Realtime Collision Avoidance for Mobile Robots in Dense Crowds using Implicit Multi-sensor Fusion and Deep Reinforcement Learning

Jing Liang*, Utsav Patel*, Adarsh Jagan Sathyamoorthy, and Dinesh Manocha.*†

ABSTRACT

We present a novel learning-based collision avoidance algorithm, CrowdSteer, for mobile robots operating in dense and crowded environments. Our approach is end-to-end and uses multiple perception sensors such as a 2-D lidar along with a depth camera to sense surrounding dynamic agents and compute collision-free velocities. Our training approach is based on the sim-to-real paradigm and uses high fidelity 3-D simulations of pedestrians and the environment to train a policy using Proximal Policy Optimization (PPO). We show that our learned navigation model is directly transferable to previously unseen virtual and dense real-world environments. We have integrated our algorithm with differential drive robots and evaluated its performance in narrow scenarios such dense crowds, narrow corridors, T-junctions, L-junctions, etc. In practice, our approach can perform real-time collision avoidance and generate smooth trajectories in such complex scenarios. We also compare the performance with prior methods based on metrics such as trajectory length, mean time to goal, success rate, and smoothness and observe considerable improvement.

KEYWORDS

Collision Avoidance; Deep Reinforcement Learning; Crowd Navigation; Sensor Fusion

1 INTRODUCTION

Mobile robots are frequently deployed in indoor and outdoor environments such as hospitals, hotels, malls, airports, warehouses, sidewalks, etc. These robots are used for surveillance, inspection, delivery, and cleaning, or as social robots. Such applications need to be able to smoothly and reliably navigate in these scenarios by avoiding collisions with obstacles, including dynamic agents or pedestrians.

Some earlier work on mobile robot navigation was limited to open spaces or simple environments with static obstacles. Over the last decade, there has been considerable work on collision-free navigation among pedestrians using visual sensors like lidars or cameras [6, 8, 12, 14, 21, 24]. However, many challenges arise when such robots are used in dense or cluttered environments and need to move at speeds that are close to that of human pedestrians (1.3 meters/sec). A high crowd density corresponds to 1-3 (or more) pedestrians per square meter. In these scenarios, the pedestrian trajectories are typically not smooth and may change suddenly. Moreover, it is difficult to predict their trajectories due to occlusion or non-smooth motion. Many sensor-based navigation algorithms either tend to stall the robot's motion or are unable to avoid collisions with the pedestrians.

Figure 1: A Turtlebot and Jackal robot using CrowdSteer to navigate in scenarios with pedestrians in a narrow corridor and areas with high occlusion. Our method uses data from multiple sensors such as a 2-D lidar and a depth camera to generate smooth collision avoidance maneuvers. We compare with methods such as DWA[11] and Fan et al[17].

A recent trend is to use learning methods for sensor-based robot navigation in crowds. These include techniques based on end-to-end deep learning [14, 21], generative adversarial imitation learning [24], and deep reinforcement learning [10, 17]. Most of these methods use one or more perception sensors like an RGB, or RGB-D camera, or a 2-D lidar.

In practice, using a single robot sensor may not work well in dense or cluttered scenarios. Moreover, this sensor choice affects the efficiency of the collision avoidance scheme in terms of reaction time or the optimality of the trajectory. Some of these methods work well in static environments, but fail in scenarios with even a few dynamic obstacles [24]. Algorithms that use a lidar may perform well in dense scenarios but lack the ability to detect thin obstacles such as poles or differentiate between animate and inanimate obstacles. They also suffer from the freezing robot problem and exhibit oscillatory behaviors in dense situations. Other algorithms [6, 7] use either 2-D or 3-D lidars along with several RGB cameras to detect

*Authors contributed equally.
†Video: https://youtu.be/N83Mg9oW-0c

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand

https://doi.org/doi

obstacles and predict pedestrian motions. They exhibit good performance in moderately dense crowds but may not work well with high density crowds, scenes with occlusions and are susceptible to perception errors.

Main Results: We present CrowdSteer, a novel learning-based local navigation method that uses hybrid sensing for densely crowded scenarios. We formulate the navigation among pedestrians as a POMDP problem and solve it using deep reinforcement learning. Our approach uses a combination of perception sensors such as a 2-D lidar along with an depth (RGB-D) camera to sense obstacle features and help the policy implicitly learn different kinds of interactions with the obstacles. We use the Proximal Policy Optimization (PPO) algorithm to train the collision avoidance policy and reduce the sim-to-reality gap by using high-fidelity complex simulations of environments with pedestrians. Some of the novel components of our approach include:

- A new end-to-end learning method that fuses inputs from multiple perception sensors for implicitly characterizing the robot's interactions with obstacles and pedestrians. This results in reliably handling complex, high-density scenarios for navigation.
- A trained model which generates smoother collision avoidance trajectories and is resilient to perception noises and occluded obstacles, as compared to previous methods.
- Custom made complex high-fidelity training and testing scenarios of indoor environments with pedestrians for training and testing Deep Reinforcement Learning (DRL) models. We show that using such simulations eases sim-to-real transfer and leads to better generalization of the trained policy.

