# Real-Time Collision Forecasting from Monocular Video

Aashi Manglik

CMU-RI-TR-19-42

July 2, 2019

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

**Thesis Committee:**
Prof. Kris M. Kitani, Chair, CMU
Prof. Aaron Steinfeld, CMU
Ishani Chatterjee, CMU

*Thesis proposal submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics*

# Abstract

We explore the possibility of using a single monocular camera to forecast the time to collision between a suitcase-shaped robot being pushed by its user and other nearby pedestrians. We develop a purely image-based deep learning approach that directly estimates the time to collision without the need of relying on explicit geometric depth estimates or velocity information to predict future collisions. While previous work has focused on detecting immediate collision in the context of navigating Unmanned Aerial Vehicles, the detection was limited to a binary variable (*i.e.*, collision or no collision). We propose a more fine-grained approach to collision forecasting by predicting the exact time to collision in terms of milliseconds, which is more helpful for collision avoidance in the context of dynamic path planning. To evaluate our method, we have collected a novel large-scale dataset of over 13,000 indoor video segments each showing a trajectory of at least one person ending in a close proximity (a near collision) with the camera mounted on a mobile suitcase-shaped platform. Using this dataset, we do extensive experimentation on different temporal windows as input using an exhaustive list of state-of-the-art convolutional neural networks (CNNs). Our results show that our proposed multi-stream CNN is the best model for predicting time to near-collision. The average prediction error of our time to near collision is $0.75$ seconds across the test videos.

# Acknowledgement

I would first like to thank my advisor, Professor Kris Kitani, for giving me the opportunity to work on the CaBot project. Through all the triumphs and failures accompanying this Master program, I am glad to have his experience and guidance by my side. I would further like to thank my thesis committee member, Professor Aaron Steinfeld, for his support, encouragement and valuable feedback along the course of this work and journey ahead. Thanks to Eshed Ohn-Bar for getting me started on this two-year long ride and the insightful discussions which helped me overcome any doubts.

I would like to thank Xinshuo Weng for the support during a hard deadline and help improve the quality of my work. I would like to thank Ishani Chatterjee whose keen intellect has helped me develop deeper understanding of the field. Thanks to all the members of KLab and Cognitive Assistance Lab for providing a thriving workplace and help me grow. I cannot go without thanking my friends for providing both academic and emotional support in difficult times.

Finally, I would like to express a deep gratitude to my parents and my brother for their endless support and love.

# Contents

# Chapter 1

# Introduction

Automated collision avoidance technology is an indispensable part of mobile robots. As an alternative to traditional approaches using multi-modal sensors, purely image based collision avoidance strategies [9], [17] have recently gained attention in robotics. These image-based approaches use the power of large data to detect immediate collision as a binary variable - collision or no collision. In this work, we propose a more fine-grained approach to predict the exact time to collision from images, with a much longer prediction horizon.

A method frequently used for forecasting time to collision is to track the 3D location of the surrounding pedestrians and extrapolate their trajectories using a constant velocity model [21], [14]. When it is possible to use high quality depth imaging devices, this type of physics-based modeling can be very accurate. However, physics-based approach can also be prone to failure in the presence of sensor noise and uncertainty in detection of nearby pedestrians. Small mistakes in the estimation of depth (common to low-cost depth sensors) or noise in 2D bounding box detection (common to image-based object detection algorithms) can be misinterpreted to be very large changes of velocity. Many physics-based approaches can be brittle in the presence of such noise. Accordingly, errors in either pedestrian detection, tracking or data association can result in very bad future trajectory estimates. Other more advanced physics-based models and decision-theoretic models also depend heavily on accurate state estimates of nearby people and can be significantly influenced by sensor and perception algorithm noise. We propose to address the issue of sensor noise and perception algorithm noise by directly estimating the time to near-collision from a sequence of images.

To create a dataset for learning time to collision, we designed a training prototype as shown in Fig. 1.1. It is both unnatural and infeasible to record or insist that people actually collide with the mobile platform to collect large scale data. As an abstraction, we define the presence of a person within a 1 meter radius around the mobile platform as a near-collision. If a person other than the user is present within this radius, we mark it as a near-collision that should be forecasted using an earlier segment of video. The proposed approach is designed for a mobile robot that is being pushed by a person with visual impairment as shown in Fig. 1.1. The goal of the system is to forecast the time to near-collision, few seconds before the near-collision event. While most of the existing datasets for human trajectory prediction are from a fixed overhead camera [20], [24], our dataset of **13,658** video segments targets the first-person view which is more intuitive for mobile robots. In the work on robust multi-person tracking from mobile platforms [6], the dataset is egocentric at walking speed but


Figure 1.1: The left image shows an assistive suitcase with a camera sensor and speaker to guide people. The right image shows the corresponding suitcase-shaped training prototype mounted with stereo camera and LIDAR for data collection.

with 4788 images it is insufficient to deploy the success of convolutional neural networks.

We formulate the forecasting of time to near-collision as a regression task. To learn the mapping from spatial-temporal motion of the nearby pedestrians to time to near-collision, we learn a deep network which, takes a sequence of consecutive frames as input and outputs the time to near-collision. To this end, we evaluate and compare two popular video network architectures in the literature: (1) The high performance of the image-based network architectures makes it appealing to reuse them with as minimal modification as possible. Thus, we extract the features independently from each frame using an image-based network architecture (*e.g.*, VGG-16) and then aggregate the features across the temporal channels; (2) It is also natural to directly use a 3D ConvNet (*e.g.*, I3D [4]) to learn the seamless spatial-temporal features.

Moreover, it is a nontrivial task to decide how many past frames should form the input. Thus, we do extensive experimentation on different temporal windows as input using aforementioned video network architectures. Our results show that our proposed multi-stream CNN trained on the collected dataset is the best model for predicting time to near-collision.

