrian varies with respect to camera, occlusion and many other factor. A multi-scale network tries to deal with the scaling issue. In addition to detect the whole pedestrian, there are also part-based detectors ([38, 46, 25, 13] available. The reason behind this is, it is hard to detect the whole-body of a pedestrian in a crowded scene. Hence, part-based detectors are very useful. These part-based detectors tried to detect different human parts, more specifically heads, arms in an occluded scene. There are also some crowd counting algorithms [44, 42, 35, 36, 24, 19] which tried to estimate the number of pedestrians in an occluded scene. [42, 35] employed multi-task learning to take advantage of the relation of different task in a pedestrian scene. [7, 26] employed multi-scale or multi-stage network to improve the detection. [36] tried to learn from long-range contextual information while [32] applied deep negative co-relation to tackle over-fitting and estimates the number of people.

2.3 Pedestrian Datasets

Deep Convolutional Neural Network (CNN) requires huge amount of data to train successfully. This section discusses mainly on the available datasets for pedestrian detection. Also, there are some synthetic dataset available which were previously used for crowd counting and can be extended for pedestrian detection as well.

2.3.1 Real-World Datasets

The available pedestrian datasets are INRIA [9], Caltech [12], KITTI ([15], CityPerson [43] and so on. The limitations of these datasets are - they did not focused on traffic-intersections and the camera height is very close to the pedestrians. For example, the INRIA [9] dataset has pedestrians from all different backgrounds. The Caltech [12] dataset focused on mainly pedestrians. One limitation of [12] dataset is that the camera is parallel to the pedestrians which makes it difficult for monitoring pedestrian flow at a traffic intersection. The KITTI [15] (Fig. 2.1) benchmark captured frames from a car and mainly focused on detecting objects for assisting self-driving cars. So, it is not suitable for pedestrian flow detection either. The CityPerson [43] (Fig. 2.2) is similar to [12] which focused on pedestrian detection but the camera height remains parallel to the pedestrians and not suitable for surveillance at a traffic-intersection. Also, the number of pedestrians in the available datasets are not very high and variation in the pedestrian flow is small. There are some datasets [8, 45] for crowd counting which can be used for pedestrian detection as well. This research adopted suitable frames from both [8, 45] to evaluate the proposed algorithm's performance on real-world scenario.

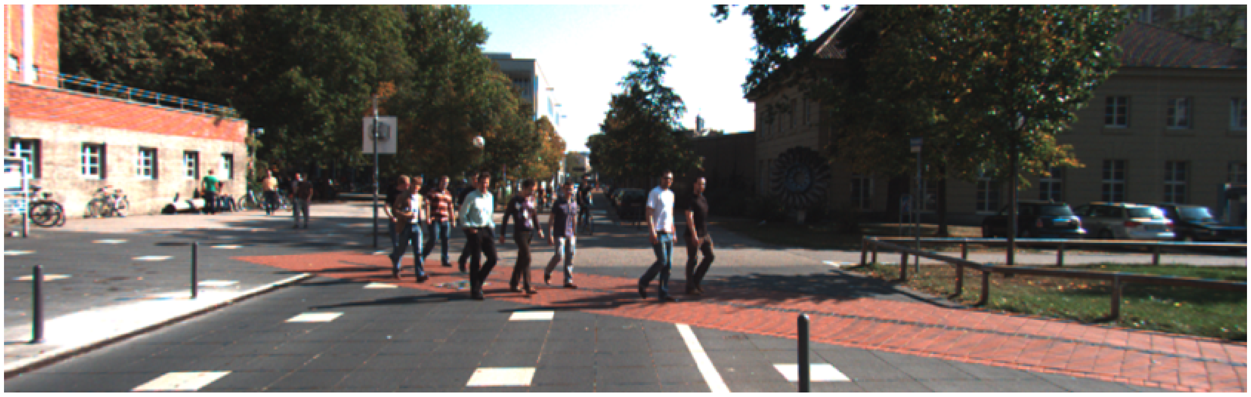


Figure 2.1: An image frame from Kitti [15] dataset

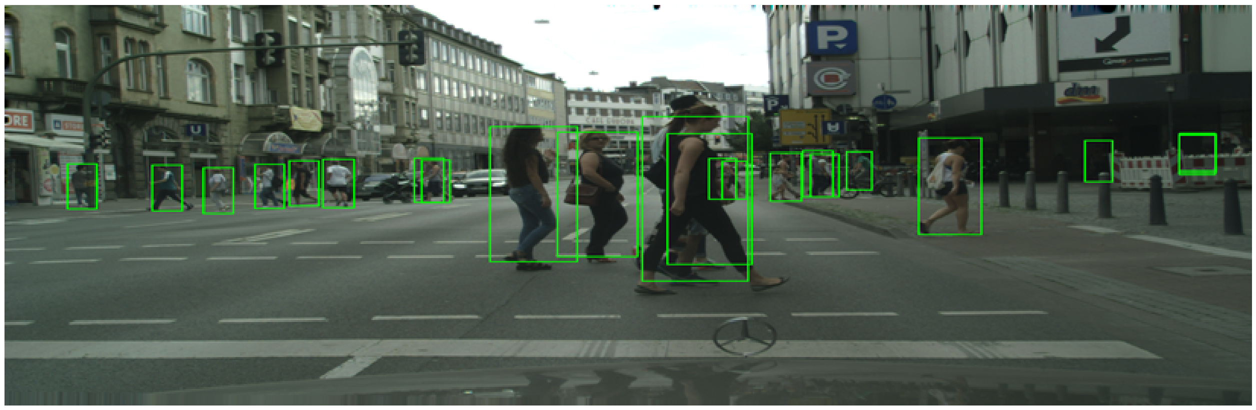


Figure 2.2: An image frame from Cityperson [43] dataset

2.3.2 Synthetic Datasets

Creating a huge human-annotated dataset for pedestrian flow detection requires a lot of time. There is also chance of annotation error which will cause problems while training a deep CNN network. One way to solve this problem can be - creating a simulated pedestrian scenario and extracting frames from there. Using video games can be very useful for this purpose. There are some available datasets [30, 20, 29, 31] available which have successfully extracted image-frames and ground truth from video games. Among them, [30, 20, 29] collected scenes from GTA V. [30] extracted annotation from the rendering pipeline. [20] proposed a method to explore the internal engine buffer and get the depth information which can produce accurate object-masks. [29] used open source game engines and used them to collect six different types of ground truth from the GTA V game. [31] intended to create a dataset for autonomous driving and employed unity game engine to construct synthetic street scenes.

CHAPTER III

GTA-INTERSECTION DATASET

Grand Theft Auto (GTA V) was released by Rockstar games [1] in 2013. It is an action-adventure open world game, where the characters and the environment are very much similar to real-world scenario. Based on a fictional city - Los Santos, the game has used Rockstar Advanced Game Engine(RAGE) to improve it's rendering details. The game has high quality texture details, different weather conditions, pedestrian and vehicle movements which are very similar to real-world conditions. The company also allows players to develop non-commercial mod which also helped to create our intended dataset. There is a C++ library, Script Hook V [2] (Fig. 3.1) which can be used to develop game plug-in. The GCC Crowd Counting Dataset [39] has successfully exploited the features of Script Hook V and created a crowd counting dataset. The difference between the [39] dataset and ours is, we have developed specific video scenes at the traffic-intersections and focused only on pedestrians who are crossing the roads. On the other hand, GCC dataset focused on creating random crowd scenarios at different crowded locations like beach, parking lots and so on. In our dataset, the pedestrians crossing the traffic-intersection is very similar to the real-world and there were cameras at different height to capture each scene simultaneously. To accomplish this task, we took the advantage of another game mod, scene director [3] in addition to Script Hook V.

3.1 Creating Traffic-Intersection Scenes

Scene Director [3] (Fig. 3.2) for GTA V is intended to create a movie by choosing suitable characters, controlling their movements, recording the scene from different camera angles. We have utilized all the available features of Scene Director. Our dataset contains three different pedestrian density levels - small (0-10), medium (10-20) and high (20-40). In the beginning, we created

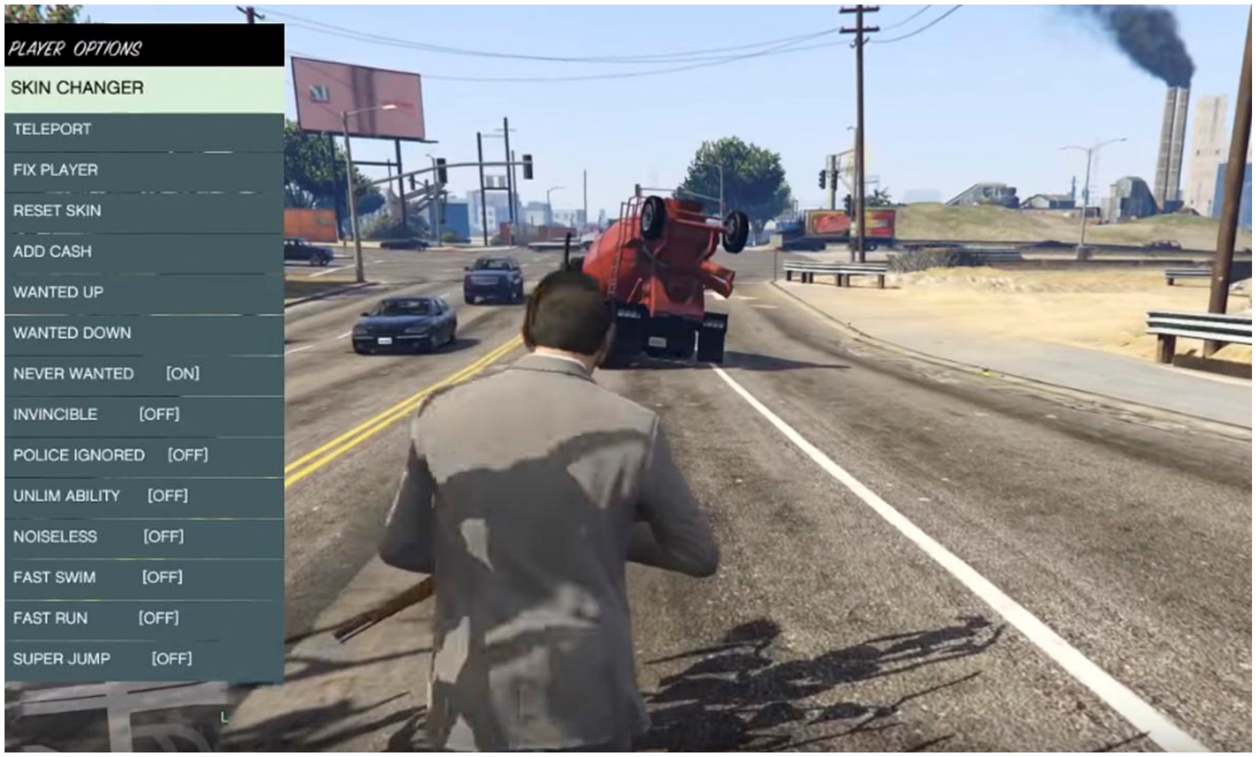
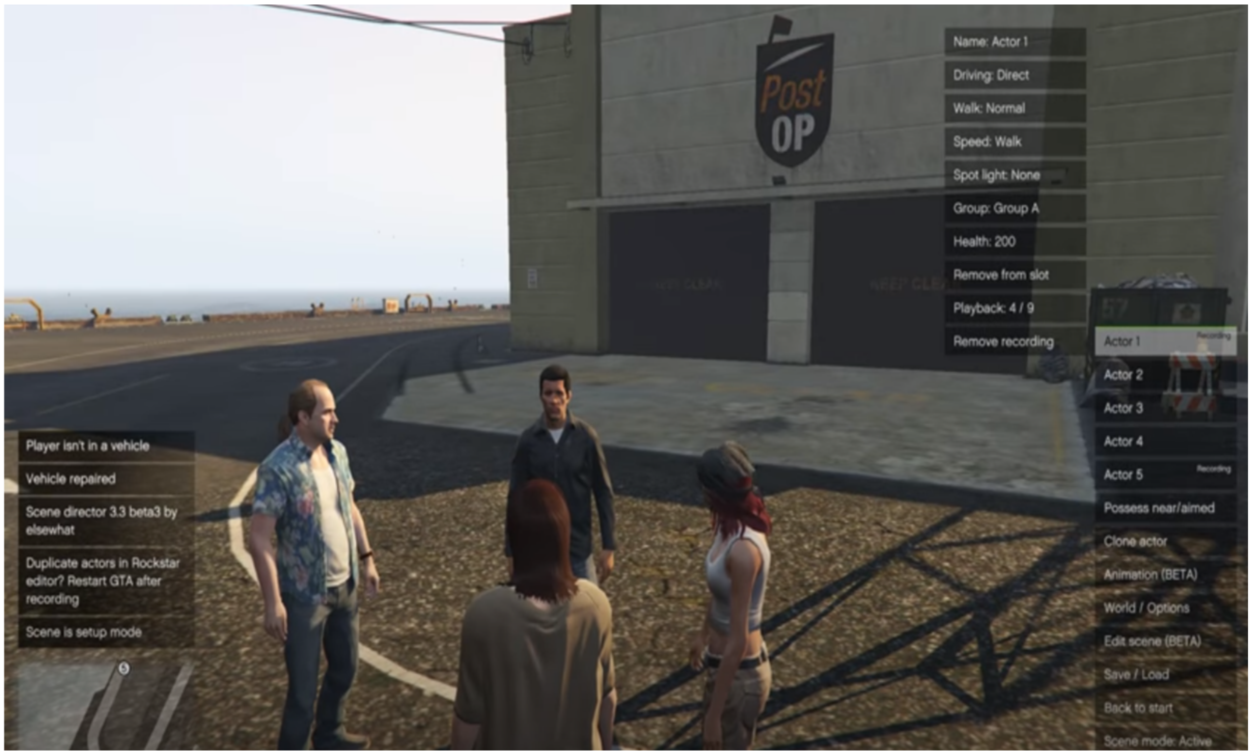


Figure 3.1: Script Hook V [2]

pedestrians with different body shapes, colors, genders and so on and saved them in the Scene Director directory. We then utilized these pedestrians according to the density of each scene. We chose five different traffic-intersections which are very similar to real-world traffic-intersections. We decided to record the pedestrian scene from three different heights, low, medium and high. So, after selecting a traffic intersection, we placed the pedestrians with intended density and recorded their movements. As the pedestrians were different from each other, each scene is very similar to real-world pedestrian scenario. In total, we have generated 75 videos, the frame-rate is 60 frame/second. Then we down-sampled 30 suitable frames from a single second of each video and trained our two-branch Pedestrian Counting Net (PCNet). The resolution of our GTA-Intersection dataset is 1080x720p.