We implemented and evaluated our algorithm on a Turtlebot robot and a ClearPath Jackal robot with a Hokuyo 2-D lidar and an Orbbec Astra camera in indoor environments such as corridors, L-junctions, and T-junctions with varying pedestrian density (as high as 1-2 humans/ m^2). We also compare CrowdSteer's performance with prior traditional methods such as Dynamic Window Approach (DWA) [11] and a state-of-the-art learning-based crowd navigation algorithm [17]. We observe that our approach surpasses these methods in terms of success rates, and shows a reduction of up to 68.16% in time to goal, and 6.12% reduction in trajectory length when compared to the current state of the art Fan et al. [17].

2 RELATED WORK

In this section, we give a brief overview of the prior work on traditional and learning-based navigation algorithms.

2.1 Navigation in Dynamic Scenes

There is extensive work on collision avoidance in dynamic scenes for robots. These include techniques based on potential-field methods [26], social-forces [13], velocity obstacles [1, 27], etc. These methods have been used in simulated environments and can scale to a large number of agents. In terms of real-world scenarios, these methods require accurate sensing of obstacles' positions and velocities and parameter tuning that is scenario-dependent. These requirements make it difficult to directly apply them for navigation in dense crowds. At the same time, these methods have been used to generate training trajectories for initializing some learning-based methods. The Dynamic Window Approach (DWA) [11] is another widely used method which calculates reachable dynamically-constrained velocities for collision avoidance within a short time interval. However, it does not scale well to large numbers of dynamic obstacles.

2.2 Sensor-based Navigation among Pedestrians

Sensor-based navigation algorithms are widely used to navigate a robot among pedestrians [12]. In [2], data from a radar and far infrared (IR) camera were synchronized and fused to track obstacles for collision avoidance. A significant problem in navigating among pedestrians is modeling their unknown intentions. [25] used the observed motions of humans to generate a motion probability grid to model pedestrian intentions. Some methods [16] learned obstacle motions from trajectories in a captured video or used laser scan data and Hidden Markov Models to estimate human trajectories [4].

Other techniques use a Partially Observable Markov Decision Process (POMDP) to model the uncertainties in the intentions of pedestrians. A POMDP-based planner was presented in [3] to estimate the pedestrians goals for autonomous driving which was later augmented with an ORCA-based pedestrian motion model [18]. The resulting POMDP planner runs in near real-time and is able to choose actions for the robot. Our approach is complimentary to these methods in that we model the navigation problem as a POMDP.

2.3 Learning-based Collision Avoidance

There is considerable work on using different learning methods for collision avoidance and navigation in dense environments.

2.3.1 ***Using single perception sensor:*** Several works have used data from a single perception sensor to train collision avoidance behaviors in a robot. A map-less navigation method using expert demonstrations in simulation was trained with a single 2-D lidar for static environments in [20]. In [24] a GAIL (Generative Adversarial Imitation Learning) strategy was trained using raw data from a depth camera over a pre-trained behavioral cloning policy to generate socially acceptable navigation through a crowd. However, the robot's navigation is limited by the depth camera's field of view (FOV) and works well only in sparse crowds and near-static environments.

An end-to-end visuomotor navigation system using CNNs that are trained directly with RGB images was developed in [14]. Similarly, [28] used a deep double-Q network (D3QN) to predict depth information from RGB images and used it for static obstacle avoidance in cluttered environments. [29] improved the generalization capability of deep reinforcement learning by including a visual goal in the policy of their actor-critic models. While these methods perform well for mostly static scenarios, they may not work well with dense crowds.

A decentralized sensor-level collision avoidance method that was trained with multi-robot Proximal Policy Optimization (PPO) [22] using a 2-D lidar in [17]. This approach was extended in [10] based on a hybrid control architecture, which switched between different policies based on the density of the obstacles in the environment. This approach works well in open spaces, but exhibits oscillatory

and jerky motions in dense scenarios since the 2-D lidar only senses the proximity data and fails to sense more complex interactions.

2.3.2 ***Using multiple sensors:*** [6] presents a decentralized agent-level collision avoidance method by utilizing a trained value network that models the cooperative behaviors in multi-agent systems. An LSTM-based strategy that uses observations of arbitrary numbers of neighboring agents during the training phase and makes no assumptions about obstacles' behavior rules is described in [7]. Other methods [5] explicitly model robot-human interactions in a crowd for robot navigation. These algorithms use a 2-D or 3-D lidar along with several RGB cameras for pedestrian classification and obstacle detection. However, these methods use a time parameter (Δt) for which obstacle motions are assumed to be linear. The value of Δt is important for their training to converge and their performances are susceptible to perception errors. In contrast, our training process is more robust and makes no such assumptions.

[19] presents an uncertainty-aware reinforcement learning method for collision avoidance that identifies novel scenarios and performs careful actions around the pedestrians. A method to solve both the freezing robot problem and loss of localization simultaneously by training an actor-critic model to learn localization recovery points in the environment is shown in [8]. This approach was also based on a single sensor and susceptible to jerky/oscillatory motion.

3 OVERVIEW

In this section, we introduce our multi-sensor based navigation problem and give an overview of our approach based on deep reinforcement learning. Unlike prior methods, our goal is to simultaneously use multiple sensors that sense the proximity data and various interactions and behaviors of dynamic agents and pedestrians and generate smooth trajectories.