In summary, the contributions of our work are as follows: (1) We contribute a large-scale dataset of **13,658** egocentric video snippets of humans navigating in indoor hallways. In order to obtain ground truth annotations of human pose, the videos are provided with the corresponding 3D point cloud from LIDAR; (2) We explore the possibility of forecasting the time to near-collision directly from a single RGB camera; (3) We provide an extensive analysis on how current state-of-the-art video architectures perform on the task of predicting time to near-collision on the proposed dataset and how their performance varies with different temporal windows as input.

# Chapter 2

# Related Work

## 2.1 Monocular-Based Collision Avoidance

Existing monocular collision avoidance systems mostly focus on avoiding the immediate collision at the current time instant. Learning to fly by crashing [9] presented the idea of supervised learning to navigate an unmanned aerial vehicle (UAV) in indoor environments. The authors create a large dataset of UAV crashes and train an AlexNet [15] with single image as input to predict from one of these three collision avoidance actions - go straight, turn left or turn right. Similarly, DroNet [17] trains a ResNet-8 [11] to safely navigate a UAV through the streets of the city. DroNet takes the images from an on-board monocular camera on UAV and outputs a steering angle along with the collision probability.

While predicting an action to avoid the immediate collision is useful, it is more desirable to predict a possible collision in the short future.

Therefore, our work focuses on forecasting the exact time to a possible collision occurring within next 6 seconds, which will be helpful for collision avoidance in the context of dynamic path planning [21], [16], [30]. For planning in the presence of dynamic obstacles such as pedestrians, Safe Interval Path Planning (SIPP) [21] defines a safe interval as a contiguous period of time during which there is no collision. While the number of timesteps may be unbounded, the number of safe intervals is finite making the search space tractable for the planner. Our approach finds the time-to-collision in real-time and thus can be useful for SIPP planner to define safe interval. Another path planning algorithm called time-bounded lattice [16] merges together short-term planning in time with less expensive long-term planning without time. This planner computes a single bound for how long the planning should be done in time. The near-collision time predicted by our approach can thus serve the purpose of deciding the time bound for the short-term spatio-temporal planning.

## 2.2 Predicting Time to Collision by Human Trajectory

Instead of predicting the time to collision directly from images, one can also predict the human trajectories [10], [1], [31], [20], [33] as the first step, then the time to collision can

easily be computed from the predicted trajectories. [20] introduces a dynamic model for human trajectory prediction in a crowd scene by modeling not only the history trajectory but also the surrounding environment. [1] proposes a LSTM model to learn general human movement pattern and thus can predict the future trajectory. As there are many plausible ways that humans can move, [10] proposes to predict diverse future trajectories instead of a deterministic one. [31] proposes an attention model, which captures the relative importance of each surrounding pedestrian when navigating in the crowd, irrespective of their proximity.

However, in order to predict future trajectory reliably, these methods rely on accurate human trajectory history. This usually involves multi-people detection and tracking, and thus has two major disadvantages: (1) Data association is very challenging in crowded scenarios. Small mistakes can be misinterpreted to be very large changes of velocity, resulting in very bad trajectory estimate; (2) Robust multi-person tracking from mobile platform is often time-consuming. For example, [6] takes 300ms to process one frame on a mobile GPU, making it impossible to achieve real-time collision forecasting.

In contrast, our approach can predict the time to collision directly from a sequence of images, without requiring to track the surrounding pedestrians explicitly. We demonstrate that our data-driven approach can implicitly learn reliable human motion and also achieve real-time time to collision.

## 2.3 Learning Spatio-Temporal Feature Representation

Existing video architectures for spatio-temporal feature learning can be split into two major categories. To leverage the significant success from image-based backbone architectures (*e.g.*, VGG and ResNet) pre-trained on large-scale image datasets such as ImageNet [5] and PASCAL VOC [7], methods in the first category reuse the 2D ConvNet to extract features from a sequence of images with as minimal modifications as possible. For example, [18] proposes to extract the image-based features independently from each frame using GoogLeNet and then apply a LSTM [12] on the top for feature aggregation for video action recognition.

The second category methods explore the use of 3D ConvNets for video tasks [29], [8], [22], [26] that directly operate 3D spatio-temporal kernels on video inputs. While it is natural to use 3D ConvNets for spatio-temporal feature learning, 3D ConvNets are unable to leverage the benefits of ImageNet pretraining directly and often have huge number of parameters which makes it most likely to overfit on small datasets. Recently, the two-stream inflated 3D ConvNet (I3D) [4] is proposed to mitigate these disadvantages by inflating the ImageNet-pretrained 2D weights to 3D. Also, the proposed large-scale video dataset, Kinetics, has shown to be very successful for 3D kernel pre-training.

To validate how current state-of-the-art video architectures perform on the task of predicting time to near-collision on the proposed dataset, we evaluate methods from both categories in our experiments.

# Chapter 3

# Dataset

Table 3.1: Video Datasets with egocentric viewpoint

| Dataset | Number of near-collision video sequences | Structure of scenes | Setup for recording |
|---|---|---|---|
| **Ours** (Near-collision) | 13,685 | Indoor hallways | Suitcase |
| UAV crashing [9] | 11,500 | Indoor hallways | UAV |
| DroNet [17] | 137 | Inner-city | Bicycle |
| Robust Multi-Person Tracking [6] | 350 | Busy inner-city | Chariot |

We sought to analyze a large-scale, real-world video dataset in order to understand challenges in prediction of near-collision events. However, based on our survey, existing datasets had a small number of interaction events as reported in Table 3.1 and lacked diversity in the capture settings. Therefore, in order to train robust CNN models that can generalize across scenes, ego-motion, and pedestrian dynamics, we collected an extensive dataset from a mobile perspective. Next, we describe our hardware setup and methodology for data collection.