Some samples from our GTA-Intersection dataset are provided in Fig. 3.3.


Name: Actor 1
Driving: Direct
Walk: Normal
Speed: Walk
Spot light: None
Group: Group A
Health: 200
Playback: 4 / 9

Player isn't in a vehicle
Vehicle repaired
Scene director 3.3 beta3 by elsewhat
Duplicate actors in Rockstar editor? Restart GTA after recording
Scene is setup mode

Figure 3.2: Scene Director V [3]

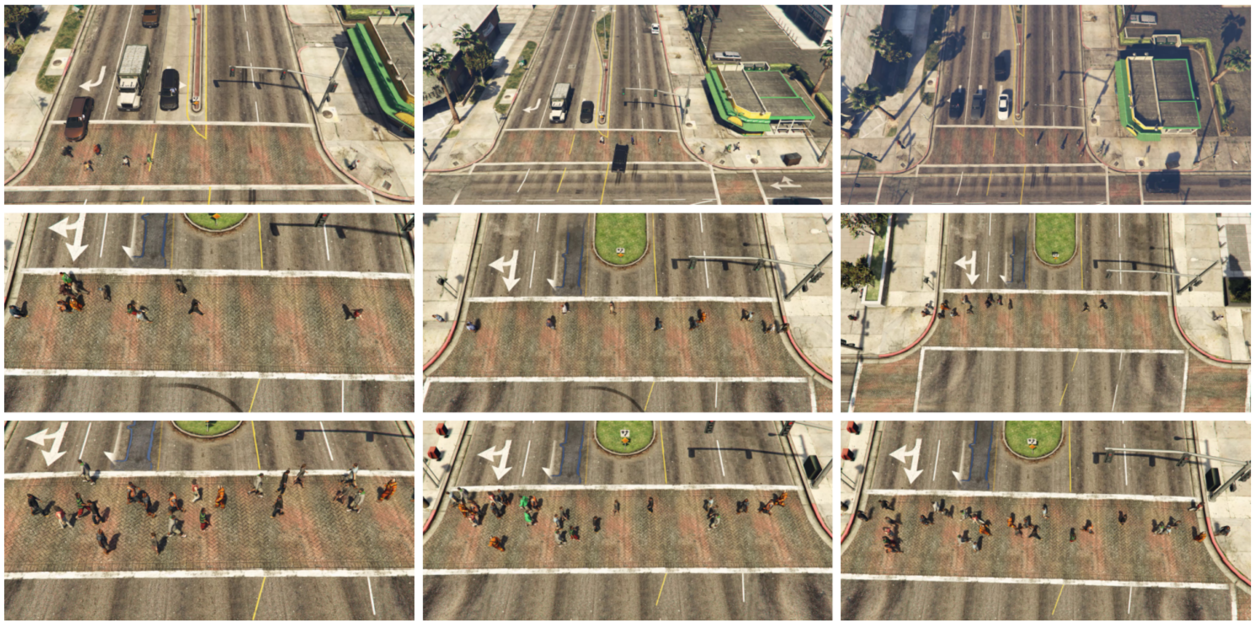


Figure 3.3: GTA-Intersection dataset: 1st row: small pedestrian density (0-10), 2nd row: medium pedestrian density (10-20), c) 3rd row: high pedestrian density (20-40); each column corresponds to different camera height.

3.2 Examples from GTA-Intersection Dataset

Below, there are some examples, Fig 3.4, 3.5, 3.6, 3.7 from the GTA-Intersection dataset. It can be observed that the images frames have high-resolution (1080x720p) and well-defined to train a deep CNN framework. Also, the scenario is very similar to a real-world pedestrian scene.

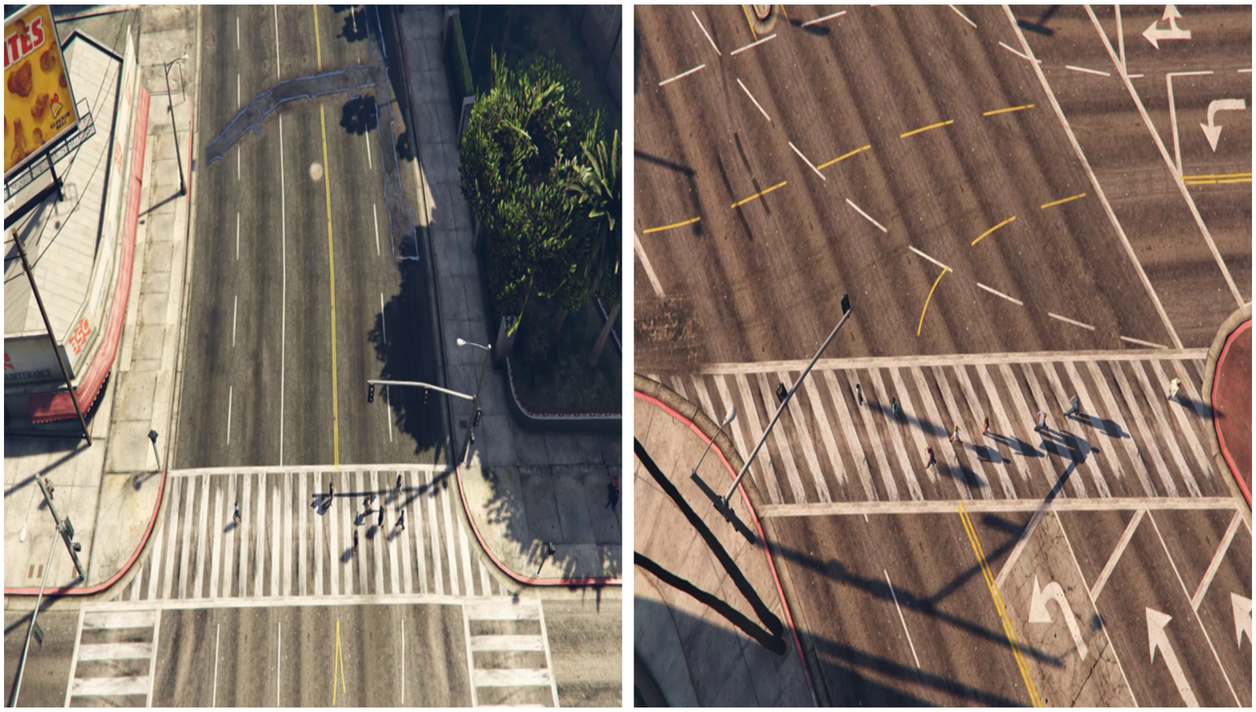


Figure 3.4: Examples from GTA-Intersection dataset.

3.3 Real-Intersection Dataset

To evaluate our proposed framework, we tried to find out some real-world dataset. Unfortunately, we did not find any real-world crossroad-pedestrian scenario similar to our synthetic dataset. We collected some video frames from [45, 8] and labeled them as Subset-1 and Subset-2 respectively. As we did not find many traffic-intersection images similar to our GTA-Intersection dataset, we have collected some similar videos from the street. We have then extracted 80 suitable frames from each of these videos. As some videos contained very high density traffic in the background, we have removed those regions. Fig. 3.8 contains some exapmle from the Real-Intersection dataset.

Here is a summary, Tab. 3.1 contains information about both GTA-Intersection and Real-

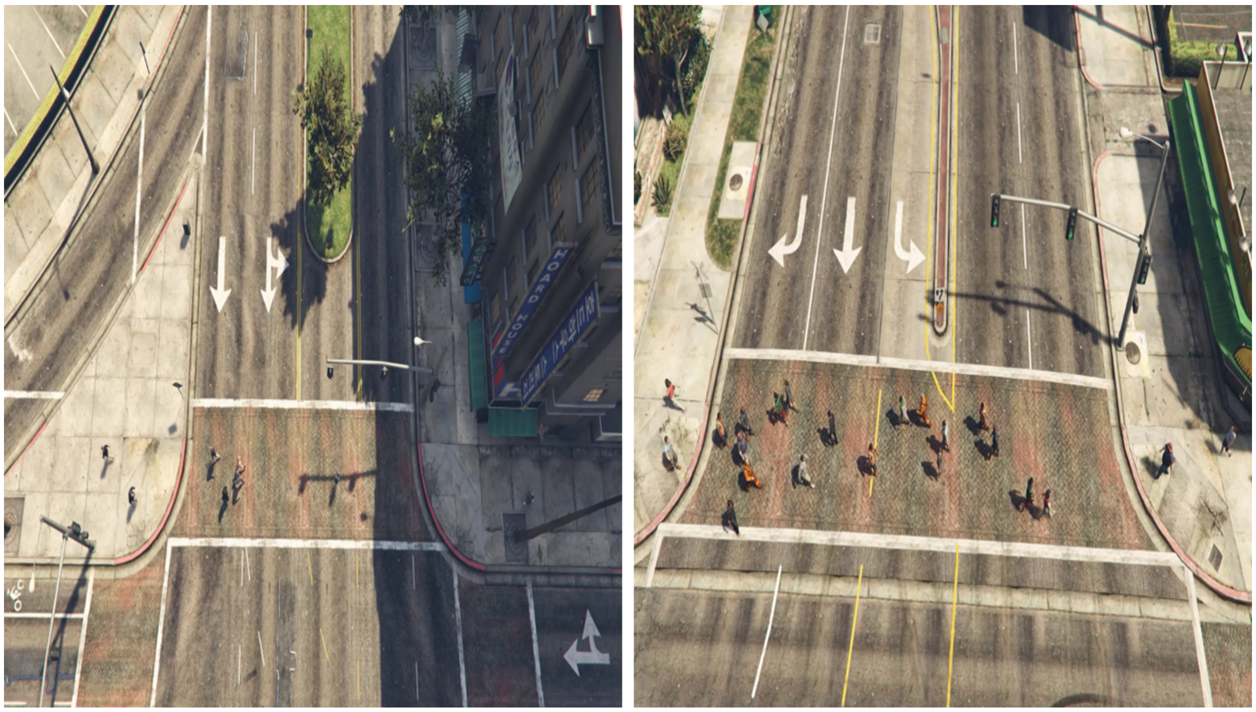


Figure 3.5: Examples from GTA-Intersection dataset.

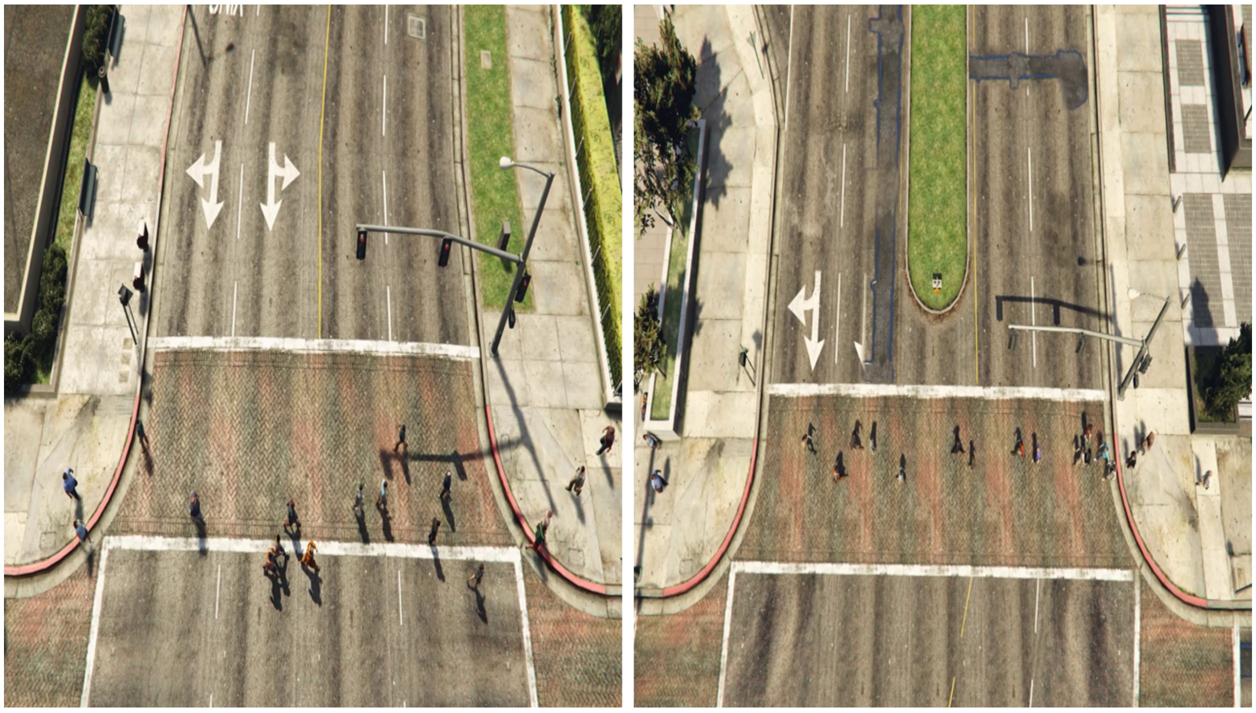


Figure 3.6: Examples from GTA-Intersection dataset.