3.1 Robot Navigation

We assume that the dynamics of the robot we train is bounded by non-holonomic constraints [1]. The robot at any point in time knows its environment only to the extent of its sensor observations, and no global knowledge of the state of the environment may be available. In addition, in accordance with real-world scenarios, the robot may not be able to access other hidden parameters of pedestrians and other dynamic agents, such as goals and states. At each time step t, the robot has access to an observation vector o^t which it uses to compute a collision-free action that drives it towards its goal from the current position by avoiding collisions with the static and dynamic obstacles. We assume that it is the sole responsibility of the robot to avoid collisions with pedestrians.

3.2 Multiple Sensors

Our approach is designed to exploit multiple sensors simultaneously. These include a lidar to measure the proximity to different obstacles and pedestrians. However, the lidar's raw data does not provide sufficient information to differentiate between animate and inanimate obstacles or sudden changes in the orientation of obstacles. As a result, it is difficult to infer whether the obstacles are moving towards or away from the robot. Therefore, we also use RGB or depth cameras for such observations. Furthermore, these cameras are able to capture the interactions between the obstacles and pedestrians in the scene.

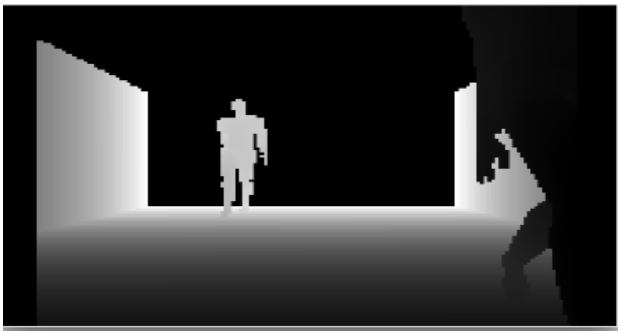

Figure 2: Contours of a pedestrian walking at a distance, and surrounding walls captured by the depth camera on our robot. Features such as the pedestrian's pose and motion can be extracted from such frames. Our CrowdSteer algorithm exploits these features for collision avoidance and generating smooth trajectories.

3.2.1 ***2-D lidar:*** Each scan/frame from a 2-D lidar consists of a list of distance values on the plane of sight of the lidar (See left scenario in Fig. 4). With its high accuracy, field of view (FOV), and low dimensional output data, the 2-D lidar allows us to detect clusters of closest points in the robot's surroundings.

3.2.2 ***Cameras:*** Depth images possess an additional dimension over and above a 2-D lidar. Therefore, features such as obstacle contours and changes in obstacles' poses are more prominently recorded even in low resolution images (Fig 2). This facilitates feature extraction to differentiate between moving and non-moving objects. For instance, a pedestrian changing direction away from the robot's trajectory would result in a contour with a lower area in the depth image. If we consider several consecutive frames from the camera, the approximate positions, orientations and velocities of all obstacles in the frame can be extracted. The same principles apply to RGB or grayscale images from an RGB camera. RGB cameras, in general, have a higher FOV than depth cameras and have an infinite sensing range similar to the human eye. Therefore, features corresponding to the obstacles can be captured in a single frame even when they are far from the robot.

The lidar and the camera sensors provide complimentary information about the environment. Using multiple consecutive frames of this combined information, helps our method learn that a change in features in the sensor data leads to a change in interaction with the obstacles and produces actions to accommodate it. In addition, a sensor's limited FOV can be overcome if another sensor with a high FOV is used in tandem with it. Using such combination of sensors is highly useful in indoor scenarios with a lot of occlusion. In our work, we assume that the data from the 2-D lidar and the camera are synchronized.

3.3 Problem Formulation

We formulate the navigation among pedestrians as a POMDP, which is solved using deep reinforcement learning. Formally, a POMDP is

modeled as a 6-tuple (S, A, P, R, Ω, O) [10], where the symbols represent the state space, action space, state-transition model, reward function, observation space and observation probability distribution given the system state, respectively. As stated earlier, our robot has access only to the observations, which can be sampled from the system's state space. Next, we describe our observation and action spaces.

Using data from the 2-D lidar along with a camera makes our observation space high-dimensional. In addition, since the individual sensor streams have different dimensions (1-D for lidar and 2-D for images), they cannot be processed together. Therefore, we split the robot's observation vector into four components, $\mathbf{o}^t = [\mathbf{o}^t_{lid}, \mathbf{o}^t_{cam}, \mathbf{o}^t_g, \mathbf{o}^t_v]$, where $\mathbf{o}^t_{lid}$ denotes raw noisy 2-D lidar measurements, $\mathbf{o}^t_{cam}$ denotes the raw image data from either a depth or an RGB camera, $\mathbf{o}^t_g$ refers to the relative goal location with respect to the robot, and $\mathbf{o}^t_v$ denotes the current velocity of the robot. $\mathbf{o}^t_{lid}$ is mathematically represented as:

$$\mathbf{o}^t_{lid} = \{l \in \mathbb{R}^{512} : 0 < l_i < 4\} \tag{1}$$

Where l represents a list of proximity values and l_i denotes the i^{th} element of the list. $\mathbf{o}^t_{cam}$ can be mathematically denoted as:

$$\mathbf{o}^t_{cam} = \{C \in \mathbb{R}^{150\times120} : 1.4 < C_{ij} < 5\} \tag{2}$$