## 3.1 Hardware Setup

The mobile platform used for data collection is shown in Fig. 1.1, and includes a stereo camera and LIDAR sensor. While during inference we only utilize a monocular video, the stereo camera serves two purposes. First, it provides a depth map to help in automatic ground truth annotation. Second, it doubles the amount of training data by providing both a left and right image perspective which can be used as separate training samples. However, during the process of automatic data annotation, it was observed that the depth maps from stereo camera are insufficient for extracting accurate distance measurements. In particular, when the pedestrian is close to camera the depth values are missing at corresponding pixels due to motion blur. To resolve this issue we utilize a LIDAR sensor, which is accurate to

within a few centimeters. The images and corresponding 3D point clouds are recorded at the rate of 10Hz.

## 3.2 Camera-LIDAR Extrinsic Calibration

We use the camera and the LIDAR for automatic ground truth label generation. The two sensors can be initially calibrated with correspondences [13, 32]. An accurate calibration is key to obtaining the 3D position of surrounding pedestrians and annotating the large number of videos in our dataset. Let $R$ and $t$ denote the rotation matrix and the translation vector defining the rigid transformation between the LIDAR to the camera frame and $K$ the $3 \times 3$ intrinsic matrix of camera. Then, the LIDAR 3D coordinates $(x, y, z)$ can be related to a pixel in the image with coordinates $(U, V) = (\frac{u}{w}, \frac{v}{w})$ using following transformation:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K[R \mid -R^T t] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.1}$$

Given this calibration, we can now project LIDAR points onto the image and obtain estimated depth values for the image-based pedestrian detection.

## 3.3 Data Collection and Annotation

The platform is pushed through three different university buildings with low-medium density crowd. Our recorded videos comprise of indoor hallways of varying styles. We experimented with several techniques for obtaining pedestrian detections in the scene from the image and LIDAR data. As 2D person detection is a well-studied problem, we found an image-based state-of-the-art person detection (Faster-R-CNN [23]) to perform well in most cases, and manually inspect and complete any missing detections or false positives. To obtain the 3D position of each detected bounding box, we compute a median distance of its pixels using the 3D point cloud. An illustration of the resulting processing is shown in Fig. 3.1. Each image is annotated with a binary label where a positive label indicates the presence of at least one person within a meter distance from setup. We understand that the people in camera's view who move in the same direction as camera might not be important for collision. However, the number of such instances is insignificant in our dataset.

Now we want to estimate the time to near-collision, in terms of milliseconds, based on a short temporal history of few RGB frames. Let us consider a tuple of $N$ consecutive frames $(I_1, I_2, \ldots, I_N)$ and using this sequence as history we want to estimate if there is a proximity over the next 6 seconds. Since the framerate is 10 fps, we look at the next 60 binary labels in future annotated as $\{label_{n+1}, label_{n+2}, \ldots, label_{n+60}\}$. If we denote the index of first positive label in this sequence of labels as $T$ then our ground truth time to near-collision is $t = \frac{T}{10}$ seconds.

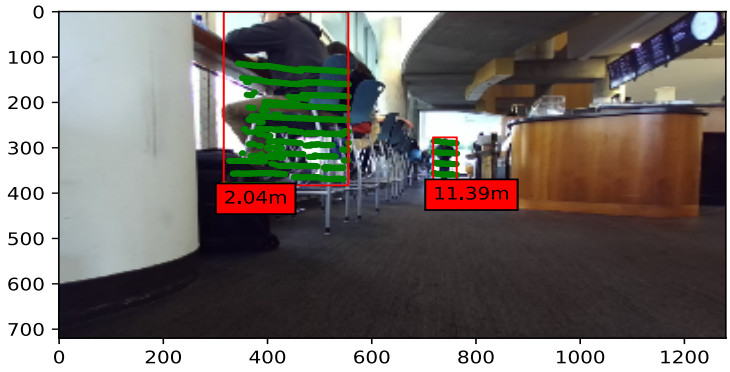


Figure 3.1: Multi-modal ground truth generation. The two red bounding boxes indicate people detected by Faster R-CNN. The green points are projected to the image from the LIDAR, with the relative distance between LIDAR and person shown as well.

## 3.4 Comparison with Existing Datasets

In Table 3.1, we compare our proposed dataset with existing datasets recorded from egocentric viewpoint in terms of (1) number of near-collision video sequences, (2) structure of scenes, and (3) setup used for recording. UAV crashing [9] dataset is created by crashing the drone 11,500 times into random objects. DroNet [17] has over 137 sequences of starting far way from an obstacle and stopping when the camera is very close to it. The two main reasons for collecting proposed dataset over existing datasets of UAV crashing and DroNet are: (1) applicability to assistive suitcase system [14], and (2) focus on pedestrian motion. While the dataset provided by Ess et al [6] suited to our application, we find only 350 near-collision instances making it infeasible to deploy CNNs.

# Chapter 4

# Approach

Our goal is to predict the time at which at least one person is going to come within a meter distance of the mobile setup using only a monocular image sequence of $N$ frames. The video is recorded at $10$ fps and thus a sequence of $N$ frames, including the current frame and past $(N-1)$ frames, correspond to the history of $\frac{N-1}{10}$ seconds. We first provide a formal definition of the task and then the details of network architecture for reproducibility.

## 4.1 Problem Formulation - Classification or Regression?

Learning time to near-collision can be formulated as a multi-class classification into one of the 60 classes where $i^{th}$ class corresponds to time range between $(\frac{i-1}{10}, \frac{i}{10}]$ seconds. The disadvantage of training it as a classification task is that all the mispredictions are penalized equally. For example, let us consider two different mispredictions given the same ground truth of 0.5 seconds - one where the network categorized it into the class $(0.6, 0.7]$ and other when the network predicted $(5.5, 5.6]$. The multi-class cross-entropy loss on both of these will be equal while we want the latter to be penalized much more than the former. Thus, we formulate it as a regression problem and use the mean-squared error as the loss function. In this work, we formulated it as a regression problem as follows:

$$t = f(I_1, I_2, \ldots, I_N) \text{ where } t \in [0, 6]$$

Here, we make short term predictions of 6 seconds into the future as it is sufficient time for notifying blind users via speaker.