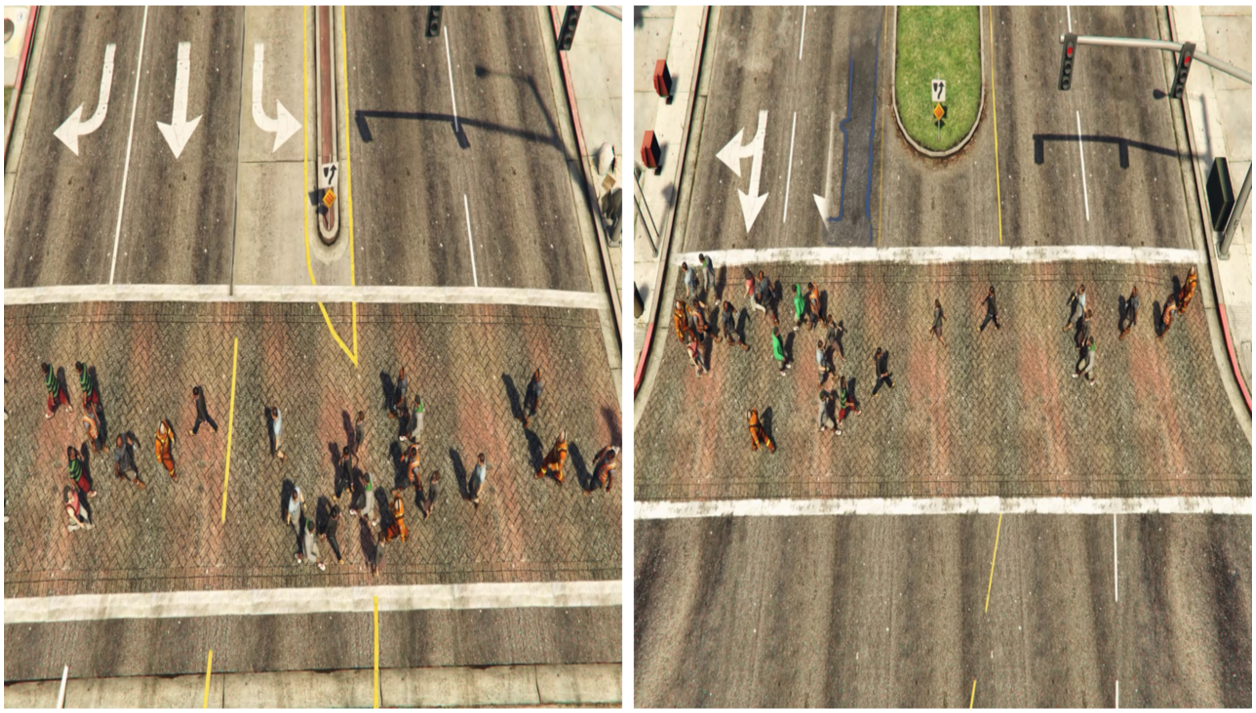


Figure 3.7: Examples from GTA-Intersection dataset.

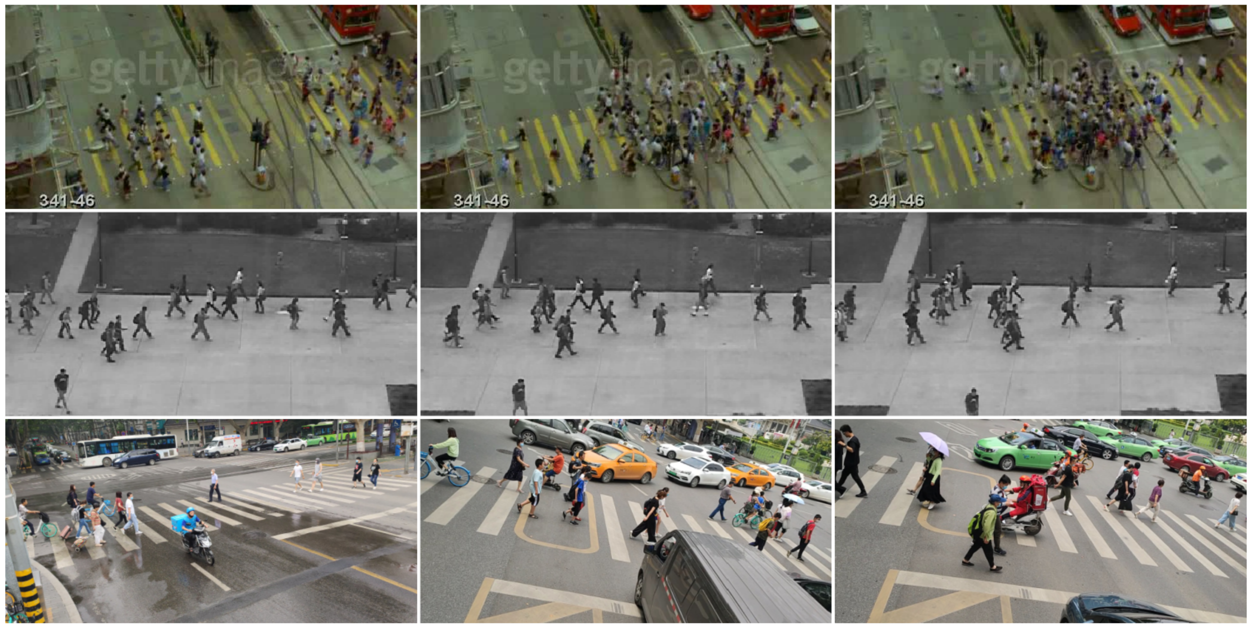


Figure 3.8: Real-Intersection dataset: 1st row: frames collected from [45], 2nd row: frames collected from [8], c) 3rd row: frames collected from the street from us.

Intersection dataset.

Table 3.1: Details of the datasets used in this experiment: image numbers, pedestrain density, and image resolution. There are 75 videos in the GTA-Intersection and 3 videos in Real-Intersection.

Dataset Name	Training	Validation	Testing	Pedestrian Density	Resolution
GTA-Intersection	1,522	218	435	4 to 36	1,080x720
Real-Intersection	130	-	658	6 to 76	720x560

CHAPTER IV

METHODOLOGY

4.1 Generating Optical Flow

As our goal was to detect the pedestrian flow, our initial experiments using only video frames shows that video frames alone was not sufficient enough to achieve that goal. We decided to use both video frame and corresponding optical flow [17] to train our framework. An Optical flow measures the relative motions of objects between two consecutive frames and assigns them into different groups with respect to their directions of motions. As our research focused on detecting the flow of pedestrians in two different directions, the addition of optical flow proved to be very useful. In order to generate a optical flow, we chose two consecutive video frames and generate the corresponding optical flow.

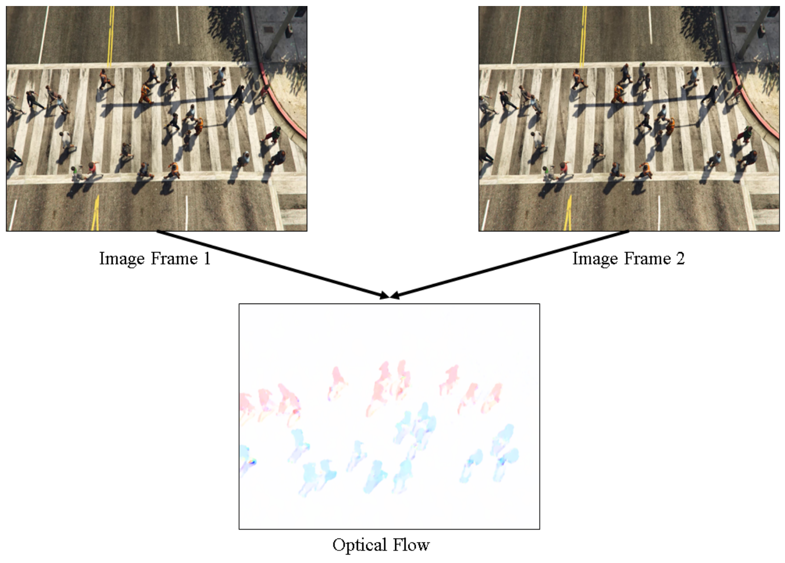


Figure 4.1: Optical flow generation

Optical flow is the motion of objects between consecutive frames of sequence. Let us consider the location an object is (x,y) in the first frame and it's displacement in the second frame can be described as $(x+dx,y+dy)$. Let's assume that $I(x,y,t)$ is the image intensity of the first frame and by moving I to (dx,dy) direction over time dt, we obtain the new image intensity, $I(x+dx,y+dy,t+dt)$. If we assume that the pixel intensities of an object is constant between two consecutive frames, we achive that,

$$I(x,y,t) = I(x+dx,y+dy,t+dt). \tag{4.1}$$

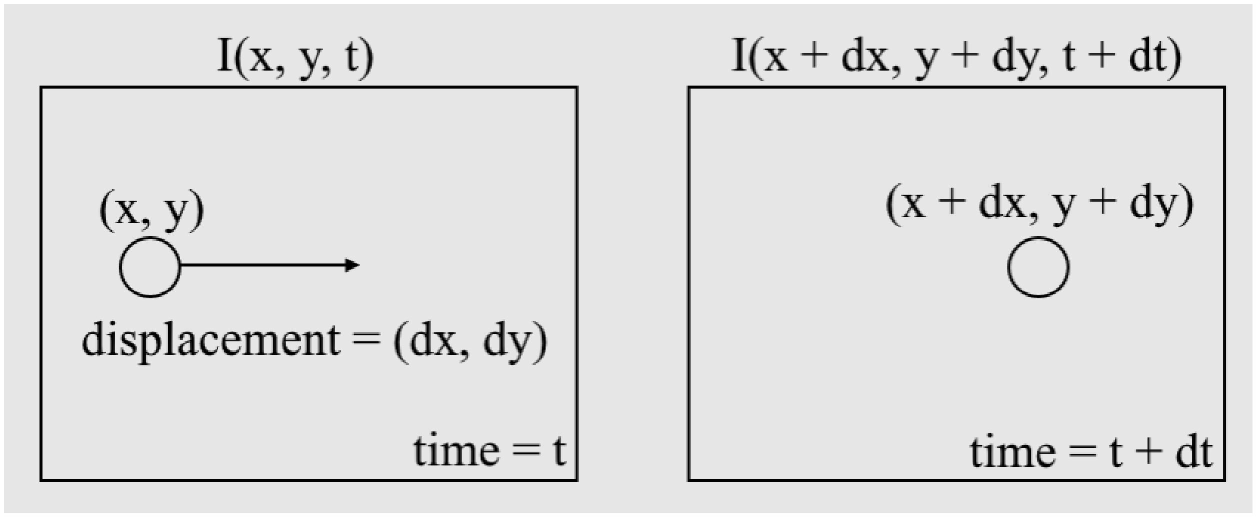


Figure 4.2: Optical flow generation (Collected from Internet)

Optical flow can be obtained by solving the equation,

$$\frac{dI}{dx}u + \frac{dI}{dy}v + \frac{dI}{dt} = 0. \tag{4.2}$$

where u is $\frac{dx}{dt}$ and v is $\frac{dy}{dt}$. From Fig. 4.1, we can observe that the group of people moving from left to right are assigned red color, on the other hand, the group of people moving from right to left are assigned blue color. Optical flow is assigning groups according to their direction of velocity between two consecutive frames.

4.2 Pedestrian Counting Net

As our input consists of both video frame and corresponding optical flow, traditional single branch CNN framework was not suitable for that. We decided to use a pretrained AlexNet [21] as the CNN backbone and modified it so that it can take two inputs. Here, the AlexNet mainly works as a feature extractor, which extracts features (4096) from both video frames and optical flows. Combining them we have got 8192 one-dimensional features. We then added three fully-connected layers and our final layer predicts three outputs - 1) number of pedestrians moving from left to right, 2) number of pedestrians moving from right to left and 3) the total number of pedestrians. As there is inter-dependency in the flow of pedestrians in different directions, we have tried to extract that advantage. Fig. 4.3 describes our proposed two branch Pedestrian Counting Net (PCNet).