The *action space* of the robot is composed of its linear and angular velocities $\mathbf{a}^t = [v^t, \omega^t]$. The objective of the navigation algorithm is to select an action $\mathbf{a}^t$ at each time instance, sampled from a trained policy π_θ as:

$$\mathbf{a}^t \sim \pi_\theta(\mathbf{a}^t|\mathbf{o}^t). \tag{3}$$

This action drives the robot towards its goal while avoiding collisions with pedestrians and static obstacles, until a new observation $\mathbf{o}^{t+1}$ is measured. We use the minimization of the mean arrival time of the robot to its goal position as the objective function to optimize the policy π_θ as:

$$\underset{\pi_\theta}{\operatorname{argmin}}\, \mathbb{E}[\frac{1}{N}\sum_{i=1}^{N} t^g_i|\pi_\theta]. \tag{4}$$

3.4 Reinforcement Learning Training and PPO

We use a policy gradient based [23] reinforcement learning method called Proximal Policy Optimization (PPO) [22] to solve the optimization problem in Equation 4. We adapt a policy gradient method since it directly models the strategy that generates actions, given the observations from the agents, and is more suitable for continuous action spaces such as ours. Compared with other policy gradient methods, PPO provides better stability during training by bounding the parameter (θ) updates to a trust region, i.e., it ensures that the updated policy does not diverge from the previous policy. At each training episode, a robot in simulation collects a batch of observations until a time T_{max} and the policy is then updated based on a loss function. PPO uses a surrogate loss function which is optimized using the Adam optimizer under the Kullback-Lieber(KL) divergence constraint which is given as:

$$\begin{aligned} L^{PPO}(\theta) = &\sum_{t=1}^{T_{max}} \frac{\pi_\theta(a^t_i|o^t_i)}{\pi_{old}(a^t_i|o^t_i)}\hat{A^t_i} - \beta KL[\pi_{old}|\pi_\theta] \\ &+ \xi max(0, KL[\pi_{old}|\pi_\theta] - 2KL_{target})^2 \end{aligned} \tag{5}$$

Where $\hat{A^t_i}$ is the advantage function, β and ξ are hyperparameters. The key issue is to design methods that take into account multiple sensor inputs and make sure that the training module converges fast and is able to handle all kind of observations.

4 OUR APPROACH: CROWDSTEER

In this section, we present our sensor-fusion based collision avoidance method that directly maps multiple sensor observations to a collision-free action. We describe the network that models our policy.

4.1 Network Architecture

Since the data from the lidar and camera have different dimensions (see Eqns 1, 2), they cannot be processed together. Therefore, our network (Fig.3) consists of four branches, each processing a single component of the observation o^t.

Branches 1 and 2 consist of multiple 1-D and 2-D convolutional layers to process the lidar and image observations, respectively, as the convolutional layers exhibit good performance in terms of extracting features from the input data [15]. Three consecutive lidar frames, each containing a list of proximity values are fed into two 1-D convolutional layers and a fully connected layer for processing. The first hidden layer has 32 filters and stride length = 2. The second hidden layer has 16 filters, and stride length = 2. Kernels, which capture the relationship among nearby elements in an array, of size 5 and 3 are used in the convolutional layers respectively.

Branch 2 is used to learn to detect motion from the three image frames, characterize the interaction based on obstacle motion and to move towards the free space detected in the frames. Three consecutive image frames are fed into a 2-D convolutional layer with 64 filters, kernel size = 5, and stride length = 2. The next layer uses 64 2-D convolutional filters, kernel size = 5 and stride length = 1. The output of this layer is passed on to 32 2-D convolutional layers of kernel size = 3, and stride length = 2. This is followed by two fully connected layers of sizes 512 and 256. Layer FC4 ensures that the output of branch 2 has the same dimensions as the output of branch 1. Layers FC1 and FC4 at the end of branches 1 and 2 also ensure that the data from the perception sensors have the most effect on the actions generated by the network. ReLU activation is applied to the outputs of all hidden layers in branches 1 and 2.

The output of branches 1 and 2 and the goal and current velocity observations are then correlated together by the fully connected layer FC2 with 128 rectifier units. In the output layer, a sigmoid activation is used to restrict the robot's linear velocity between (0.0, 1.0) m/s and a tanh function restricts the angular velocity between (-0.4, 0.4) rad/s. The output velocity is sampled from a Gaussian distribution which uses the mean and log standard deviation which were updated during training. The training of the network is end-to-end and all parts of the network are trained simultaneously.