## 4.2 Network Architecture

VGG-16 [27] is a 16-layer convolutional neural network which won the localization task in ImageNet Challenge 2014. It used parameter efficient $3 \times 3$ convolutional kernels pushing the depth to 16 weight layers. It was shown that its representations generalize well to other datasets achieving state-of-the-art results. We propose a multi-stream VGG architecture as shown in Fig. 4.1 where each stream takes a $224 \times 224$ RGB frame as input to extract spatial features . These spatial features are then concatenated across all frames preserving the temporal order and then fed into a fully-connected layer to output time to collision.

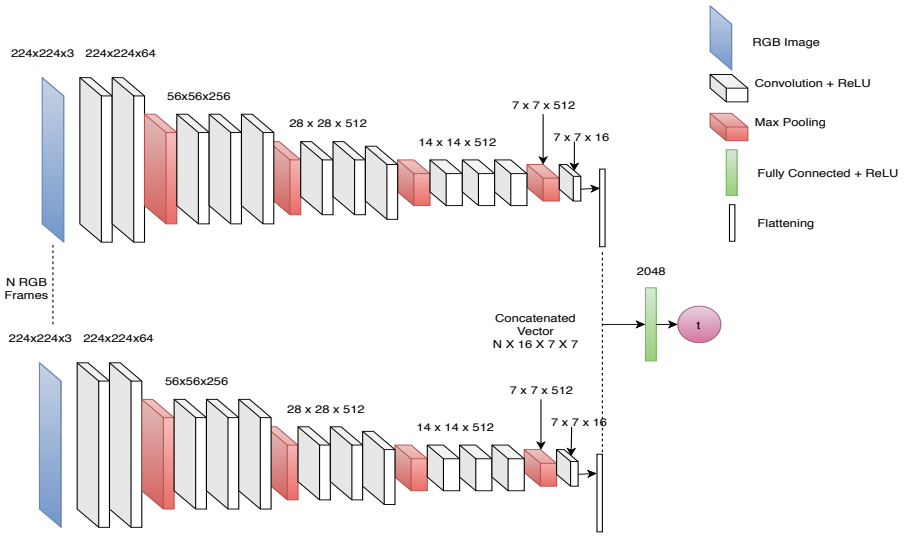


Figure 4.1: Our model with VGG-16 as backbone where the final output is time to collision denoted as $t$

**Feature Extraction from VGG-16**

We extracted the features of dimensions $7\times7\times512$ from the last max pool layer of VGG-16. These features pass through an additional convolution layer to reduce the feature size to $7\times7\times16$ and then flattened. These flattened features for each frame are concatenated into a vector and fed into the successive fully-connected layer of size 2048 which finally leads to a single neuron denoted as $t$ in Fig. 4.1.

In this network, the convolutional operators used spatial 2D kernels. A major question in current video architectures is whether these 2D kernels should be replaced by 3D spatio-temporal kernels [4]. To answer this we also experimented with 3D spatio-temporal kernels and report the results in chapter 5.

**Training N-stream VGG**

We initialized the VGG-16 network using ImageNet-pretrained weights. As the ImageNet dataset does not have a person class, we fine-tuned the network weights on PASCAL VOC [7] dataset. Using these weights as initialization, we train a multi-stream architecture with shared weights. The network is trained using the following loss function.

$$L_{MSE} = \frac{1}{2}||t_{true} - f(I_1, I_2, \ldots, I_N)||^2$$

Here, $L_{MSE}$ is the mean squared loss between the predicted time, *i.e.*, $f(I_1, I_2, \ldots, I_N)$ and ground truth time denoted as $t_{true}$.
The loss is optimized using mini-batch gradient descent of batch size 24 with the learning rate of $0.001$. The training data is further doubled by applying horizontal flip transformation.

The complete approach implemented in real-time using a single GPU is summarised in the flow diagram shown in Fig. 4.2.

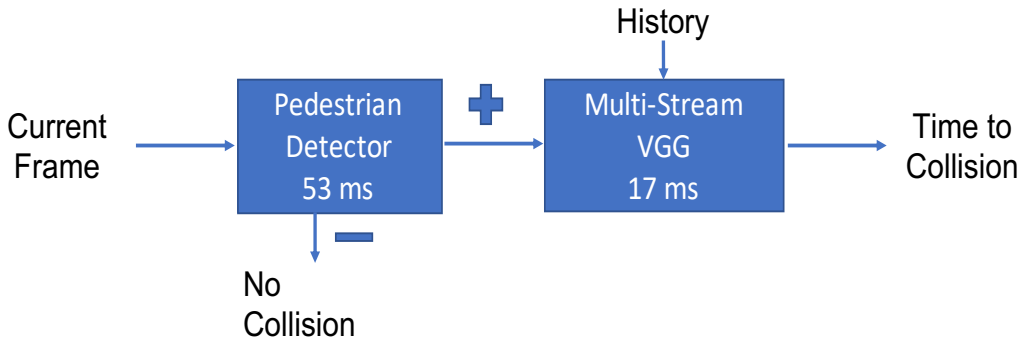


Figure 4.2: Summary of complete approach - The current frame passes through a pedestrian detector. If a person is detected, the cached history frames along with the current frame is passed through the proposed multi-stream network to output time to near-collision.

# Chapter 5

# Experimental Evaluation

## 5.1 Initial Experimentation - Classification

Here we report our initial experiments where we formulated the task of predicting near-collision time as a classification model.

### 5.1.1 Binary Classification: What happens 1 second into the future?

In initial experimentation, we formulated the task as binary classification whether there is a near-collision in next one second or not. The output layer in Fig. 4.1 is replaced by a two-neuron layer followed by softmax function. The binary cross entropy loss is used to train the network. An alternative naive way to solve this task is to compare the foot of pedestrians' bounding boxes with a predefined threshold on pixel's vertical image coordinate. The Table 5.2 compares the F1 score from our learning approach versus the naive baseline. F1 score is the harmonic mean of precision and recall and thus higher F1 score is more accurate. For the naive baseline, if the foot of the pedestrian lies in the lower $37.5\%$ (empirically found to be best) of the vertical size of image, we classify it as collision within one second.