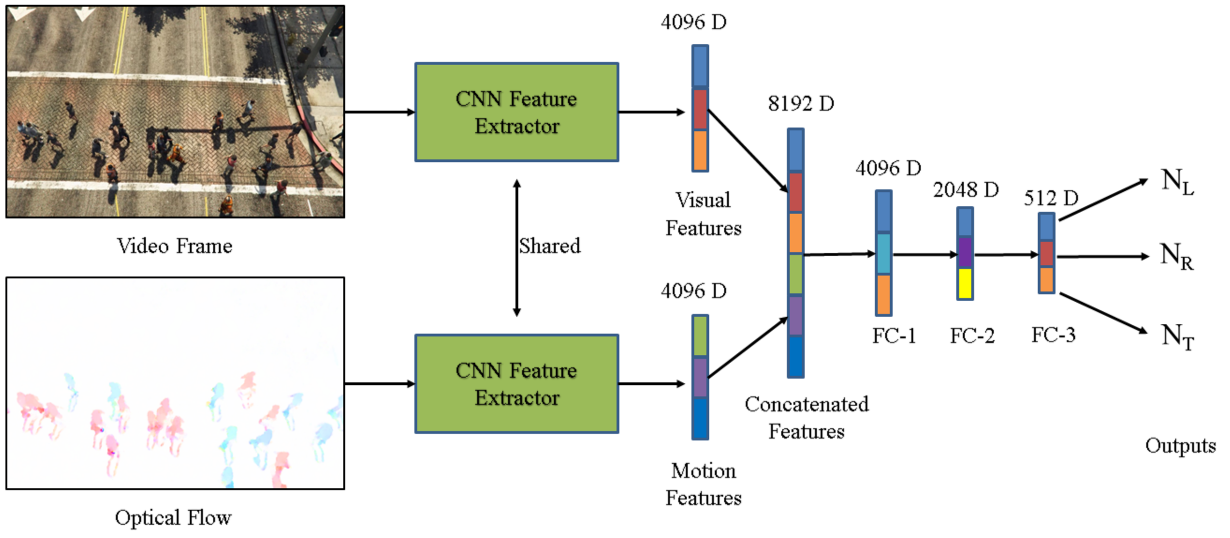


Figure 4.3: The proposed Pedestrian Counting Net (PCNet). The two branches with the inputs of original image and motion(optical flow) image share the weights and the outputs are the N_L, N_R, N_T which stands as pedestrian counting numbers moving from left to right, moving from right to left, and total respectively.

The input is a pair of images - video frames and their corresponding optical flow. Each input dimension is 227x227x3 (a color image with size 227x227). The original AlexNet architecture has 5 convolutional layers, used for creating a feature map that indicates locations and strengths of detected features in the input, 3 max pooling layers for down sampling the data size and 3 fully-connected layers which hold the composite information from all layers before it. We have removed the final layer and then concatenated the features from both video frame and optical flow. We then added 3 more fully-connected layers. The final layer gives output of a 3 dimensional vector - the interpretation is mentioned above.

4.2.1 Review on AlexNet

AlexNet [21] has brought huge improvement in object recognition using deep CNN layers. It was the first successful attempt in large-scale training of deep CNN framework on thousands of object class. The performance of AlexNet is state-of-the-art. AlexNet has total eight layers - five Convolutional layers and three fully-connected layers. first and second CNN layer has a kernel size of 11x11 and 5x5 respectively. The rest of the CNN layers have a 3x3 kernel size. The CNN layers are fused in the end which provides a feature vector of length 4192. The output layer gives a output vector of size 1000. We replace the output layer with three more fully-connected layer. In total, AlexNet has 60 million parameters.

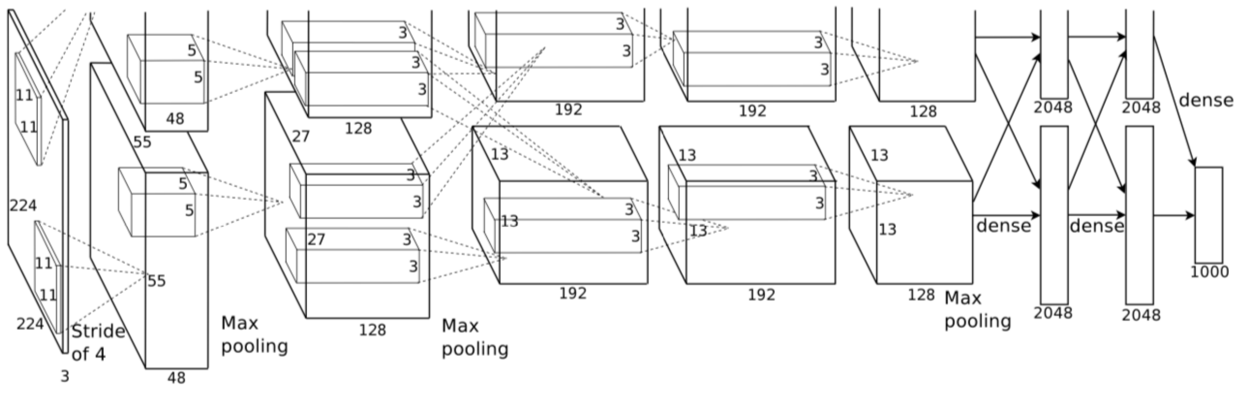
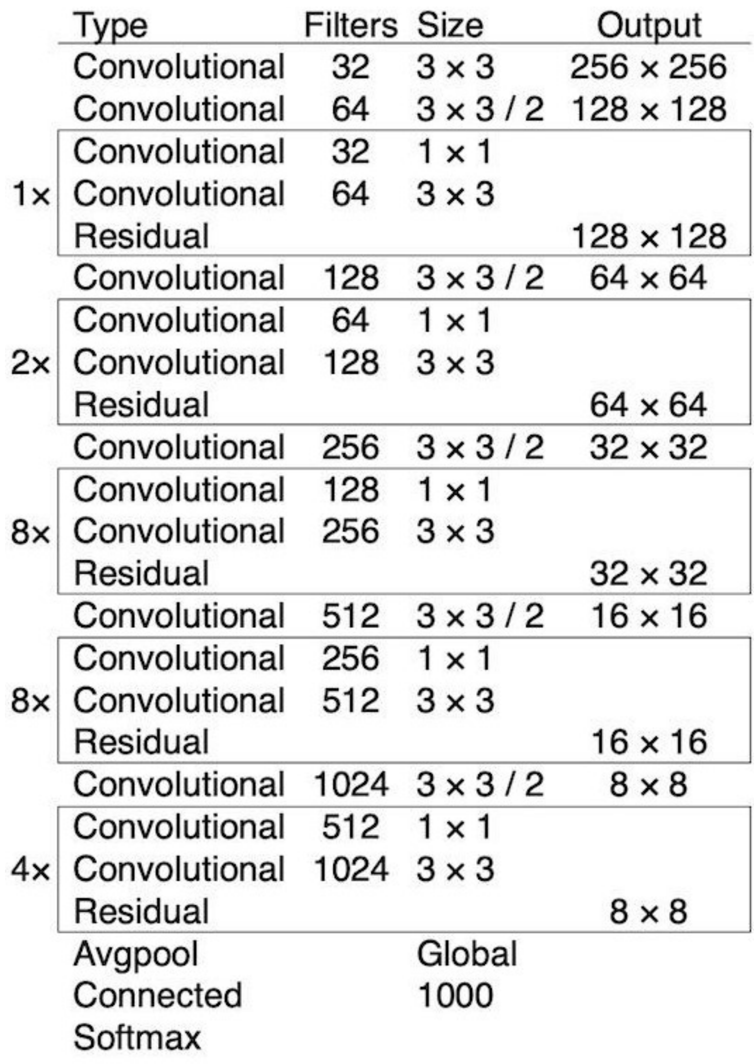


Figure 4.4: AlexNet framework. Image credit [21].

4.2.2 Review on YOLOv3

Since, we have evaluated the performance of Pedestrian Counting Net (PCNet) with YOLOv3, this section provides a review about that. Yolov3 [27] is currently a state-of-the-art framework in object detection. Since, pedestrians can be treated as objects, we utilized YOLOv3 as a human detector.YOLOv3 uses DarkNet-53 [27] as a feature extractor. It has 53 convolutional layers with residual connections. YOLOv3 has 53 more layers on top of the DarkNet, hence it has total 106 layers.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4.5: DarkNet framework. Image credit [27]

The output from the detection kernel is 1x1x(B x (5 + C)) where *B* stands for the number of bounding boxes a cell can predict, 5 is to represent four bounding box attributes and one object confidence and *C* is the number of classes. YOLOv3 makes prediction at three stages, where the output dimensions in each stage is 32, 16 and 8 respectively. To calculate the pedestrian flow in left to right and right to left direction using YOLOv3 framework, we chose to use two consecutive frames, calculate the displacement of bounding boxes in both *x* and *y* dimension and measured the relative motions.

4.3 Loss Function for the training of Pedestrian Counting Net

We used Mean Square Error (MSE) loss function to train the proposed Two-Branch Pedestrian Counting Net (PCNet). As we have preset the number of pedestrians in each video, the ground truth for the numbers of pedestrians moving from left to right, from right to left and combined were readily available to us. Then we calculated the mean square difference between the predicted number of pedestrians by PCNet and ground truth. The equation for MSE is described below:

$$MSE = (\frac{1}{N}) \sum_{i=1}^{n} (\hat{y}_i - y_i)^2. \tag{4.3}$$

Where N is the number of total samples, $\hat{y}$ is the predicted number of pedestrians by PedestrianNet and y represents the ground truth.

4.4 Pedestrian Flow Inference Model

In our proposed method, one Pedestrian Flow Inference Model (PFIM) is designed to estimate the pedestrian flow parameters based on the output of PCNet. Pedestrian flow on a crosswalk or sidewalk can be described by the mathematical model provided by [14]:

$$V = \frac{S}{M}. \tag{4.4}$$

Where V is the volume of flow (p/h), S is the average speed of the pedestrians (km/h) and M is pedestrian area module (km/p), $M = \frac{1}{Density}$. In our experiments, the crosswalk has an length of $\approx 30m$. At first, PCNet detects the number of pedestrians in each frame. The output from PCNet goes into Pedestrian Flow Inference Model (PFIM) that calculates the pedestrian density, $Density = \frac{Number of Pedestrians}{Area}$. According to the [4], the average pedestrian crosswalk walking speed is $1.2\,m/s\,(4.32\,km/hr)$ for most conditions. We utilized this speed as the average pedestrian speed. Our proposed PFIM multiplies the average pedestrian speed with the density to compute the desired pedestrian flow/volume. For example, if 10 people are crossing a crossroad of 20 meter length with the average speed of 4.32 km/h, the pedestrian volume using eq. 4.4 will be, 2160 p/h.

CHAPTER V

DOMAIN ADAPTATION

In many cases, it is very difficult to annotate a real-world dataset because of severe occlusion, low image resolution, bad weather condition or other relevant problems. So, if we can reduce the domain gap between our synthetic dataset and real-world pedestrian scene, we will no longer require a human-annotated real-world dataset and hence there will be no chance of annotation error or other errors. Because, we can preset the number of pedestrians in each scene. More elaborately, we tested our proposed framework on real-world datasets, Subset1 [45], Subset2 [8] and on our collected videos from the street. They were all taken from different camera height and angle and different daytime. So, it is difficult for a single framework to perform well in all these different conditions without considering the domain gap. Recently, [39] showed promising result in reducing domain gap between synthetic and real-world images by using Cycle GAN [48] and Structural Similarity Index Measure (SSIM) [47] as a loss to assess the quality of the reconstructed image produced by Cycle GAN. Hence, we have embraced this technique into our work, performed extensive experiments and achieved state-of-the-art performance.

5.1 Cycle GAN

Cycle GAN [48] is a state-of-the-art algorithm for unpaired image-to-image translation that preserves the contents of the input images. As there is significant domain gap between synthetic pedestrian image and real-world pedestrian image, Cycle GAN can be utilized to reduce the domain gap and to translate the synthetic image into a photo-realistic image. Traditional Cycle GAN consists of two generator, $G_{Syn \to Re}$ and $G_{Re \to Syn}$. In our research work, $G_{Syn \to Re}$ learns the mapping function to translate synthetic images to photo-realistic images and vice-versa for $G_{Re \to Syn}$. Each generator is associated with a Discriminator D_{Syn} and D_{Re} respectively. They try to distinguish between real image and generated image from their corresponding generator.

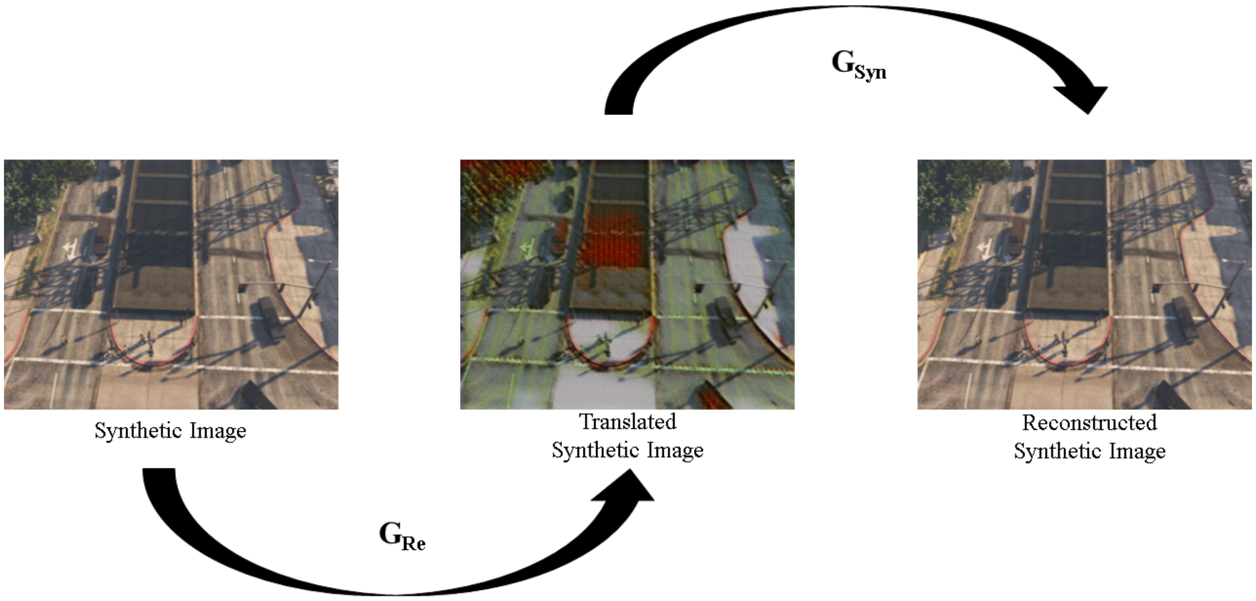


Figure 5.1: CycleGAN [48] framework

The Cycle Consistency Loss [48] enables the generators to learn the mapping function between two different domain. Suppose, *Syn* denotes synthetic domain and *Re* denotes real-world domain. Now, we have a pair of samples $i_s\ \varepsilon\ I_{Syn}$ and $i_r\ \varepsilon\ I_{Re}$ from domain *Syn* and domain *Re*, respectively. So, the objective of cycle consistency loss will be, $i_s \rightarrow G_{Syn\rightarrow Re}(i_s) \rightarrow_{Re} (i_s) \rightarrow G_{Re\rightarrow Syn}(_{Re}(i_s)) \approx i_s$ and vice versa for i_r.