4.2 Reward Function

We train the robot to reach its goal in the least possible time, while avoiding the obstacles. Therefore, the robot is rewarded for heading towards and reaching its goal, and penalized for moving too close or colliding with an obstacle. In addition, the robot is expected to avoid oscillatory velocities, follow smooth trajectories and reach

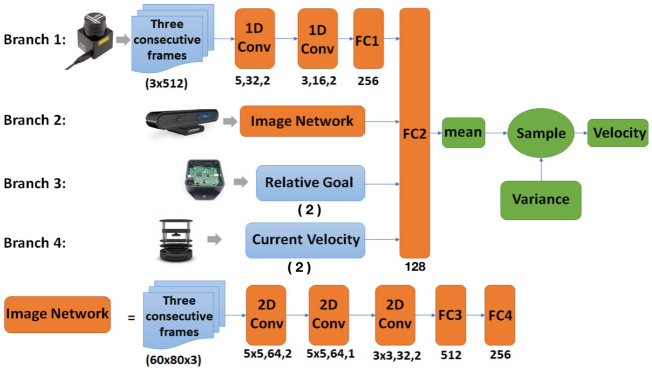

Figure 3: Architecture of our hybrid sensing network with four branches for different observations. The input layer is marked in blue, the hidden layers are marked in orange. Fully connected layers in the network are marked as FCn. The second branch extracts features from three consecutive image frames, which are fused with features extracted from three frames of the lidar in FC2 layer. The three values underneath each hidden layer denote the kernel size, number of filters, and stride length respectively.

intermediate waypoints before reaching the goal. Although the penalty for collision teaches the robot not to collide, it does not specifically result in the robot maintaining a safe distance from obstacles. This needs to be considered when the dynamic obstacles are pedestrians. Some previous algorithms [6, 17] train their model for multi-agent collision avoidance, which results in the robot not maintaining a safe distance from pedestrians. The intermediate waypoints provide the robot with a sense of direction and guide it towards its goal. Formally, the total reward collected by a robot i at time instant t can be given as:

$$r_i^t = (r_g)_i^t + (r_c)_i^t + (r_{osc})_i^t + (r_{safedist})_i^t \tag{6}$$

where the reward for reaching the goal $(r_g)_i^t$ or an intermediate waypoint is given as:

$$(r_g)_i^t = \begin{cases} r_{wp} & if\ ||\mathbf{p}_i^t - \mathbf{p}_{wp}|| < 0.1, \\ r_{goal} & if\ ||\mathbf{p}_i^t - \mathbf{g}_i|| < 0.1, \\ 2.5(||p_i^{t-1} - g_i|| - ||p_i^t - g_i||) & otherwise. \end{cases} \tag{7}$$

The collision penalty $(r_c)_i^t$ is given as:

$$(r_c)_i^t = \begin{cases} r_{collision} & if\ ||\mathbf{p}_i^t - \mathbf{p}_{obs}|| < 0.3, \\ 0 & otherwise. \end{cases} \tag{8}$$

The oscillatory behaviors (choosing sudden large angular velocities) are penalized as:

$$(r_{osc})_i^t = -0.1|\omega_i^t| \qquad if\ |\omega_i^t| > 0.3. \tag{9}$$

The penalty for moving too close to an obstacle is given by:

$$(r_{safedist})_i^t = -0.1||R_{S_{max}} - R_{min}^t||. \tag{10}$$

We set r_{wp} = 10, r_{goal} = 20, and $r_{collision}$ = -20 in our formulation. Critical behaviors such as collision avoidance and goal reaching have a higher priority in terms of the overall reward collected by the robot, while choosing smoother velocities and maintaining a safe distance from obstacles contribute to the reward with a slightly lower priority.

4.3 Training Scenarios

A major challenge in learning based methods is to close the sim-to-reality gap that arises when using a policy trained using simulated environments and sensor observations is evaluated in the real world. Additionally, when using multiple sensors, especially cameras, the simulation should contain as many real world features or characteristics as possible for the training to generalize well. One of our goals is to take advantage of the camera's ability to observe such complex features (Fig. 2). To address these issues simultaneously, we use high-fidelity 3-D environments that replicate real-world open spaces, and indoor scenarios with realistic moving pedestrians, and occluded environments.

The policy training is carried out in multiple stages, starting with a low-complexity static scenario, to dynamic scenarios with pedestrians. The simple scenarios initialize the policy π_θ with capabilities such as static collision avoidance and goal-reaching, while dynamic collision avoidance capability is learned in complex scenarios with moving pedestrians. The different type of environments used in training are shown in (Fig.4). These include:

Static Scenario: We use two independent robots that must start from fixed initial locations and head towards fixed goals straight in front of them while avoiding a wall. A single policy is shared between the two robots so that both left and right turning maneuvers are learned simultaneously. Using this initialized policy, we then provide random initial locations and random goals to the two robots for the goal-reaching and static collision avoidance capabilities to generalize well.

Random Static and Dynamic Obstacles: In this scenario, a single robot has to move towards a random goal in the presence of static human models and randomly moving pedestrians. We ensure that there is more than enough space through which the robot can navigate. To generate pedestrian trajectories, we assign several waypoints or goals in the environment for the pedestrian models to move to at different time instants. The simulated pedestrians move at near human walking speeds and tend to mimic natural human walking motions.

Scenario with occluded obstacles: The saved model from the previous scenario is next trained on a complex corridor scenario, which requires the robot to perform several sharp turns and avoid pedestrians who can only be observed in close proximity. This trains the robot to perform quick maneuvers in real-world scenarios where occluded pedestrians walk towards the robot suddenlyr.

