Forecasting Time-to-Collision from Monocular Video: Feasibility, Dataset, and Challenges

Aashi Manglik, Xinshuo Weng, Eshed Ohn-Bar and Kris M. Kitani[1]

***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 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. The project webpage can be found at https://aashi7.github.io/NearCollision.html.**

I. 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 [1], [2], [3], [4] 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 [5], [6] for forecasting time to collision is to track [7], [8], [9], [10] the 3D location of the surrounding pedestrians and extrapolate their trajectories using a constant velocity model. 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 collision from a sequence of images. Using a monocular camera is an alternate solution to using inexpensive RGB-D cameras like Kinect which are inapplicable for outdoor use.

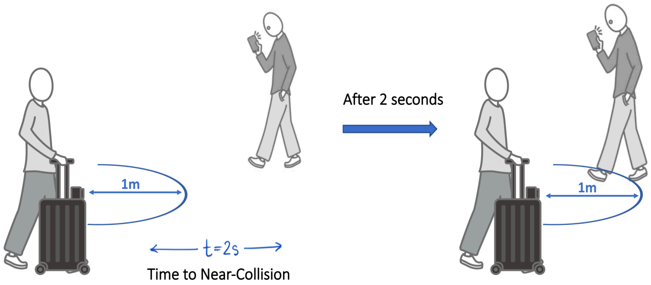

Fig. 1. Forecasting time to near-collision between a suitcase-shaped robot being pushed by its user and nearby pedestrian.

To create a dataset for learning time to collision, we designed a training prototype that facilitates efficient annotation of large amounts of data. 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 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. 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 [11], [12], 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 [7], the dataset is egocentric at walking speed but with 4788 images it is insufficient to deploy the success of deep learning.

We formulate the forecasting of time to near-collision as a regression task. To learn the mapping from spatial-

[1] Authors are affiliated with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {amanglik, xinshuow, eohnbar, kkitani}@andrew.cmu.edu. Eshed Ohn-Bar is now at the Max Planck Institute for Intelligent Systems.

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 [13]) 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 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 monocular images; (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.

II. Related Work

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 [1] 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 [14] with single image as input to predict from one of these three actions - go straight, turn left or turn right. Similarly, DroNet [2] trains a ResNet-8 [15] 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. To anticipate traffic accidents, Kataoka et al. [16] constructed a near-miss incident database of traffic scenes annotated with near-miss incident duration. By passing a video input to a quasi-recurrent neural network, their approach outputs a probability that an accident will occur in future. On an average, their approach is shown to anticipate a near-miss incident or accident 3.65 seconds in advance. While their study explored how early can a collision be anticipated, the exact time to collision was not predicted. 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 [5], [17], [18].

Predicting Time to Collision by Human Trajectory. Instead of predicting the time to collision directly from images, one can also predict the human trajectories [11], [19], [20], [21], [22] as the first step, then the time to collision can be predicted based on the speed and heading directions of all the surrounding pedestrians. [11] introduces a dynamic model for human trajectory prediction in a crowd scene by modeling not only the history trajectory but also the surrounding environment. [20] proposes a LSTM model to learn general human movement pattern and thus can predict better future trajectory. As there are many plausible ways that humans can move, [19] proposes to predict diverse future trajectories instead of a deterministic one. [21] 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 [23], [24], [25], [26] 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, [7] 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.

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 [27] and PASCAL VOC [28], methods in the first category reuse the 2D ConvNet to extract features from a sequence of images with as minimal modification as possible. For example, [29] proposes to extract the image-based features independently from each frame using GoogLeNet and then apply a LSTM [30] on the top for feature aggregation for video action recognition.

The second category methods explore the use of 3D ConvNets for video tasks [31], [32], [33], [34], [35] 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 easily 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) [13] 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-based architectures perform on the novel task of predicting time to near-collision on the proposed dataset, we evaluate methods from both categories in our experiments.

III. DATASET

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 I 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.

A. Hardware Setup

The mobile platform used for data collection is shown in Fig. 2, 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 ensure stable and reliable depth annotations, 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.