The objective ensures that translation from one domain to another preserves the local and global patterns and contents of the input image. In another word, cycle consistency loss can be described as $L1$ penalty,

$$\begin{aligned} L_{Cycle}(G_{Syn\rightarrow Re}, G_{Re\rightarrow Syn}, Syn, Re) = \\ \mathrm{E}_{i_s\sim I_{Syn}}[||G_{Re\rightarrow Syn}(G_{Syn\rightarrow Re}(i_s)) - i_s||_1] \\ +\mathrm{E}_{i_r\sim I_{Re}}[||G_{Syn\rightarrow Re}(G_{Re\rightarrow Syn}(i_r)) - i_r||_1]. \end{aligned} \tag{5.1}$$

In addition to the cycle consistency loss, we have adversarial loss [16] for two discriminator that detects whether the image is original or the image is generated from the generator. The adversarial loss for discriminator D_{Syn} can be described as,

$$\begin{aligned} L_{GAN}(G_{Re\rightarrow Syn}, D_{Syn}, Syn, Re) = \\ \mathrm{E}_{i_s \sim I_{Syn}}[\log(D_{Syn}(i_s))] \\ +\mathrm{E}_{i_r \sim I_{Re}}[\log(1 - D_{Syn}(G_{Re\rightarrow Syn}(i_r))]. \end{aligned} \tag{5.2}$$

Hence, the loss function for the training of Cycle GAN,

$$\begin{aligned} L_{CycleGAN}(G_{Syn\rightarrow Re}, G_{Re\rightarrow Syn}, D_{Syn}, D_{Re}, Syn, Re) \\ = L_{GAN}(G_{Syn\rightarrow Re}, D_{Syn}, Syn, Re) \\ +L_{GAN}(G_{Re\rightarrow Syn}, D_{Re}, Syn, Re) \\ +\alpha L_{Cycle}(G_{Syn\rightarrow Re}, G_{Re\rightarrow Syn}, Syn, Re). \end{aligned} \tag{5.3}$$

where α is the weight factor for cycle consistency loss.

5.2 Structural Similarity Index Measure Loss

Structural Similarity Index Measure (SSIM) [47] is used widely as an image reconstruction assessment metric. [39] showed that SSIM can be used as a loss function to train Cycle GAN while performing domain adaptation from synthetic to real-world. As there is large difference in texture quality and pattern between synthetic image and real-world image, synthetic image lost many of its properties while performing domain adaptation. Hence, we have decided to include SSIM as a loss function in our work to ensure that our translated images hold its basic contents. Here, the goal is similar to cycle consistency loss, $G_{Syn\rightarrow Re}(G_{Re\rightarrow Syn}(i_r)) \approx i_r$. The range of SSIM is $[-1, 1]$ and when our input image and reconstructed image from the Cycle GAN generators are identical, the SSIM value will be 1. The equation can be described as,

$$\begin{aligned} L_{SSIM}(G_{Syn\rightarrow Re}, G_{Re\rightarrow Syn}, Syn, Re) \\ = \mathrm{E}_{i_s \sim I_{Syn}}[1 - SSIM(i_s, G_{Re\rightarrow Syn}(G_{Syn\rightarrow Re}(i_s)))] \\ +\mathrm{E}_{i_r \sim I_{Re}}[1 - SSIM(i_r, G_{Syn\rightarrow Re}(G_{Re\rightarrow Syn}(i_r)))]. \end{aligned} \tag{5.4}$$

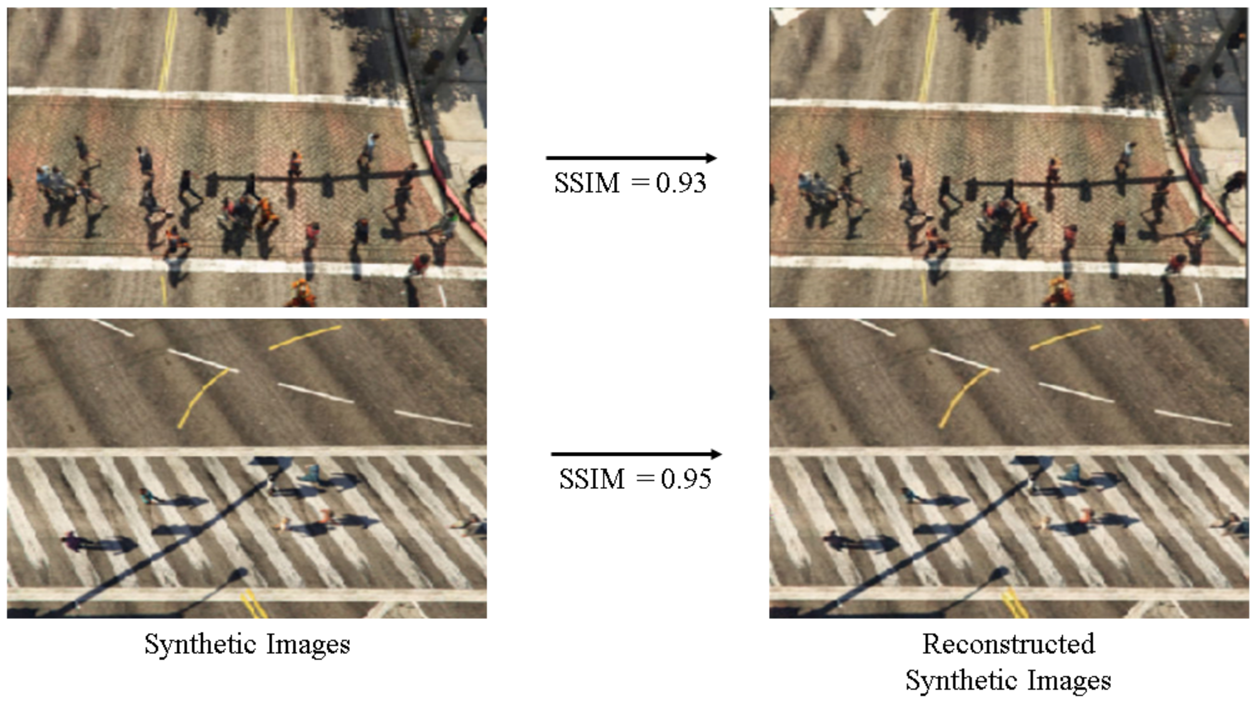


Figure 5.2: Structural Similarity Index Measure (SSIM) between Synthetic Images and Reconstructed Synthetic Images from CycleGAN

5.3 Losses for Domain Adaptation

Combining both CycleGAN loss and SSIM loss the complete loss function for domain adaptation is described below which is similar to [39],

$$\begin{aligned} & L_{Complete}(G_{Syn\to Re}, G_{Re\to Syn}, D_{Syn}, D_{Re}, Syn, Re) \\ & = L_{GAN}(G_{Syn\to Re}, D_{Re}, Syn, Re) \\ & + L_{GAN}(G_{Re\to Syn}, D_{Syn}, Syn, Re) \\ & + \alpha L_{Cycle}(G_{Syn\to Re}, G_{Re\to Syn}, Syn, Re) \\ & + \beta L_{SSIM}(G_{Syn\to Re}, G_{Re\to Syn}, Syn, Re). \end{aligned} \tag{5.5}$$

Here α abd β are associated with Cycle Consistency Loss and SSIM Loss, respectively. We set α equals to 10 and β equals to 2 during training.

5.3.1 Some Examples of Structure-Aware Domain Adaptation

Below, there are some examples of translated synthetic images using CycleGAN in combination with SSIM. It is noticeable that the translated images retains most of it's properties regardless of the style-transfer.

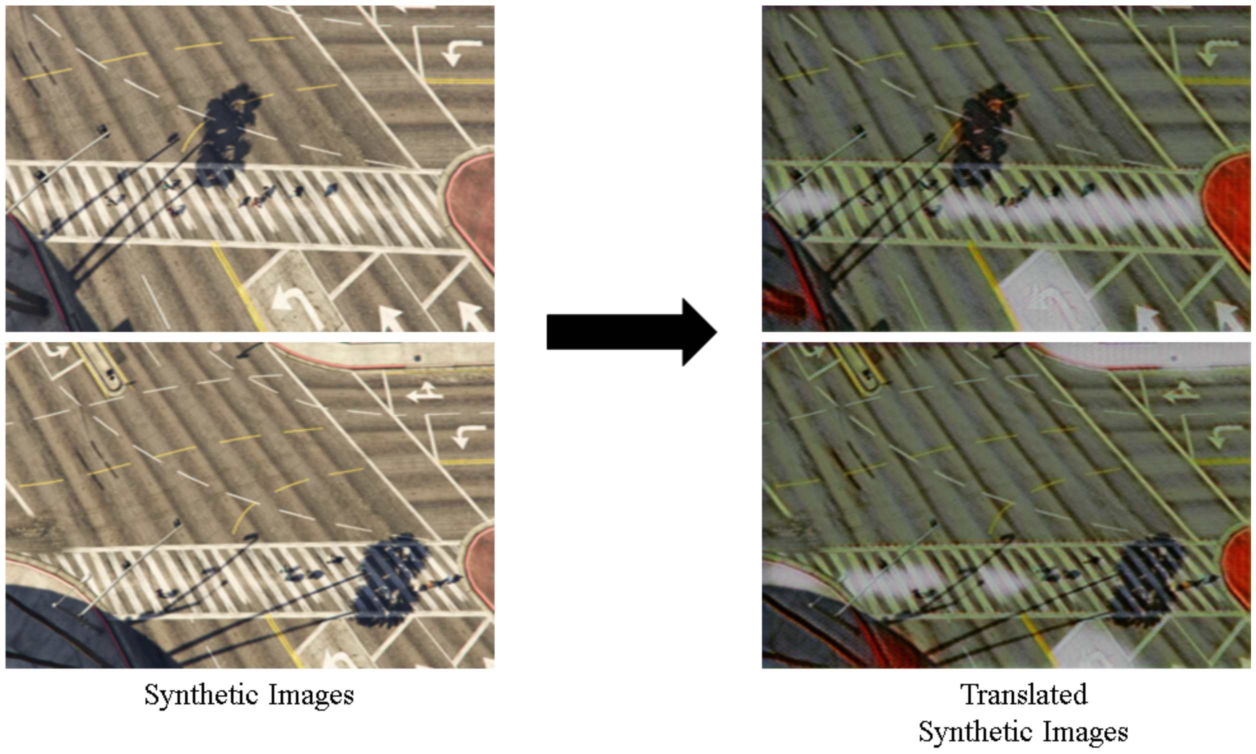


Figure 5.3: Domain adaptation example

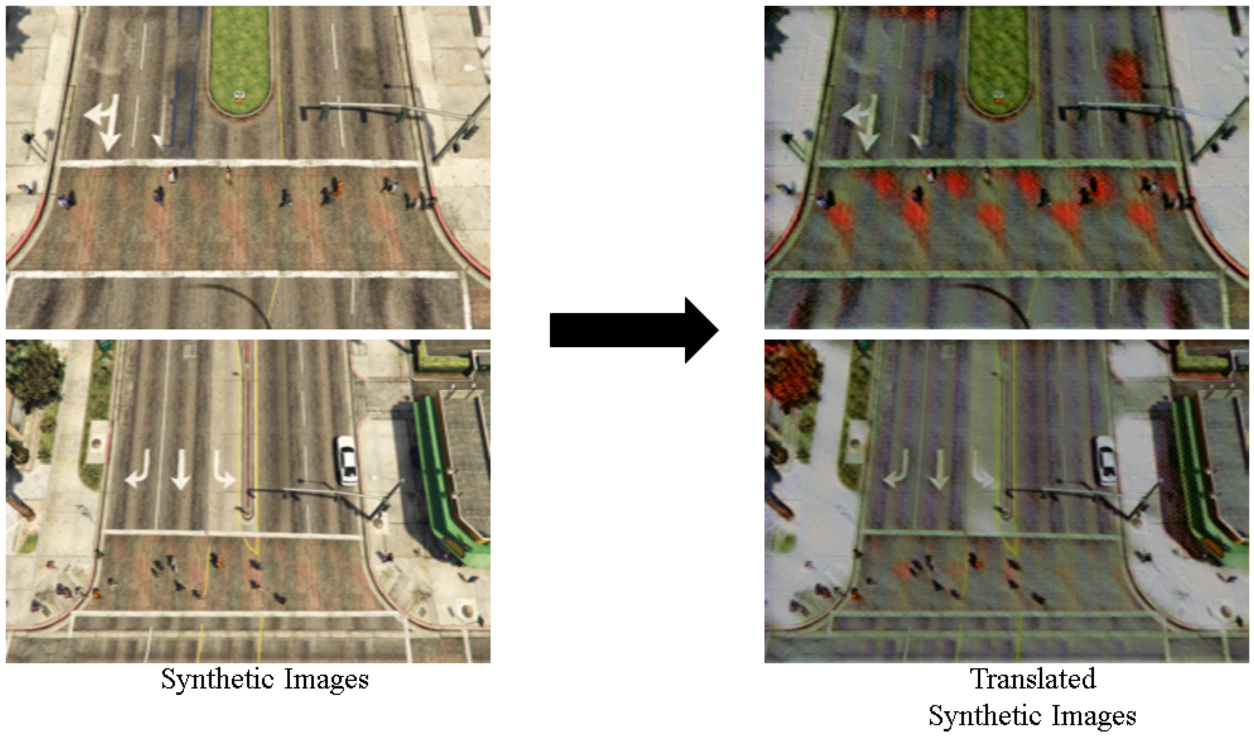


Figure 5.4: Domain adaptation example

CHAPTER VI

EXPERIMENT

6.1 Experimental Setting for Pedestrian Counting Net

We have implemented the two branch Pedestrian Counting Net on Pytorch and our training and testing were performed on a workstation with 3.8 GHz CPU and 12GB memory with a NVIDIA 2080 ti GPU. We used the pre-trained AlexNet to initialize the CNN backbone and randomly initialize other layers of our model, and then we fine-tuned the initialized model using our training set. The setting for the training: batch size of 16, an initial learning rate of 10^{-6}, and a momentum of 0.9. We have applied a learning rate scheduler with a factor of 0.1 and step size 16. We applied mini-batch gradient descent to the optimization during the back propagation and the number of training epoch is 100.