Sim-to-real transfer: Both the scenarios with dynamic obstacles make the sim-to-real transfer and generalization easier, as the depth camera observes noisy 3-D real-life like data such as pedestrian contours and walls, which is fused with more accurate lidar data. Such fusion was not possible in previous methods [10, 17], as they were trained in a 2.5D simulator. However, in the corridor scenario, there is a danger of the policy overfitting to follow walls and moving to goals which are straight ahead of the robot instead of generalizing to all scenarios. To prevent this, we also parallelly run

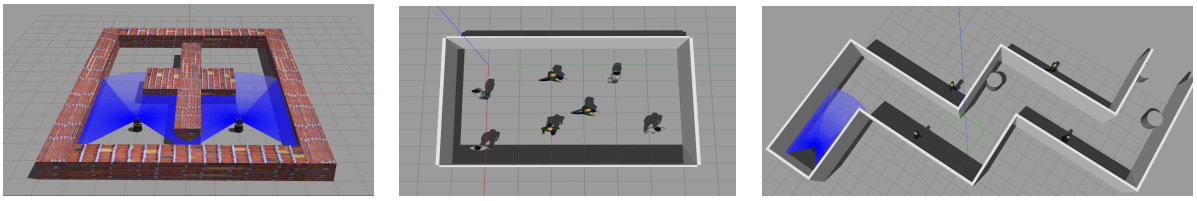

Figure 4: Left to right: The different training scenarios used by our algorithm from simplest to complex. Left: Static Scenario. Middle: Scenario with static and random dynamic obstacles. Right: Scenario with occlusions and sharp turns.

robots in the simple static scenarios with random goals alongside the corridor scenario.

5 RESULTS AND EVALUATIONS

In this section, we describe our implementation and highlight its performance in different scenarios. We also compare our navigation algorithm with prior methods and perform ablation studies to highlight the benefits of implicit multi-sensor fusion and our reward function.

5.1 Implementation

We train our model in simulations that were created using ROS Kinetic and Gazebo 8.6 on a workstation with an Intel Xeon 3.6GHz processor and an Nvidia GeForce RTX 2080Ti GPU. We use Tensorflow, Keras and Tensorlayer for implementing our network. We use models of the Hokuyo 2-D lidar and the Orbbec Astra depth camera in Gazebo to simulate sensor data during training and evaluation. The Hokuyo lidar has a range of 4 meters, FOV of 240° and provides 512 range values per scan. The Astra camera has a minimum and maximum sensing range of 1.4 meters and 5 meters respectively. We use images of size 150×120 with added Gaussian noise $\mathcal{N}(0, 0.2)$ as inputs to our network training. We mount these sensors on a Turtlebot 2 and a Clearpath Jackal robot to test our model in real-world scenarios such as crowded corridors and occluded scenes.

5.2 Training Convergence and Data Efficiency

The convergence of our reward function versus the number of iterations for different training scenarios is shown in Fig.5. The training in all scenarios starts to converge around 100 iterations, and stabilizes around 200 iterations and the total process completes in six days. This increase in training time when compared with previous methods is due to the high dimensionality of the 3-D depth data used during training. Previous methods do not perform any kind of implicit sensor fusion to detect and avoid obstacles. However, the training time does not affect our run-time performance, which can be observed from our real-world tests.

5.3 Testing scenario

We consider five different test scenarios that have narrower or different sections, as compared to the synthetic datasets used during our training phase. This demands tight maneuvers from the robot to reach the goal. These testing scenarios are more challenging than existing simulation benchmarks and help to better test CrowdSteer's sim-to-real, and generalization capabilities. We define *Least Passage Space* (LPS) as the minimum space available to the robot

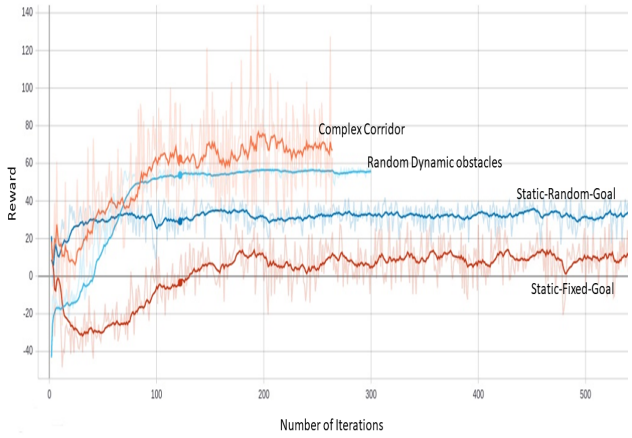

Figure 5: Convergence of our reward function vs the number of iterations for different training scenarios. Training in all scenarios converges within 200 iterations.

in-between obstacles when moving towards the goal. The scenarios we consider are:

1. **Narrow-Static**: Scenario with only static obstacles where the LPS is < 0.7 meters.
2. **Narrow-ped**: Scenario with 8 pedestrians walking in the opposite direction to the robot's motion in a narrow corridor, with an LPS of < 1.5 meters.
3. **Occluded-ped**: Scenario with sharp turns with static obstacles and pedestrians that are occluded by walls. The LPS is < 1 meter.
4. **Dense-Ped**: Scenario with 18 pedestrians in a corridor of width 6 meters. Pedestrians may walk together as pairs, which require a robot to make sharp turns in the presence of multiple dynamic obstacles. The LPS is < 1 meter.
5. **Circle**: To test the generalization of our method for multi-robot collision avoidance, we make 4 robots move towards antipodal positions on a circle. The trajectories of the 4 robots is shown in Fig. 6.