B. 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 [36], [37]. 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 3×3 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:

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

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

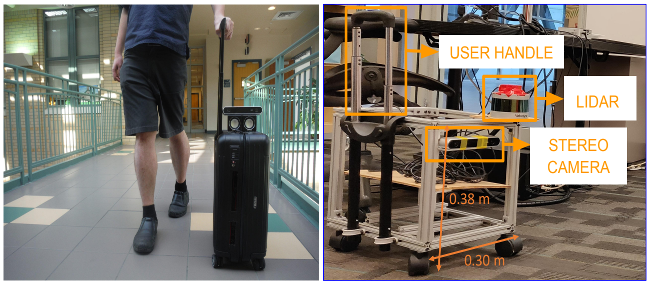

Fig. 2. **Left**: We show an assistive suitcase with a camera sensor and speaker to guide people. **Right**: We show the corresponding suitcase-shaped training prototype mounted with stereo camera and LIDAR for data collection.

C. Data Collection and Annotation

The platform is pushed through three different university buildings with low-medium density crowd. Our recorded videos comprise of hallways of varying styles. We experimented with several techniques for obtaining pedestrian detections [23], [24] 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. 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 some people in immediate vicinity moving in the same direction as camera might not be important for collision, but due to close proximity of 1 meter should still be recognized and planned over carefully.

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.

D. Comparison with Existing Datasets

In Table I, 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 [1] dataset is created by crashing the drone 11,500 times into random objects. DroNet [2] 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 [6], and (2) focus on pedestrian motion. While the dataset provided by

TABLE I

PUBLIC VIDEO DATASETS WITH EGOCENTRIC VIEWPOINT

Dataset	Number of near-collision video sequences	Structure of scenes	Setup for recording	Modalities Available
Ours (Near-collision)	13,685	Indoor hallways	Suitcase	Stereo Depth + 3D Point Cloud
UAV crashing [1]	11,500	Indoor hallways	UAV	Monocular
DroNet [2]	137	Inner-city	Bicycle	Monocular
Robust Multi-Person Tracking [7]	350	Busy inner-city	Chariot	Stereo Depth

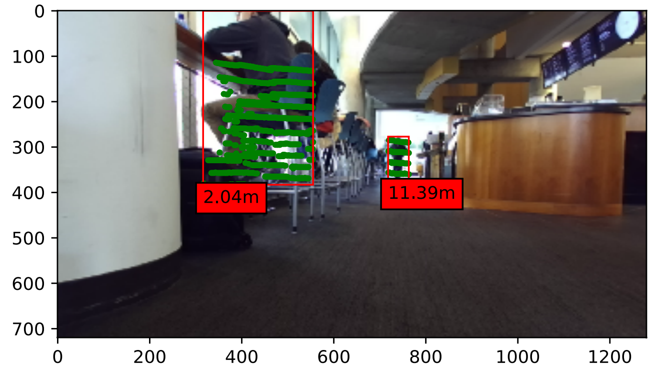

Fig. 3. Multi-modal ground truth generation. The two red bounding boxes indicate people detected by Faster R-CNN. The green points are projected to the image from the LIDAR, with the relative distance between LIDAR and person shown as well.

Ess et al [7] suited to our application, we find only 350 near-collision instances making it infeasible to deploy CNNs.

IV. 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 $(N-1)/10$ seconds. We first provide a formal definition of the task and then the details of network architecture for reproducibility.

A. *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 $((i-1)/10, i/10]$ 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. One of the solutions is to design a differentiable loss function which keeps this preference in mind. Another solution is to formulate it as a regression problem and use the mean-squared error as the loss function. In this paper, we formulated it as a regression problem as follows:

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

B. *Network Architecture*

VGG-16 [38] is a 16-layer convolutional neural network which won the localization task in ImageNet Challenge 2014. It used parameter efficient 3×3 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 where each stream takes a 224×224 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.

Feature Extraction from VGG-16 We extracted the features of dimensions $7{\times}7{\times}512$ from the last max pool layer of VGG-16. These features pass through an additional convolution layer to reduce the feature size to $7{\times}7{\times}16$ 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.

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 [13]. To address this question, we also experimented with 3D spatio-temporal kernels and report the results in following section.

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 [28] 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 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.

V. EXPERIMENTAL EVALUATION

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.