6.2 Experimental Setting for Cycle GAN

We used the generators described in [48] but we have modified the discriminators. We have found a proportional relationship between the degree of domain transfer and the size of discriminator. Using a discriminator too deep can completely transform the domain of the input image and translate it completely to the intended domain which is undesirable. Hence, we only kept two convolution layers in the discriminator, first one with 64 filters of size of 5x5 and second one with 32 filters with size of 3x3. We have used random initialization to initialize both generators and discriminator. The original Cycle GAN has used 200 epoch but we have found 50 epoch is more effective for our project. The initial learning rate was fixed 0.0002 for initial 25 epoch then reduced to zero linearly with each other epoch, similar to the [48]. We have used a replay-buffer similar to the original Cycle GAN to feed the discriminator from outputs of previous epochs.

6.3 Comparison Methods

In the beginning, we evaluated the performance of our proposed PCNet. Our goal of this research project was to explore how much synthetic data can contribute to pedestrian flow detection at a traffic intersection. Hence, in our PCNet+SYN method, we have trained our proposed PCNet only on GTA-Intersection dataset and then tested directly on Real-Intersection dataset mentioned at Table 3.1. In PCNet+SYN+DA method, we have performed domain adaptation using Cycle GAN, fine-tuned the existing PCNet+SYN, on photo-realistic synthetic data. After fine-tuning, we again tested our proposed algorithm directly on Real-Intersection dataset. In our proposed method named as PCNet+SYN+SDA, we performed domain adaption using Cycle GAN and SSIM loss, fine-tuned the existing PCNet+SYN, and tested directly on Real-Intersection dataset because we wanted to demonstrate that the addition of SSIM loss improved the domain adaptation quality. In addition to the above experiments, we chose a small fraction of frames (20%) from Real-Intersection dataset randomly, finetuned the existing PCNets from the PCNet+SYN, PCNet+SYN+DA and PCNet+SYN+SDA methods and perform testing on the rest of the Real-Intersection datasets. The PCNet+REAL$_S$ method was not trained on our synthetic datasets but trained directly on the Small portion of Real-Intersection dataset. The reason is, we wanted to find out whether training on synthetic data improved the pedestrian counting or not. The results are described in Table 6.1, 6.2, 6.3, 6.4, 6.5. We have compared our proposed PCNet with a pre-trained YOLOv3 [27] as a human detector. The YOLOv3 remained pre-trained in all the experiments described below. The results have demonstrated that synthetic datasets can improve pedestrian flow detection significantly without being trained on a huge amount of real-world dataset.

6.4 Experimental Results

Table 6.1 and 6.2 describes our results on our Real-Intersection datasets. Table 6.1 describes the results where the PCNet was only trained on synthetic dataset (PCNet+SYN) or synthetic dataset and then fine-tuned on translated images from Cycle GAN (PCNet+SYN+DA and PC-Net+SYN+DA). Table 6.2 describes results where a small portion of real-world image frames were used for fine-tuning PCNet+SYN, PCNet+SYN+DA and PCNet+SYN+SDA methods. In this case, PCNet+REAL$_S$ method was not trained on our synthetic datasets but on the small portion of Real-Intersection Dataset.

Table 6.1: Performance on the Real-Intersection dataset of the proposed method without using any real-data for fine-tuning.

Method	N_L MAE	N_L MSE	N_R MAE	N_R MSE	N_T MAE	N_T MSE
YOLOv3 [27]	22.48	505.35	18.71	350.06	35.95	1292.67
PCNet + SYN	19.74	389.77	17.83	317.95	37.04	1372.54
PCNet + SYN + DA	18.86	356.47	**17.03**	**290.2**	35.32	1248.0
PCNet + SYN + SDA	**18.41**	**339.06**	17.23	296.9	**35.03**	**1227.27**

Table 6.2: Performance on the Real-Intersection dataset of the proposed method using a small portion of real-data for fine-tuning.

Method	N_L MAE	N_L MSE	N_R MAE	N_R MSE	N_T MAE	N_T MSE
YOLOv3 [27]	22.48	505.35	18.71	350.06	35.95	1292.67
PCNet + REAL$_S$	6.07	36.9	5.22	27.3	5.82	34.66
PCNet + SYN	1.73	3.00	1.70	2.91	3.1	10.21
PCNet + SYN+ DA	1.63	2.67	1.59	2.55	3.01	9.08
PCNet + SYN+ SDA	**1.3**	**1.69**	**1.2**	**1.44**	**2.3**	**5.29**

From both Tables 6.1 and 6.2, we can draw conclusion that domain adaptation using Cycle GAN and SSIM loss (PCNet+SYN+SDA method) has improved the pedestrian counting results. Though using a small portion of real data (Table 6.2) achieved the best MAE and MSE, methods that were only trained and fine-tuned on synthetic data (Table 6.1) also showed promising results. We also did an analysis on each subsets of Real-Intersection dataset. At first, we are going to discuss the performance on the Subset1 [45] of Real-Intersection dataset. We have chosen 298 suitable frames from [45] that were similar to our GTA-Intersection dataset. As adding a small portion of images from Real-Intersection dataset significantly improves the performance, we only focused on the methods described in 6.2. The results on Subset1 are described in Table 6.3.

We can see at Table 6.3 that the performance of our proposed methods is significantly better than the pre-trained YOLOv3 although not very close to the ground truth. The discrepancy is

Table 6.3: Performance of our proposed method on the Subset1 [45] using a small portion of real-data for fine-tuning.

Method	N_L MAE	N_L MSE	N_R MAE	N_R MSE	N_T MAE	N_T MSE
YOLOv3 [27]	23.33	544.45	24.91	620.93	46.94	2204.01
PCNet + REAL$_S$	7.57	57.4	6.18	38.3	6.92	47.9
PCNet + SYN	2.43	5.94	2.53	6.42	4.67	21.90
PCNet + SYN + DA	2.31	5.37	2.36	5.58	4.63	19.68
PCNet +SYN + SDA	**2.07**	**4.30**	**2.70**	**7.29**	**3.50**	**12.26**

explainable. First, the range of pedestrians in our GTA-Intersection dataset was 4 to 36 while in the Subset1 [45], the range of pedestrian was 61 to 76. Hence, training on GTA-Intersection dataset was not sufficient enough. Second, there was severe occlusion in the Subset1 [45] as it was quite dificult even for the human-eye to count the number of pedestrians in each frame. Third, the resolution of the Subset1 [45] was too low compared to our GTA-Intersection dataset. Hence, even domain adaptation was not enough to reduce the domain gap. Still, domain adaptation and addition of SSIM loss (PCNet +SYN + SDA) got improved result than PCNet trained on a small portion of Real-Intersection data only (PCNet + REAL$_S$).

Next, we have tested the performance on Subset2 [8] of Real-Intersection dataset. In Subset2, we chose 120 suitable frames from [8]. The performance of our proposed method on Subset2 is described at Table 6.4.

Table 6.4: Performance of our proposed method on the Subset2 [8] dataset using a small portion of real-data for fine-tuning.

Method	N_L MAE	N_L MSE	N_R MAE	N_R MSE	N_T MAE	N_T MSE
YOLOv3 [27]	9.22	85.17	4.22	17.86	6.08	36.97
PCNet + REAL$_S$	2.70	7.31	1.41	1.99	1.75	3.06
PCNet + SYN	1.09	1.20	0.47	0.22	1.31	1.72
PCNet + SYN + DA	**1.03**	**1.07**	**0.47**	**0.22**	1.22	1.51
PCNet + SYN + SDA	1.26	1.60	0.60	0.36	**1.09**	**1.20**

We can see improved performance on Subset2 [8] than Subset1 [45] but still there is difference between the ground truth and our predictions. The reason is that the frames we have adopted from [8] were in black and white format instead of RGB format. Hence, since our GTA-

Intersection dataset was typical RGB image, when we perform testing on black and white formatted frames, it has lost a significant amount of information as it has only one channel. Hence, the prediction was not accurate. As we did not find anymore dataset that is similar to our GTA-Intersection dataset, we have collected three videos from the streets. We then chose 80 suitable frames from each of these video, in total 240 frames. We removed the background partially as there was heavy flow of cars and vehicles with the flow of pedestrians which could degrade the performance. The results are shown at Table 6.5.

Table 6.5: Performance of our proposed method on Subset3 (we have collected from the street) using a small portion of real-data for fine-tuning.

Method	N_L MAE	N_L MSE	N_R MAE	N_R MSE	N_T MAE	N_T MSE
YOLOv3 [27]	4.78	22.84	3.51	12.55	4.63	21.48
PCNet + REAL$_S$	2.48	6.18	2.93	8.60	2.31	5.31
PCNet + SYN	0.72	0.53	0.51	0.27	1.08	1.17
PCNet + SYN +DA	0.65	0.43	**0.524**	**0.275**	0.94	0.89
PCNet + SYN + SDA	**0.55**	**0.31**	0.67	0.45	**0.77**	**0.60**

Here, in Table 6.5, we can see the performance of our proposed methods trained only on synthetic data is very close to the methods that were fine-tuned on a small portion of Real-Intersection dataset. The dataset we have collected from the streets has similar resolution and background to our GTA-Intersection dataset. Also, the range of pedestrian in both street dataset was GTA-Intersection dataset quite similar. Hence, applying domain adaption was helpful to improve the performance as well. We can see that our PCNet trained on GTA-Intersection dataset and fine-tuned on translated synthetic images (PCNet + SYN + SDA) is capable of detecting pedestrian better than traditional human detector algorithm like YOLOv3. Fig. 6.1 shows our predictions on each subsets from Real-Intersection dataset using $PCNet + SYN + SDA$ method.

After our PCNet Detects the number of pedestrians in each frame, the output goes into our Pedestrian Flow Inference Model (PFIM). At first, PFIM calculates the pedestrian density in each frame using the output from PCNet. After that, it multiplies the pedestrian density and pedestrian speed [4] to get the desired pedestrian flow/volume. Fig. 6.2 shows both pedestrian density and

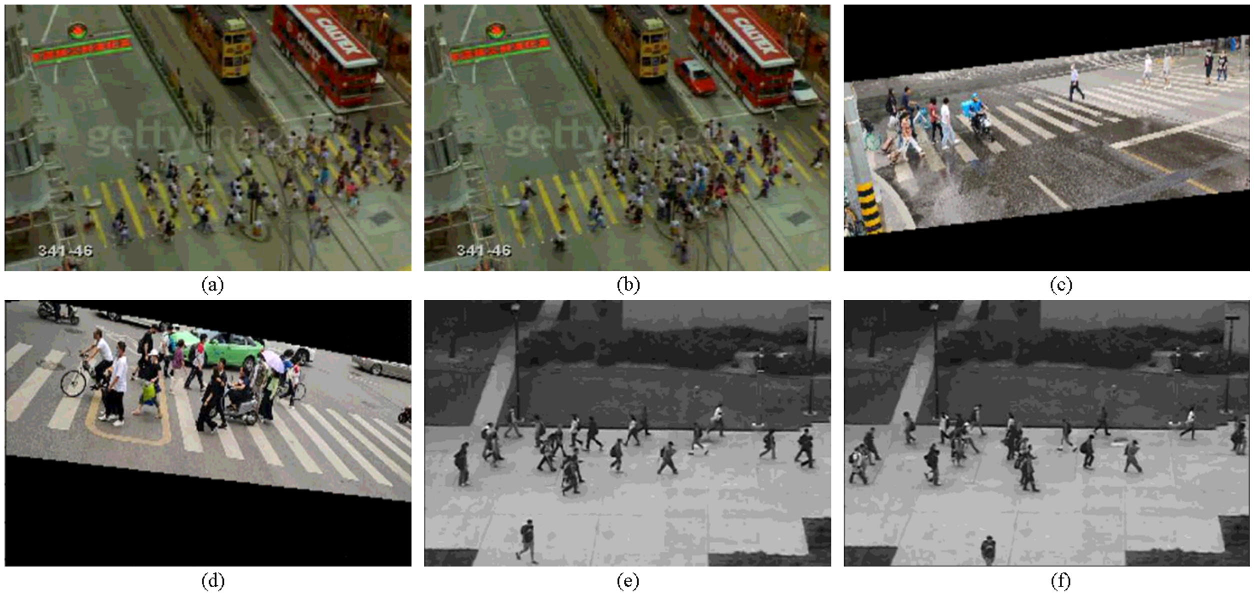


Figure 6.1: Pedestrian counting results by our proposed PCNet framework: (a) Detection (N_L: 33, N_R: 34, N_T: 68), Ground Truth (N_L: 35 , N_R: 37, N_T: 72), (b) Detection (N_L: 34, N_R: 36, N_T: 71), Ground Truth (N_L: 36, N_R: 38, N_T: 74), (c) Detection (N_L: 5, N_R: 4, N_T: 10), Ground Truth (N_L: 6, N_R: 5, N_T: 11), (d) Detection (N_L: 6, N_R: 8, N_T: 15), Ground Truth (N_L: 7, N_R: 9, N_T: 16), (e)Detection (N_L: 19, N_R: 5, N_T: 24), Ground Truth (N_L: 18, N_R: 5, N_T: 23), (f) Detection (N_L: 20, N_R: 5, N_T: 25), Ground Truth (N_L: 20, N_R: 6, N_T: 26).

pedestrian volume in three different directions detected by our proposed Pedestrian Flow Inference Model.