5.4 Performance Benchmarks and Metrics

We compare the benefits of our hybrid sensing method with three prior algorithms: (i) DWA [11] which uses a lidar / proximity sensors to detect obstacles along with a global planner which requires a map of the environment.(ii) An implementation that uses a single depth camera that was trained using PPO and our reward functions;

Metrics	Method	Narrow-Static	Narrow-Ped	Occluded-Ped	Dense-Ped
Success Rate	DWA	1	0.1	0.5	0.0
	Depth Camera	0.85	0.55	0.2	0.2
	Fan et al.	1	0.0	0.0	0.0
	CrowdSteer	1	1	0.9	0.67
Avg Trajectory Length	DWA	6.33	14.86	27.2	7.46
	Depth Camera	6.18	16.30	25.7	14.95
	Fan et al.	6.86	6.16	13.63	9.17
	CrowdSteer	6.44	15.51	27.18	16.58
Mean Time	DWA	20.9	44.1	60.4	25.72
	Depth Camera	58.7	43.6	78.20	90.73
	Fan et al.	106.93	13.76	28.025	38.05
	CrowdSteer	34.04	41.48	70.54	64.9
Avg Velocity	DWA	0.30	0.34	0.45	0.29
	Depth Camera	0.11	0.37	0.33	0.16
	Fan et al.	0.06	0.44	0.48	0.23
	CrowdSteer	0.20	0.37	0.39	0.26

Table 1: We compare the relative performance of CrowdSteer that uses multiple sensors (depth camera + lidar) with other learning methods that use a single sensor, and a traditional method (DWA) in challenging scenarios. The values in gray are until a collision or oscillation occurred. These numbers clearly highlight the benefit of our novel deep reinforcement learning algorithm (CrowdSteer) that uses multiple sensors over prior methods.

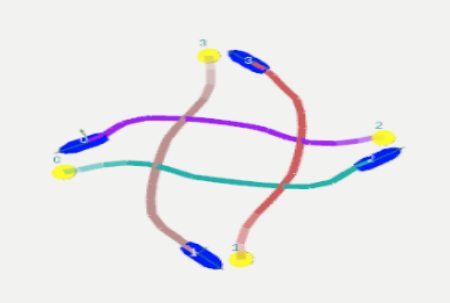

Figure 6: Testing the generalization of our training in a scenario with four agents (in blue) moving towards antipodal points on a circle (in yellow). Each robot takes full responsibility for collision avoidance, which leads to making wide maneuvers during collision avoidance.

(iii) Fan et al. [17], the current state-of-the-art learning-based collision avoidance method for dense crowd navigation, which uses a single 2-D lidar for sensing. Its real-world implementation is shown in [9]. Fan et al. [17] assumes that the pedestrians in dense scenarios would cooperate with the robot to avoid collisions. On the other hand, we assume that the robot takes the full responsibility to avoid collisions, though the pedestrians may or may not be cooperative. We use the following metrics to evaluate the performance of different navigation algorithms:

- **Success Rate** - The number of times that the robot reached its goal without colliding with an obstacle over the total number of attempts.
- **Average Trajectory Length** - The trajectory length traversed by the robot until the goal is reached, calculated as the sum of linear movement segments over small time intervals over the total number of attempts. In cases where the robot never reached the goal, we report the trajectory length *until it collided or started oscillating* indefinitely.
- **Mean Time** - Average time taken to reach the goal over all attempts. If the goal is never reached in all attempts, we report the mean time until a collision or oscillation.
- **Average velocity** - The average velocity of the robot until a collision occurs or the goal is reached over all attempts.

5.5 Analysis and Comparison

The results of our experiments and our ablation study to check the benefits of using multiple sensors (both lidar and depth camera) versus using one sensor (only depth camera) are shown in Table 1.
Comparison with DWA: DWA [12] and CrowdSteer succeed in reaching the goal 100% of the times in static scenarios. However, as the number and density of dynamic agents in the environment increase, DWA's success rate drops considerably when compared to CrowdSteer. This is due to the re-planning time it takes in the presence of dynamic obstacles or pedestrian. DWA performs well in the occluded-ped scenario due to its use of global map, which eliminates many occlusions. In cases where DWA reached the goal, it has more optimal trajectories and mean time due to the optimal global planner. CrowdSteer manages to have comparable performances without any global knowledge or planner.