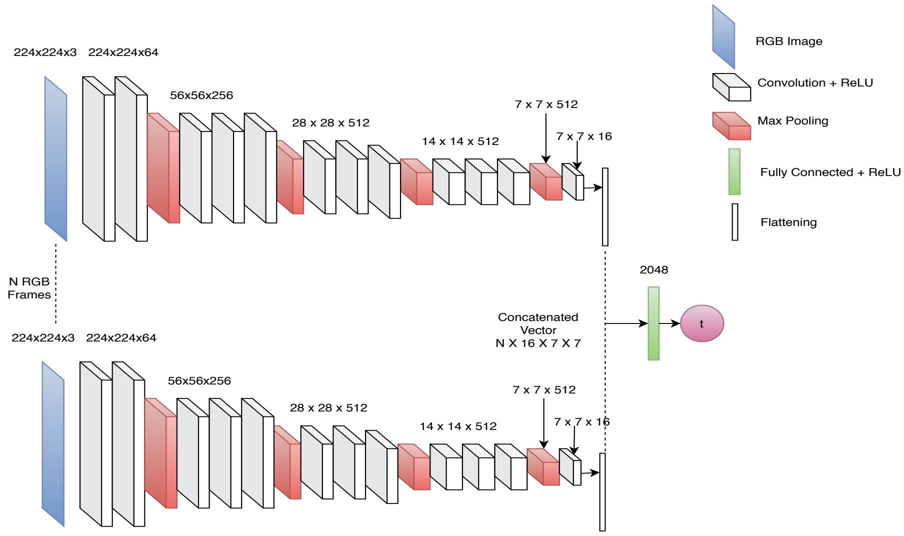

Fig. 4. Our model with VGG-16 as backbone where the final output is time to collision denoted as t

A. *Different temporal windows as input*

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 $(N-1)/10$ seconds. The temporal window of input frames was gradually increased from 2 frames (0.1 sec) to 9 frames (0.8 sec). To quantify the performance, we measure the mean absolute error (MAE) for the predictions on the test set and the standard deviation in error. From Table II, it is empirically concluded to use a temporal window of 0.5 seconds, i.e, 6 frames in multi-stream VGG for most accurate predictions.

TABLE II
DISTRIBUTION OF ABSOLUTE ERROR (MEAN ± STD) ON NEAR-COLLISION DATASET USING DIFFERENT NUMBER OF FRAMES

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

B. *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 III.

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 [5], [6]. 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 III. A major disadvantage in this method is the need for image-based tracking which is less reliable at low framerate of 10 fps.

TABLE III
DISTRIBUTION OF ABSOLUTE ERROR (MEAN ± 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 [6]	1.055	0.962
DroNet [2]	1.099	0.842
Gandhi et al [1]	0.884	0.818
Single Image VGG-16	0.879	0.762
I3D (4 frames) [13]	0.811	0.642
Multi-stream VGG (6 frames)	**0.753**	**0.687**

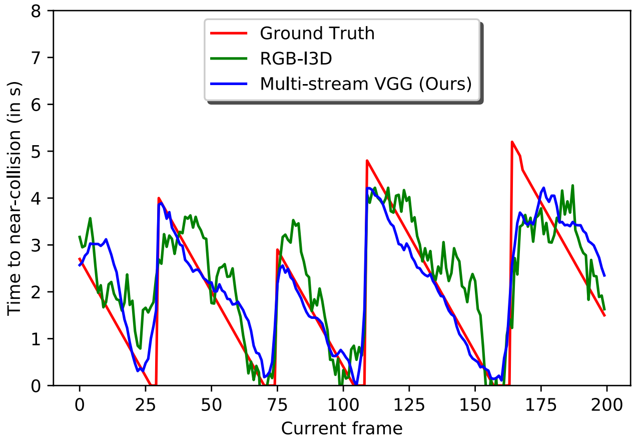

Fig. 5. Example prediction results over consecutive frames. The discontinuities in the ground truth occur when the nearest pedestrian exits the view. As shown, the multi-stream VGG approach better adheres to the ground truth compared to I3D both when the pedestrian is far and near the camera, while providing smoother and more temporally-consistent predictions.