We use GradCAM [25] to see which part of the image our trained network attends to. The attention maps of few test examples are shown in Figures 5.1, 5.2. In the attention maps, red indicates the region where the network paid most attention, i.e., the activations are highest and blue indicates the least attention. We observe that the network is able to estimate the future trajectory of person in scene, in most cases, a region on the floor. In cases as shown in top-right of Fig. 5.1, multiple trajectories are plausible relative to the mobile platform and the network learns to pay more attention to one of the trajectories either based on environment structure or social compliance implicit in the training data.

To derive these attention maps, GradCAM [25] first computes the gradient of the score $y^c$ for a class $c$ with respect to the activations of last convolutional layer. The gradients are global-average-pooled to get the importance of a feature map $k$ for class $c$, denoted as ${\alpha_k}^c$.

$$ {\alpha_k}^c = \frac{1}{Z} \sum_i \sum_j \frac{\delta y^c}{\delta {A_{ij}}^k} $$

A weighted combination of forward activation maps is finally upsampled to the size of

input image to visualize the attention map.

$$L^c = ReLU(\sum_k \alpha_k^c A_k)$$

${\alpha_k}^c$ : Weight for $k^{th}$ channel in last convolutional map for class $c$

$A_{ij}^k$ : Activation of $(i, j)$ in $k^{th}$ convolutional channel

$L^c$ : Attention map for class $c$

The confusion matrix over a test set passed through the trained network is reported in table 5.1. To tackle the data imbalance between the collision instances (2106) and no collision instances (8579), a weighted sampler is used in training with collision instances having a weight of $0.6$ and no collision instances having a lower weight of $0.4$.

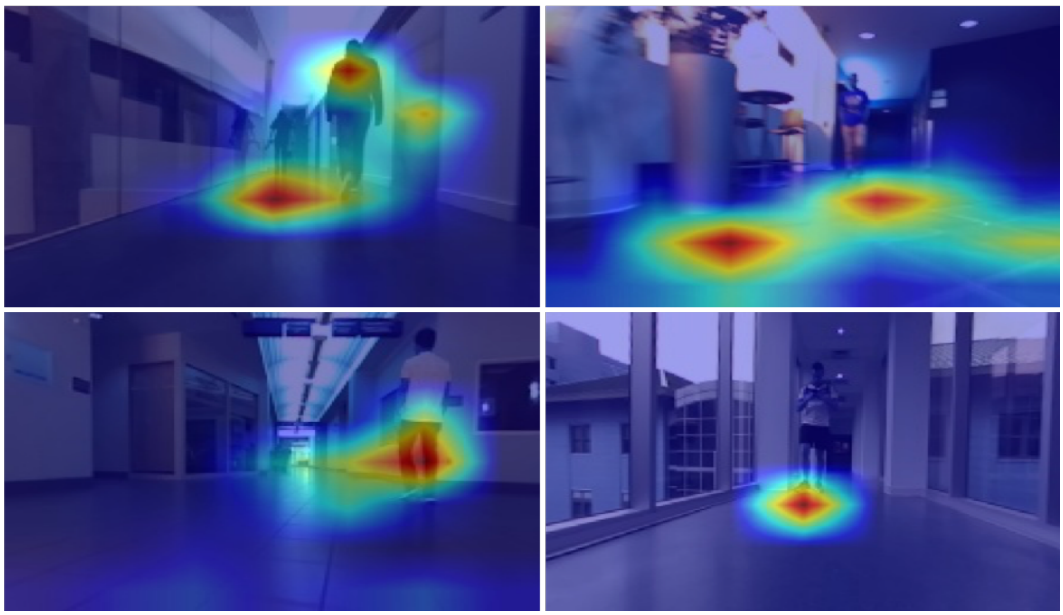

Figure 5.1: Visualization of attention maps using GradCAM; Top Left: Two Persons going away, Top Right: One person with two probable trajectories, Bottom Left: A person going away, Bottom Right: A person approaching

### 5.1.2 Multi-label Classification

On previous task of binary classification for 0-1 sec, our deep learning approach performs better than the naive thresholding as reported in Table 5.2. We thus move on to increasing the difficulty of the task, i.e, a multi-label classification problem where we classify if there is going to be a near-collision instance in (1) within a second, (2) between 1-2 seconds, (2) between 2-3 seconds or (4) no collision within 3 seconds. Our training and test data distribution for this multi-label classfication is provided in Table 5.3. For training, we used multilabel soft margin loss [19] as the loss function. The F1 scores reported in Table 5.4 indicate that the naive baseline can give better predictions within 2 seconds into the future

Figure 5.2: Left: RGB Image, Right: Corresponding GradCAM

Table 5.1: Confusion Matrix from Binary Classification

| | | **Prediction outcome** | |
|---|---|---|---|
| | | **collision** | **no collision** |
| **Actual value** | **collision** | 634 | 36 |
| | **no collision** | 53 | 2840 |

while the proposed multilabel deep neural network performs better for the latter classes, i.e., predictions in the range 2-3 seconds and no collision instances. One of the challenges of this multi-label formulation over the regression formulation is that we have to empirically decide a threshold on the confidence score for each class to classify it as the positive or negative label. The precision-recall curve or area under ROC curve [2] can be used to evaluate the performance of the trained model at different thresholds and then decide the best threshold accordingly.

Table 5.2: Near-Collision Prediction formulated as Binary Classification: F1 Scores from our approach compared with a naive baseline

| Method | F1 Score |
|---|---|
| Naive baseline $(0.625Y)$ | 0.8488 |
| N-stream VGG $(N = 4)$ | **0.9344** |

Table 5.3: Class Distribution

| Near-Collision Time | Training Data | Test Data |
|---|---|---|
| 0-1 s | 4150 | 287 |
| 1-2 s | 2590 | 185 |
| 2-3 s | 2185 | 169 |
| After 3 s | 13049 | 507 |

## 5.2 Regression Experiments - Different temporal windows as input

Overcoming the limitations of classification formulation, we move on to regression formulation which performed better as previously explained in chapter 4.