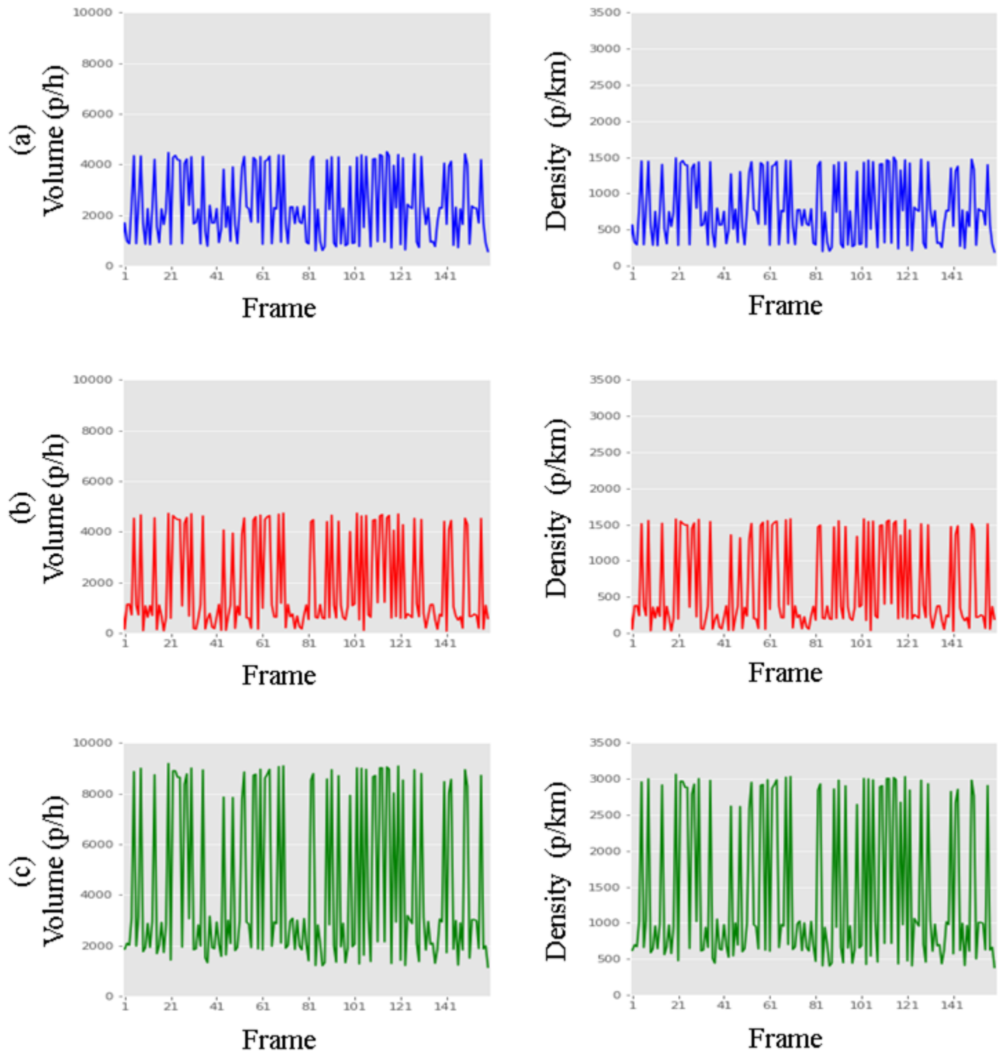


Figure 6.2: Detected pedestrian flow and density on different directions by our proposed PFIM framework : a) flow and density from left to right, b) flow and density from right to left, and c) combined pedestrian flow and density in both directions.

CHAPTER VII

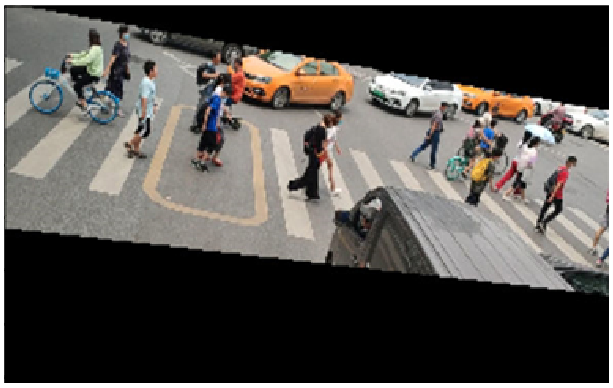
DISCUSSION

It can be observed that the pedestrian counting accuracy is state-of-the-art, very close to the ground-truth which is a indication that the synthetic dataset can be used to train a deep CNN network and it can perform pedestrian flow detection successfully on the real-world data. Also, using domain adaptation makes it possible for using synthetic image in training without the need of human-annotated real-world data which saves valuable time and resources and also reduces the chance of annotation error.

The above examples Fig. 7.1, 7.2, 7.3 is achieved using PCNet+Syn+DA method and using 20% of Real-Intersection dataset for fine-tuning. The detection from PCNet is very accurate. It has to be considered that the above performance is done on three different Subset of real-world dataset. All of them have different density of pedestrian, different camera height. Hence, the utilization of structure-aware domain adaptation has been proved to be very useful. One of the main contribution of this research was the synthetic GTA-Intersection dataset. Here, PCNet was trained on GTA-Intersection dataset first. So, the above results clearly showed that synthetic dataset can successfully replace real-world dataset for training. Other prospects of using synthetic dataset are - it can used to create different weather conditions, it can control pedestrian movement speed. All these are very impactful factor in pedestrian flow detection.

7.1 Summary of Findings

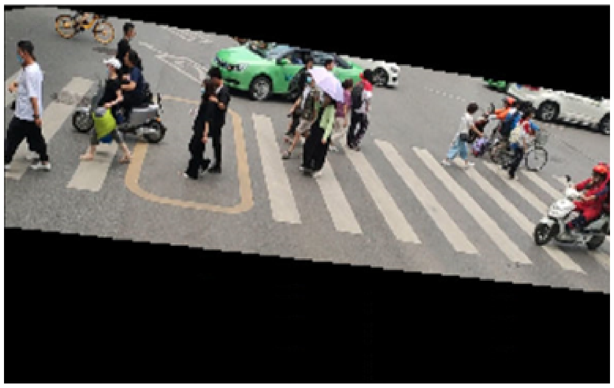
From Tab. 6.1, the best MAE and MSE is obtained using PCNet+SYN+SDA method. The MAE and MSE is higher than expected in all the methods. Hence, we have performed study on each subset of dataset separately to find out the issue. From Tab. 6.2, we can notice that fine-tuning using