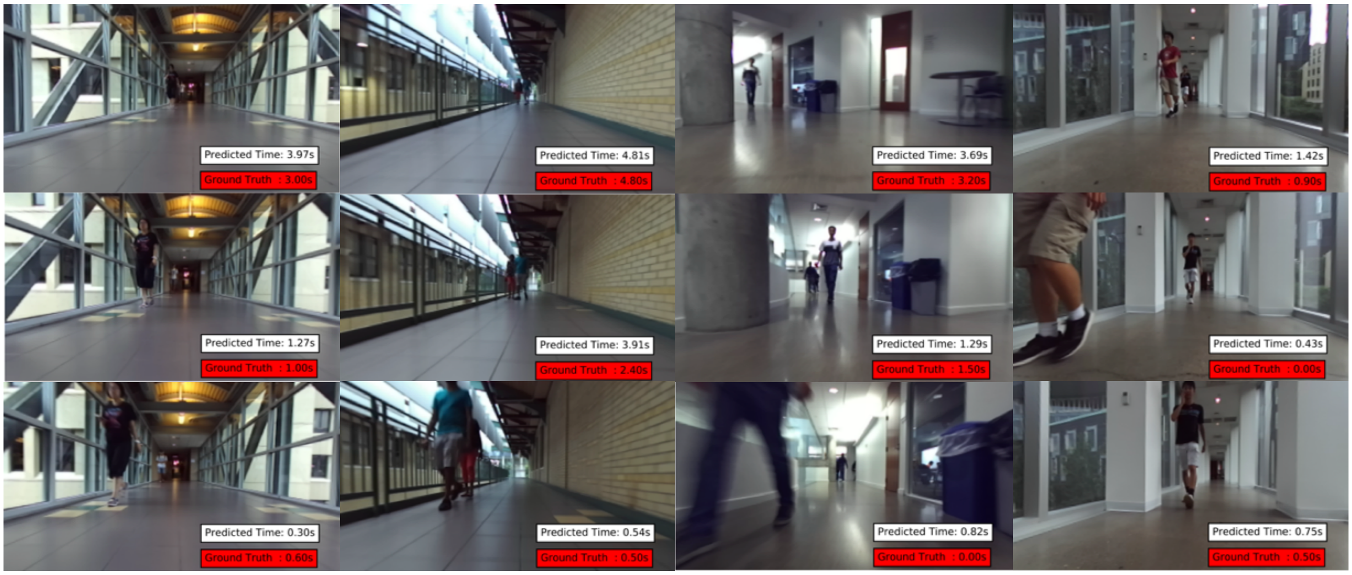

Fig. 6. Predictions on four different test videos

3) Deep learning for collision avoidance: Collision avoidance using deep learning has been previously proposed in [1] and [2]. Gandhi et al [1] 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 III.

ResNet-8 architecture used in DroNet [2] 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 III 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) [13] is a strong baseline to learn a task from videos. All the $N \times N$ filters and pooling kernels in ImageNet-pretrained Inception-V1 network [39] 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 [13] on our near-collision dataset by sending N RGB frames as input where $N = \{1, 2, \ldots, 8, 9\}$ as reported in Table II. 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 II. 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.

C. Qualitative Evaluation

From the plots shown in Fig. 5 we can observe that the predictions given by multi-stream VGG on 6 frames give smoother output as compared to undesired fluctuations in I3D output. We also qualitatively show in Fig. 6 the comparison of time to near-collision predicted by our method vs the ground truth.

D. Forecast Horizon

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

TABLE IV
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

VI. Discussion

We return to the question posed in introduction, 'Is it possible to predict the time to 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. Also, we observed that the multi-stream network of shared weights performed the task of collision forecasting better than I3D on the proposed dataset.

Regarding the 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 a temporal window incorporating 0.5 seconds of history performed best for our task, we do not observe a piece-wise 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.

During data collection, the pushing speed of suitcase varies between $0.2-1.5$ meters per second which is similar in range to human walking speed. We understand that the proposed model might not generalize well when there are significant changes in camera's height, pushing speed of suitcase or structure of the scenes in comparison to provided dataset. In the scenarios where assistive robot operates 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. We believe that the proposed approach could be extended for outdoor use given enough training data of outdoor scenes.

One of the scenarios where the current approach fails is when a person is moving away from the user in the same direction. The pedestrian detector detects the person and thus the image is passed through the proposed multi-stream network. The network outputs an inaccurate estimate of time-to-collision within the trained forecast horizon. In the future work, we can instead try to output two values - mean and variance in time-to-collision. The variance can act as an indicator on how confident or reliable is the prediction. To train the output tuple of mean and variance, the negative of gaussian log-likelihood loss can be minimized over the training data which includes such failure cases.