A single image can capture spatial information but no motion characteristics. Thus, we propose to use a sequence of image frames as history. By feeding $N$ image frames, we consider a history of $\frac{N-1}{10}$ seconds. The temporal window of input frames was gradually increased from 2 frames (0.1 sec) to 9 frames (0.8 sec). We now describe our evaluation procedure to decide the optimum temporal window as input on two different video network architectures. We further compare the performance with strong collision prediction baselines. To quantify the performance, we measure the mean absolute error (MAE) for the predictions on the test set of $1000$ instances and the standard deviation in error. From Table 5.6, it is empirically concluded to use a temporal window of 0.5 seconds, i.e, 6 frames for most accurate predictions.

We also report the error at different time-to-collision intervals in Table 5.5. It is easiest to forecast the collisions a second away and hence the error is least. In general, as forecasting horizon increases it keeps on getting more difficult to forecast the exact time-to-collision.

## 5.3 Experimental comparison of architectures

We show a comparison of the performance of multi-stream VGG model and baselines including state-of-the-art methods in Table 5.7.

1. *Constant Baseline:* On the training data of 12,620 samples, we compute the mean time to near-collision denoted by $\mathbb{E}[y_{true}]$ as a weak baseline. For each test input, we predict $\mathbb{E}[y_{true}]$ which was found to be $2.23$ seconds.

2. *Tracking followed by constant velocity model:* In dynamic environments, pedestrians are often tracked using a stereo camera or LIDAR. By saving few previous locations (0.5-2 seconds), a linear regression fit is used to predict the velocity vector of person [14],

Table 5.4: Near-Collision Prediction formulated as Multi-Class Classification

| Near-Collision Time | Vertical Image Coordinate for naive baseline $Y = 720$ | F1 Score from Naive Baseline | F1 Score from N-Stream VGG |
|---|---|---|---|
| 0-1 s | $> 0.625Y$ | 0.849 | 0.759 |
| 1-2 s | $> 0.560Y$ | 0.720 | 0.630 |
| 2-3 s | $> 0.520Y$ | **0.629** | **0.947** |
| After 3 s | $\leq 0.520Y$ | **0.522** | **0.620** |

Table 5.5: How error in prediction varies with time-to-collision?

| Ground Truth Time-to-Collision Interval (in s) | Number of Test Samples | Mean Absolute Error in Predictions (in s) |
|---|---|---|
| 0-1 | 348 | 0.6027 |
| 1-2 | 211 | 0.6446 |
| 2-3 | 188 | 0.7547 |
| 3-4 | 147 | 0.7189 |
| 4-5 | 86 | 0.9502 |
| 5-6 | 58 | 1.8369 |

[21]. This velocity vector is then linearly extrapolated to predict where the pedestrian will be over the next 6 seconds and the corresponding accuracy is reported in Table 5.7. A major disadvantage in this method is the need for image-based tracking which is less reliable at low framerate of 10 fps.

3. *Deep learning for collision avoidance:* Collision avoidance using deep learning has been previously proposed in [9] and [17]. Gandhi et al [9] created a UAV crash dataset to train AlexNet for classifying the current image into one of these three categories of drone policy: go straight, turn left or turn right. For learning time to near-collision using their approach, we take a single image as input and use AlexNet architecture with ImageNet-pretrained weights as initialization. The only difference lies in the output layer which is a single neuron regression instead of a three-neuron classifier. Our multi-stream VGG outperformed current-frame AlexNet as reported in Table 5.7.

   ResNet-8 architecture used in DroNet [17] takes in the single image and after the last ReLU layer splits into two fully-connected streams outputting steering angle and collision probability respectively. To experiment with their learning approach, we used ResNet-8 architecture with only one output, i.e., time to near-collision. The performance is close to the constant velocity prediction model as reported in Table 5.7 and thus it can be seen that it is unable to leverage real-world data. One of the reasons for its low performance could be the unavailability of ImageNet-pretrained weights for ResNet-8 architecture and thus it has to be trained from scratch on our dataset which is much smaller than the Udacity's car-driving dataset of 70,000 images used for training steering angle stream in DroNet.

4. *I3D for action classification in videos:* Two-Stream Inflated 3D ConvNet (I3D) [4] is a

Table 5.6: Distribution of absolute error (mean $\pm$ std) on near-collision dataset using different number of input frames

| Number of frames | Multi-stream VGG | I3D |
|---|---|---|
| 1 | 0.879 $\pm$ 0.762s | 0.961 $\pm$ 0.707s |
| 2 | 0.828 $\pm$ 0.739s | 0.879 $\pm$ 0.665s |
| 3 | 0.826 $\pm$ 0.647s | 0.914 $\pm$ 0.659s |
| 4 | 0.866 $\pm$ 0.696s | **0.811 $\pm$ 0.642s** |
| 5 | 0.849 $\pm$ 0.734 | 0.845 $\pm$ 0.658s |
| 6 | **0.753 $\pm$ 0.687s** | 0.816 $\pm$ 0.663s |
| 7 | 0.757 $\pm$ 0.722s | 0.848 $\pm$ 0.733s |
| 8 | 0.913 $\pm$ 0.732s | 0.811 $\pm$ 0.647s |
| 9 | 0.817 $\pm$ 0.738s | 0.855 $\pm$ 0.670s |