N_L N_R N_T

PCNet Output : 14, 1, 15

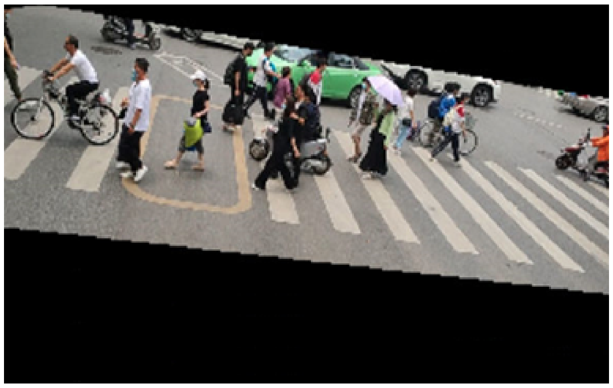
Ground Truth : 14, 2, 16



N_L N_R N_T

PCNet Output : 8, 9, 17

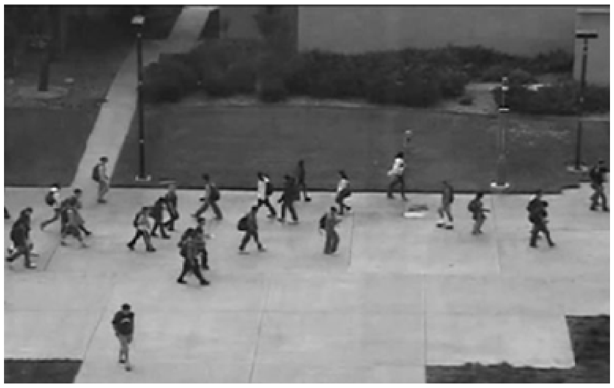
Ground Truth : 8, 10, 18



N_L N_R N_T

PCNet Output : 7, 9, 16

Ground Truth : 8, 10, 18

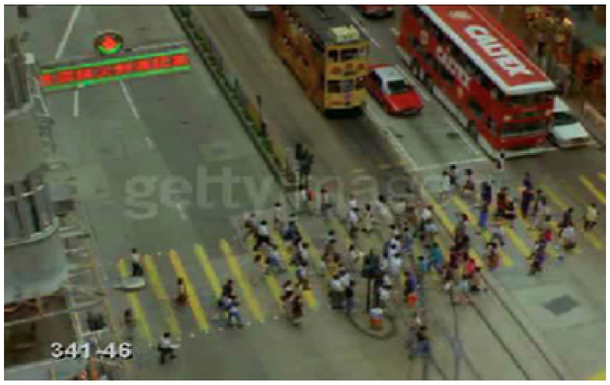


N_L N_R N_T

PCNet Output : 19, 5, 25

Ground Truth : 20, 5, 25

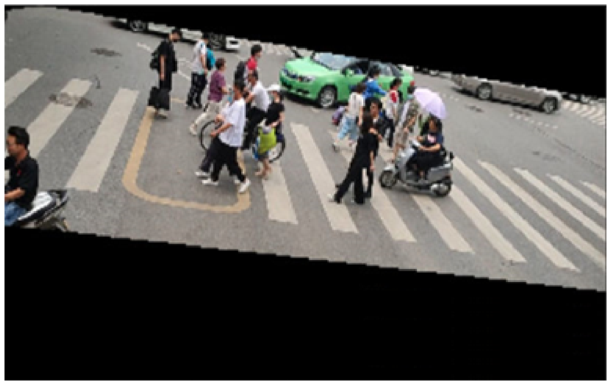
Figure 7.1: Performance of PCNet



N_L N_R N_T

PCNet Output : 36, 37, 73

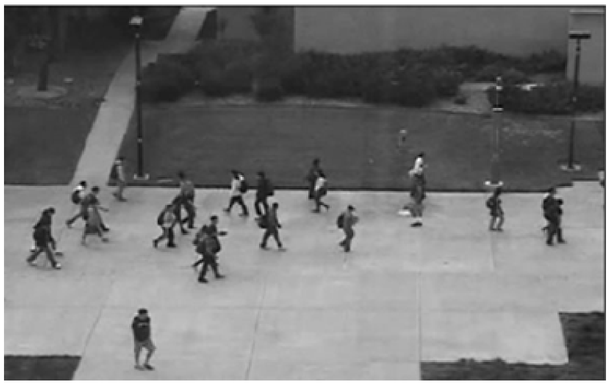
Ground Truth : 36, 37, 73



N_L N_R N_T

PCNet Output : 7, 9, 16

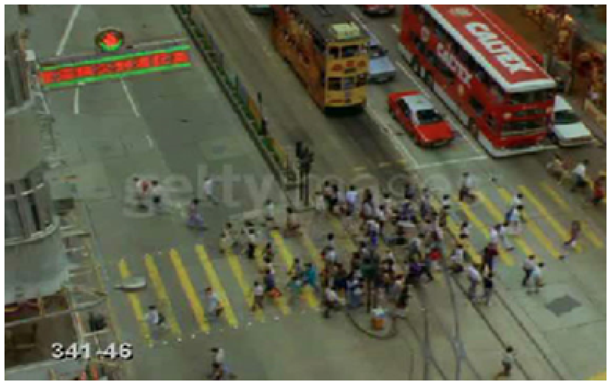
Ground Truth : 7, 10, 17



N_L N_R N_T

PCNet Output : 20, 6, 26

Ground Truth : 18, 5, 23



N_L N_R N_T

PCNet Output : 36, 38, 74

Ground Truth : 36, 39, 75

Figure 7.2: Performance of PCNet

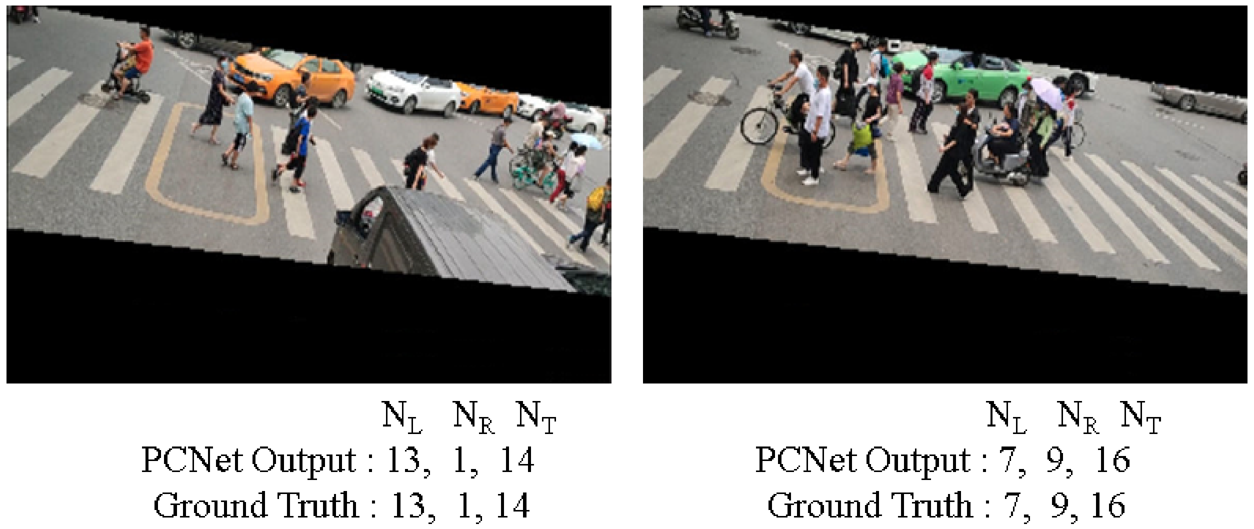


Figure 7.3: Performance of PCNet

a small portion of real-world data significantly improves the performance of each method. This indicates the necessity of successful domain adaptation. Translating an image from synthetic to real domain is a challenging task. There is significant difference in lighting, characters, color intensity between a real image and a synthetic image. From Tab. 6.3, we can observe that the total MAE using PCNet+SYN+SDA method is 3.50 while the MAE using YOLOv3 is 46.94. The subset-1 [45] is heavily occluded, hence human detector like YOLOv3 fails to detect pedestrians, on the other hand our proposed PCNet is capable. From the Tab. 6.4, we observe that PCNet+SYN+DA achieved the best MSE and MAE for detecting pedestrian flow from left to right and right to left direction while PCNet+SYN+SDA achieved the best MSE and MAE for detecting the total pedestrian flow. The performance of YOLOv3 lags behind here as well. The reason of improved performance from subset-1 [45] to subset-2 [8] is that, in subset-2, the pedestrian scenes are not much occluded. Tab. 6.5, the detection of pedestrians using PCNet+SYN+SDA is almost accurate, the combined MAE is 0.77. This is because, the subset-3 image frames has high resolution and pedestrian density similar to the GTA-Intersection dataset. Hence, the performance is state-of-the-art.

From Chapter VI, it is apparent that calculating the pedestrian density with respect to time is the most important parameter. For comparison, we have used YOLOv3 as a human detector. It is needless to say that current object detectors failed to detect in occluded scenes or when the camera height is too high. From the results, it can be concluded that our Pedestrian Flow Inference Model (PFIM) is highly effective in detecting pedestrian flow in real-world scenario.

CHAPTER VIII

CONCLUSION

In this project, we have generated a synthetic pedestrian dataset at traffic-intersection using a video gane. After that, we have proposed a two branch Pedestrian Counting Net (PCNet) for pedestrian counting and a Pedestrian Flow Inference Model (PFIM) for pedestrian flow detection. We embraced domain adaptation technique like Cycle GAN and Structural Similarity Index Measure (SSIM) loss as reconstruction assessment and achieved state-of-the-art result. This is the first approach to create an end-to-end deep CNN framework to detect pedestrian flow based on synthetic pedestrian scenes. One thing we were lacking - a good quality real-world traffic intersection dataset. In future, our work will focus on acquiring more real-world data. Also, we want to explore different domain adaptation technique to reduce the domain gap between synthetic domain and real-world domain.

8.1 Conclusion from the results

In conclusion, we see that the proposed two-branch PCNet achieved the best MSE and MAE in comparison to traditional human detector like YOLOv3. The best MAE and MSE is obtained on subset-3 of Real-Intersection dataset using PCNet+SYN+SDA method where PCNet was first trained on synthetic dataset, then fine-tuned on translated synthetic images obtained from structure-aware domain adaptation. This suggests that domain adaptation is very effective. Also, the addition of Optical Flow as well as two-branch CNN framework proved to be very effective. The best performance was found in a pedestrian density of 0-20 where the performance reduced drastically in subset-1 where pedestrian density is 60-76. This is a limitations of pedestrian detection framework since, they perform poorly on heavily occluded scenes.

8.2 Future Work

Using a human detector like YOLOv3 [27] often fails to detect pedestrians in an occluded scenes. One limitation of our proposed Pedestrian Flow Inference Model (PFIM) is that we failed to evaluate it's performance on diverse real-world conditions due to lack of available datasets. To get rid of this limitations, we will collect more real-world pedestrian scenes in future. Another important aspects is the domain adaptation method. Here, we have used Cycle GAN with Structural Similarity Index Measure (SSIM). In the beginning of domain adaptation, it proved to be very sensitive. A little change in the Discriminator of the Cycle GAN tend to reduce the translated image quality drastically. So, finding a better domain adaptation method is another scope of improvement. Also, the two Pedestrian Counting Net (PCNet) uses AlexNet as a feature extractor. There are many other powerful deep CNN framework available which can be used as well.

BIBLIOGRAPHY

[1] *Grand theft auto - gta v*, Rockstar Games, (2020, accessed January 15, 2020).

[2] Script hook v, (2020, accessed January 25, 2020).

[3] Scene Director GTA V, (2020, accessed January 26, 2020).

[4] D. P. ALLEN, N. ROUPHAIL, J. E. HUMMER, AND I. JOSEPH S. MILAZZO, *Operational analysis of uninterrupted bicycle facilities*, Transportation Research Record, (1998).

[5] R. BENENSON, M. OMRAN, J. HOSANG, AND B. SCHIELE, *Ten years of pedestrian detection, what have we learned?*, in European Conference on Computer Vision Workshops, 2014.

[6] Z. CAI, Q. FAN, R. S. FERIS, AND N. VASCONCELOS, *A unified multi-scale deep convolutional neural network for fast object detection*, in European Conference on Computer Vision, 2016.

[7] X. CAO, Z. WANG, Y. ZHAO, AND F. SU, *Scale aggregation network for accurate and efficient crowd counting*, in European Conference on Computer Vision, 2018.

[8] A. B. CHAN, ZHANG-SHENG JOHN LIANG, AND N. VASCONCELOS, *Privacy preserving crowd monitoring: Counting people without people models or tracking*, in IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[9] N. DALAL AND B. TRIGGS, *Histograms of oriented gradients for human detection*, in IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[10] P. DOLLAR, S. BELONGIE, AND P. PERONA, *The fastest pedestrian detector in the west*, in British Machine Vision Conference, 2010.

[11] P. DOLLAR, Z. TU, P. PERONA, AND S. BELONGIE, *Integral channel features*, in British Machine Vision Conference, 2009.

[12] P. DOLLAR, C. WOJEK, B. SCHIELE, AND P. PERONA, *Pedestrian detection: A benchmark*, in IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[13] P. F. FELZENSZWALB, R. B. GIRSHICK, D. MCALLESTER, AND D. RAMANAN, *Object detection with discriminatively trained part-based models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2010).

[14] J. FRUIN, *Designing for pedestrians: A level-of-service concept*, Highway Research Record, (1971).

[15] A. GEIGER, P. LENZ, AND R. URTASUN, *Are we ready for autonomous driving? the kitti vision benchmark suite*, in IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[16] I. J. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE, AND Y. BENGIO, *Generative adversarial networks*, in Advances in Neural Information Processing Systems, 2014.

[17] B. K. HORN AND B. G. SCHUNCK, *Determining Optical Flow*, in Techniques and Applications of Image Understanding, J. J. Pearson, ed., International Society for Optics and Photonics, SPIE, 1981.

[18] B. HUANG, G. MAO, Y. QIN, AND Y. WEI, *Pedestrian flow estimation through passive wifi sensing*, IEEE Transactions on Mobile Computing, (2019).

[19] H. IDREES, M. TAYYAB, K. ATHREY, D. ZHANG, S. AL-MAADEED, N. RAJPOOT, AND M. SHAH, *Composition loss for counting, density map estimation and localization in dense crowds*, in European Conference on Computer Vision, 2018.

[20] 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?*, in IEEE International Conference on Robotics and Automation, 2017.

[21] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, Communications of the ACM, (2017).

[22] J. LI, X. LIANG, S. SHEN, T. XU, J. FENG, AND S. YAN, *Scale-aware fast r-cnn for pedestrian detection*, 2016.

[23] J. LI, Z. XU, L. FU, X. ZHOU, AND H. YU, *Domain adaptation from daytime to nighttime: A situation-sensitive vehicle detection and traffic flow parameter estimation framework*, Transportation Research Part C: Emerging Technologies, (2021).

[24] Y. LI, X. ZHANG, AND D. CHEN, *Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes*, in IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[25] W. OUYANG, H. ZHOU, H. LI, Q. LI, J. YAN, AND X. WANG, *Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2018).

[26] V. RANJAN, H. LE, AND M. HOAI, *Iterative crowd counting*, in European Conference on Computer Vision, 2018.

[27] J. REDMON AND A. FARHADI, *Yolov3: An incremental improvement*, arXiv preprint arXiv:1804.02767, (2018).

[28] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2015).

[29] S. R. Richter, Z. Hayder, and V. Koltun, *Playing for benchmarks*, in IEEE International Conference on Computer Vision, 2017.

[30] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, *Playing for data: Ground truth from computer games*, in European Conference on Computer Vision, 2016.

[31] 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 IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[32] Z. Shi, L. Zhang, Y. Liu, X. Cao, Y. Ye, M.-M. Cheng, and G. Zheng, *Crowd counting with deep negative correlation learning*, in IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[33] N. Shlayan, A. Kurkcu, and K. Ozbay, *Exploring pedestrian bluetooth and wifi detection at public transportation terminals*, in IEEE 19th International Conference on Intelligent Transportation Systems, 2016.

[34] J. C. Silveira Jacques Junior, S. R. Musse, and C. R. Jung, *Crowd analysis using computer vision techniques*, IEEE Signal Processing Magazine, (2010).

[35] V. A. Sindagi and V. M. Patel, *Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting*, in IEEE International Conference on Advanced Video and Signal Based Surveillance, 2017.

[36] V. A. Sindagi and V. M. Patel, *Generating high-quality crowd density maps using contextual pyramid cnns*, in IEEE International Conference on Computer Vision, 2017.

[37] S. Song, Z. Miao, H. Yu, J. Fang, K. Zheng, C. Ma, and S. Wang, *Deep domain adaptation based multi-spectral salient object detection*, IEEE Transactions on Multimedia, (2020).

[38] Y. Tian, P. Luo, X. Wang, and X. Tang, *Deep learning strong parts for pedestrian detection*, in IEEE International Conference on Computer Vision, 2015.

[39] Q. Wang, J. Gao, W. Lin, and Y. Yuan, *Learning from synthetic data for crowd counting in the wild*, in IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[40] M. You, Y. Zhang, C. Shen, and X. Zhang, *An extended filtered channel framework for pedestrian detection*, IEEE Transactions on Intelligent Transportation Systems, (2018).

[41] C. Zhang, K. Kang, H. Li, X. Wang, R. Xie, and X. Yang, *Data-driven crowd understanding: A baseline for a large-scale crowd dataset*, IEEE Transactions on Multimedia, (2016).

[42] C. Zhang, H. Li, X. Wang, and X. Yang, *Cross-scene crowd counting via deep convolutional neural networks*, in IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[43] S. Zhang, R. Benenson, and B. Schiele, *Citypersons: A diverse dataset for pedestrian detection*, in IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[44] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, *Single-image crowd counting via multi-column convolutional neural network*, in IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[45] B. Zhou, X. Tang, and X. Wang, *Coherent filtering: Detecting coherent motions from crowd clutters*, in European Conference on Computer Vision, 2012.

[46] C. Zhou and J. Yuan, *Multi-label learning of part detectors for heavily occluded pedestrian detection*, in IEEE International Conference on Computer Vision, 2017.

[47] Zhou 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, (2004).

[48] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, *Unpaired image-to-image translation using cycle-consistent adversarial networks*, in IEEE International Conference on Computer Vision, 2017.

BIOGRAPHICAL SKETCH

Abhijit Baul graduated from Bangladesh University of Engineering and Technology in 2018 with a bachelor's degree in Electrical and Electronic Engineering. Abhijit entered the master's program in Electrical Engineering at the University of Texas Rio Grande Valley at 2019. Upon entering the graduate program, he began working with Dr. Hongkai Yu in the Computer Vision Intelligent Transportation Systems lab where he has helped with multiple ongoing research problems dealing with transportation as well as conducting his own research. He received the prestigious Presidential Graduate Research Assistantship (PGRA) from utrgv. Abhijit received his master's degree in Electrical Engineering from the University of Texas Rio Grande Valley in the Spring of 2021 (May 2021) and can be reached through email at abhijitbaul80@gmail.com.