ACKNOWLEDGMENT

This work was sponsored in part by NIDILRR (90DPGE0003), JST CREST (JPMJCR14E1) and NSF NRI (1637927).

References

[1] D. Gandhi, L. Pinto, and A. Gupta, "Learning to Fly by Crashing," *IROS*, 2017.

[2] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *Robotics and Automation Letters*, 2018.

[3] F.-H. Chan, Y.-T. Chen, Y. Xiang, and M. Sun, "Anticipating accidents in dashcam videos," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 136–153.

[4] Q. Fu, N. Bellotto, H. Wang, F. C. Rind, H. Wang, and S. Yue, "A Visual Neural Network for Robust Collision Perception in Vehicle Driving Scenarios," *arXiv:1904.02074*, 2019.

[5] M. Phillips and M. Likhachev, "SIPP: Safe Interval Path Planning for Dynamic Environments," *ICRA*, 2011.

[6] 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," *CHI*, 2019.

[7] A. Ess, B. Leibe, K. Schindler, and L. van Gool, "Robust Multi-Person Tracking from a Mobile Platform," *TPAMI*, 2009.

[8] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," *ICIP*, 2016.

[9] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna, "Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking," *ICRA*, 2018.

[10] X. Weng and K. Kitani, "A Baseline for 3D Multi-Object Tracking," *arXiv:1907.03961*, 2019.

[11] S. Pellegrini, A. Ess, K. Schindler, and L. V. Gool, "You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking," *ICCV*, 2009.

[12] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, "TrajNet: Towards a Benchmark for Human Trajectory Prediction," *arXiv:1805.07663*, 2018.

[13] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *CVPR*, 2017.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," *NIPS*, 2012.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CVPR*, 2016.

[16] T. Suzuki, H. Kataoka, Y. Aoki, and Y. Satoh, "Anticipating Traffic Accidents with Adaptive Loss and Large-Scale Incident DB," *CVPR*, 2018.

[17] A. Kushleyev and M. Likhachev, "Time-Bounded Lattice for Efficient Planning in Dynamic Environments," *ICRA*, 2009.

[18] A. Vemula, K. Muelling, and J. Oh, "Path Planning in Dynamic Environments with Adaptive Dimensionality," *Ninth Annual Symposium on Combinatorial Search*, 2016.

[19] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," *CVPR*, 2018.

[20] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," *CVPR*, 2016.

[21] A. Vemula, K. Muelling, and J. Oh, "Social Attention: Modeling Attention in Human Crowds," *ICRA*, 2018.

[22] 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," *IROS*, 2009.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *NIPS*, 2015.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[25] X. Weng, S. Wu, F. Beainy, and K. Kitani, "Rotational Rectification Network: Enabling Pedestrian Detection for Mobile Vision," *WACV*, 2018.

[26] X. Weng and K. Kitani, "Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud," *arXiv:1903.09847*, 2019.

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

[28] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *IJCV*, 2010.

[29] J. Y.-H. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond Short Snippets: Deep Networks for Video Classification," *CVPR*, 2015.

[30] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computing*, 1997.

[31] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning SpatioTemporal Features with 3D Convolutional Networks," *ICCV*, 2015.

[32] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "SpatioTemporal Residual Networks for Video Action Recognition," *NIPS*, 2016.

[33] Z. Qiu, T. Yao, and T. Mei, "Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks," *ICCV*, 2017.

[34] K. Simonyan and A. Zisserman, "Two-stream Convolutional Networks for Action Recognition in Videos," *NIPS*, 2014.

[35] X. Weng and K. Kitani, "Learning Spatio-Temporal Features with Two-Stream Deep 3D CNNs for Lipreading," *arXiv:1905.02540*, 2019.

[36] 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," *ICCPS*, 2018.

[37] L. Zhou, Z. Li, and M. Kaess, "Automatic Extrinsic Calibration of a Camera and a 3D LiDAR Using Line and Plane Correspondences," *IROS*, 2018.

[38] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ICLR*, 2015.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *CVPR*, 2015.