Table 5.7: Distribution of absolute error (mean $\pm$ std) on regression task compared with different baselines

| Method | Mean (in s) | Std (in s) |
|---|---|---|
| Constant baseline ($\mathbb{E}[y_{true}]$) | 1.382 | 0.839 |
| Tracking + Linear Model [14] | 1.055 | 0.962 |
| DroNet [17] | 1.099 | 0.842 |
| Gandhi et al [9] | 0.884 | 0.818 |
| Single Image VGG-16 | 0.879 | 0.762 |
| I3D (4 frames) [4] | 0.811 | 0.642 |
| **Multi-stream VGG (6 frames)** | **0.753** | **0.687** |

strong baseline to learn a task from videos. All the $N \times N$ filters and pooling kernels in ImageNet-pretrained Inception-V1 network [28] are inflated with an additional temporal dimension to become $N \times N \times N$. I3D has two streams - one trained on RGB inputs and other on optical flow inputs. To avoid adding the latency of optical flow computation for real-time collision forecasting, we only used the RGB stream of I3D. We fine-tuned the I3D architecture which was pre-trained on Kinetics Human Action Video dataset [4] on our near-collision dataset by sending $N$ RGB frames as input where $N = \{1, 2, \ldots, 8, 9\}$ as reported in Table 5.6. The outermost layer is modified from 400-neuron classifier to 1-neuron regressor. Since our $N$-frame input is smaller than the original implementation on 64-frame input, we decreased the temporal stride of last max-pool layer from 2 to 1. While 6 input frames were found to be the best for proposed multi-stream VGG network, we experimented again with the optimal history on I3D. The performance of I3D with varying number of input frames is reported in Table 5.6. For $N = 4, 6, 8$, I3D is found to give the best results among 1-9 frames though our multi-stream VGG prediction for $N = 6$ outperformed the I3D prediction in best case.

## 5.4 Qualitative Evaluation

From the plots shown in Fig. 5.3 we can observe that the predictions given by multi-stream VGG on 6 frames give smoother output as compared to undesired fluctuations in I3D out-

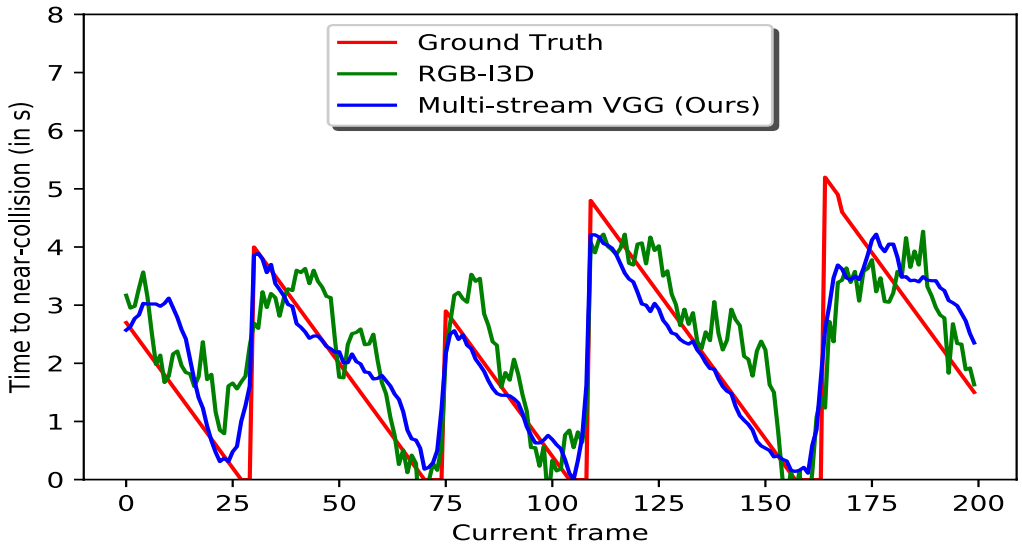


Figure 5.3: Prediction of multi-stream VGG is smoother than I3D. The discontinuities here arise in the absence of pedestrians.

put. One of the plausible reason for I3D performing worse could be the average pool layer in I3D which averages the features over input frames. On the other hand, the proposed network preserves the temporal order of features of input frames which helps in understanding the motion characteristics of people in scene. We also qualitatively show in Fig. 5.4 the comparison of time to near-collision predicted by our method vs the ground truth. From Fig. 5.4, we can observe the cases when there are two humans in the scene and the network's predictions are based on the closest person approaching the platform.

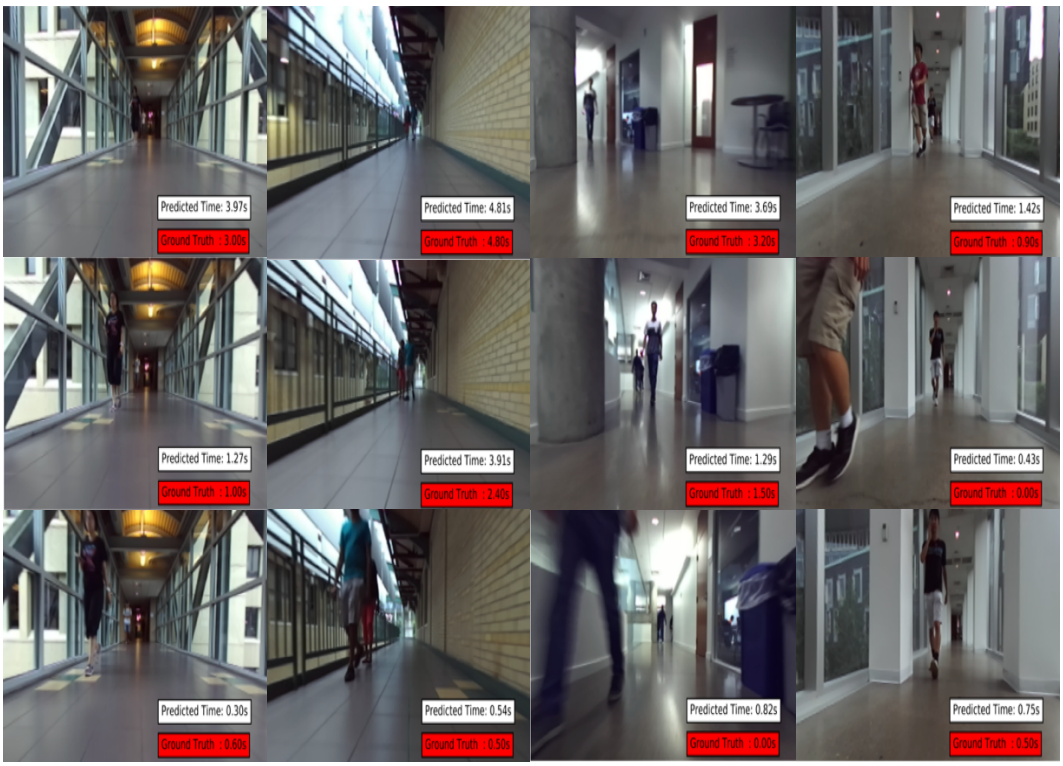


Figure 5.4: Predictions on four different test videos

# Chapter 6

# Conclusion

We return to the question posed in introduction, 'Is it possible to predict the time to near-collision from a single camera?'. The answer is that the proposed model is able to leverage spatio-temporal cues for predicting the time to collision within $0.75$ seconds on the test videos. Also, we observed that the multi-stream network of shared weights performed the task of collision forecasting better than I3D on the proposed dataset.

With regard to temporal window of input history, it is evident that using a sequence of images has a considerable benefit over prediction from single frame only. Though the history of 0.5 seconds performed best, we do not observe a piecewise monotonic relation between input frames and error in prediction. This observation aligns with the performance of constant velocity model where accuracy in prediction does not necessarily increase or decrease with the temporal footprint of past trajectory.

We understand that the proposed model might not generalize well when there are large changes in camera's height, speed of the robot or structure of the scenes in comparison to provided dataset. In the scenarios where assistive robot operates at walking speed and in a constrained domain like museums and airports, the proposed approach will be suitable for predicting time to collision requiring only a low-cost monocular camera. Inexpensive RGB-D sensors like Kinect cannot be used for outdoor navigation while our approach can be extended for outdoor use given training data.

# Chapter 7

# Future Work

In this work, the networks predicts a single real value as the time-to-collision and uses mean squared error loss for training. The current approach always outputs a real-valued estimate and there is no indicator on how confident or reliable is the prediction. In the future work, we can instead try to output two values - mean and variance in time-to-collision. The variance can tell us the confidence in predicted value. Lower the variance, stronger is the peak of predicted mean. To train the output tuple of mean and variance, the negative of gaussian log-likelihood loss can be minimized over the training data.

One of the scenarios where the proposed approach fails is when a person is moving in the same direction as user. The pedestrian detector detects the person and so the image is passed through the proposed multi-stream network. The network outputs an inaccurate estimate of time-to-collision as shown in Figure 7.1.


Figure 7.1: Inaccurate estimate of time-to-collision when a person is moving away from the assistive suitcase

One plausible way is to predict the variance term as mentioned above and hope that the variance will be large enough to not trust the prediction in such cases. As an alternate solution, instead of passing the image first through the pedestrian detector, we can pass it through a binary classifier trained to classify if there is going to be a collision within the chosen time bound (here, 6 seconds) or not. If positive, we can then pass it through the

regressor network as proposed in this work to estimate the exact time-to-collision. OpenPose [3], a real-time approach to detect the 2D pose of multiple people in an image including foot, torso, arms and face keypoints can possibly be leveraged to replace the object detection backbone, i.e, VGG-16 to enhance the prediction performance.

The prediction time bound in this work is chosen to be 6 seconds arbitrarily. Depending on the application or the planner requirements, the time bound could be chosen accordingly. For a higher time bound, it would be challenging to learn the task within an acceptable error limit. The Table 7.1 reports how the error varies with prediction time horizon.

Table 7.1: Mean Absolute Error in Predicted Time varying with the Prediction Time Horizon

| Prediction Time Horizon (in seconds) | Mean Absolute Error (in seconds) |
|---|---|
| 5 | 0.565 |
| 6 | 0.753 |
| 7 | 0.903 |

# Bibliography


[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[2] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[3] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1812.08008, 2018.

[4] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] A. Ess, B. Leibe, K. Schindler, and L. van Gool. Robust multiperson tracking from a mobile platform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1831–1846, Oct 2009.

[7] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[8] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016.

[9] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. *CoRR*, abs/1704.05588, 2017.

[10] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[13] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '18, 2018.

[14] S. Kayukawa, K. Higuchi, J. Guerreiro, S. Morishima, Y. Sato, K. Kitani, and C. Asakawa. Bbeep: A sonic collision avoidance system for blind travellers and nearby pedestrians. In *Proc. ACM CHI Conference on Human Factors in Computing Systems (CHI'19)*, CHI '19, New York, NY, USA, May 2019. ACM.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] A. Kushleyev and M. Likhachev. Time-bounded lattice for efficient planning in dynamic environments. In *2009 IEEE International Conference on Robotics and Automation*, pages 1662–1668, 2009.

[17] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, and D. Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, April 2018.

[18] J. Y.-H. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[20] S. Pellegrini, A. Ess, K. Schindler, and L. V. Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268, 2009.

[21] M. Phillips and M. Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5628–5635, May 2011.

[22] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5534–5542, 2017.

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press.

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

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[26] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 568–576, Cambridge, MA, USA, 2014. MIT Press.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[29] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, 2015.

[30] A. Vemula, K. Muelling, and J. Oh. Path planning in dynamic environments with adaptive dimensionality. In *Ninth Annual Symposium on Combinatorial Search*, 2016.

[31] A. Vemula, K. Muelling, and J. Oh. Social attention: Modeling attention in human crowds. In *Proceedings of the International Conference on Robotics and Automation (ICRA) 2018*, May 2018.

[32] L. Zhou, Z. Li, and M. Kaess. Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS*, October 2018.

[33] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3931–3936. IEEE, 2009.