Comparison with Fan et al. [17]: The robot that uses Fan et al.'s method works well in the static scenario. However, while it manages to avoid collisions, it either gets stuck or starts oscillating indefinitely in scenarios with non-cooperative dynamic obstacles and occluded spaces. The values for the trajectory length, mean time and velocity reported when Fan et al's method always failed, are *until a collision or oscillation* to give a sense of how much the

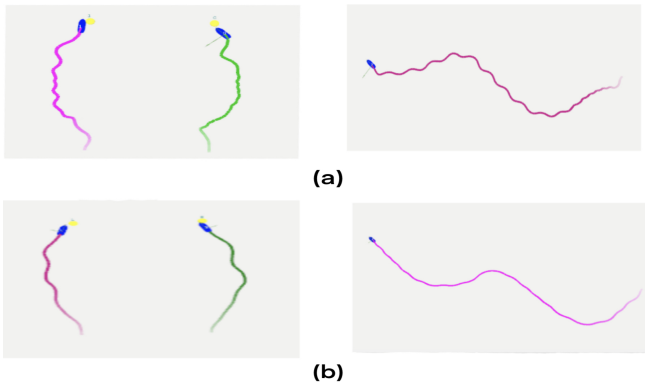

Figure 7: a) Trajectories generated by Fan et al.'s method in the Narrow-static (left) and Occluded-ped scenarios (right). b) The trajectories generated by CrowdSteer for the same scenarios.

robot traversed towards the goal before failing. These results are due to Fan et al.'s training, which is based on multi-agent collision avoidance where all the agents share the responsibility of avoiding collisions. This assumption may not always hold in dense crowds. CrowdSteer therefore manages to outperform Fan et al's method in all scenarios.

Apart from having a much better success rate, CrowdSteer has similar performance as prior methods in terms of average velocities, better trajectory lengths and mean time to Fan et al. which in turn had better success rates, 41.6% lower time taken to reach the goal, and 14% higher average speeds than works like NH-ORCA [1]. Therefore, our method outperforms the current state-of-the-art in dense and occluded scenarios.

Comparing Smoothness: We also compare the smoothness of the trajectories computed by CrowdSteer and Fan et al. [17] method (Figure 7) in the Occluded-Ped, and Narrow-static scenarios. CrowdSteer is trained to maintain a safe distance from obstacles and avoids oscillations. This results in significant improvement in the smoothness of the trajectories.

Ablation study for multi-sensor fusion: We compare the effects of using fused data from two sensors (CrowdSteer) versus a model which uses one sensor (Depth Camera) trained with our training scenarios and reward function in different scenarios. Using only a depth camera limits the robot's sensing range and field of view. Due to this, the depth camera model does not have a 100% success rate in any scenario. This also reflects in the success rate drop in the Occluded-Ped and Dense-Ped benchmarks. However, it still manages to succeed in the dynamic testing scenarios due to our training scenarios and network architecture. Our CrowdSteer algorithm takes advantage of both the high accuracy of the lidar and the complex features extracted from the depth images and demonstrates significantly better success rates, lower mean time, and higher average velocities.

Ablation study of smoothness: We study the effect of our reward function on the robot's trajectory smoothness. We trained two policies, one including the penalty for oscillations in the reward function (Eqn. 9), and the other without it. The average number of oscillations in the robot's trajectory are summarized in Table 2. The models are evaluated in two scenarios: (i) Scenario without any obstacles, and the robot moves in a straight line for 10 meters towards the goal, (ii) Scenario where the robot must maneuver to avoid a static obstacle before reaching the goal. Empty/sparse scenarios are used so that turns during dynamic obstacle avoidance does not affect the number of oscillations. We observe that there is a significant reduction in the number of oscillations when the penalty is included.

Scenario	Without Penalty	With Penalty
Empty world	9.8	2
Static obstacle	9	2

Table 2: The average number of oscillations in two scenarios for two models trained without and with the oscillations penalty term. We see a significant reduction in the number of oscillations in our model that is trained with the penalty.

Real-world scenarios: We use CrowdSteer to navigate a Turtlebot 2 and a Clearpath Jackal robot in crowds with varying densities (1-3 person/m^2), as shown in the video. The robots face high randomness in terms of the direction and velocities of pedestrians, which was not encountered during training. We compare the motion of CrowdSteer with Fan et al.[17] method in similar scenarios. Compared to Fan et al., we observe that CrowdSteer has smoother trajectories in both robots and avoids all collisions with the obstacles. On the other hand, Fan et al.[17] method to compute collision-free velocities are highly oscillatory. In occluded spaces such as corridors, our CrowdSteer algorithm was able to avoid sudden obstacles which appear in places such as T and L junctions. These tests show the advantages of implicit sensor fusion and our training scenarios with occlusions. Our real-world tests also demonstrate our method's strong sim-to-real and generalization capabilities.

Failure Cases: CrowdSteer may not work well certain cases. In highly spacious regions, CrowdSteer could exhibit oscillatory behaviors. It might also fail for acute angled turns and in environments with reflective or transparent surfaces, and high interference from infrared light in the surroundings. In crowds with density > 4 people/m^2 or scenarios with very minimal or narrow space for navigation, the robot may not find a collision-free path.

6 CONCLUSION, LIMITATIONS AND FUTURE WORK

We present a novel sensor-based navigation algorithm, CrowdSteer, that simultaneously uses multiple sensors such as 2-D lidars and cameras. Our approach is designed for dense scenarios with pedestrians and makes no assumption about their motion. In practice, our approach works well in complex, occluded scenarios and results in smoother trajectories. Our approach has some limitations and failure cases. It is susceptible to freezing robot problem in very dense settings and the computed trajectories are not globally optimal. Furthermore, the current sensors may not accurately handle glass or non-planar surfaces. There are many avenues for future work. In addition to overcoming these limitations, we need to evaluate its performance in other scenarios and outdoor settings. We may also take into account the dynamics constraints of the robots in terms of trajectory computation.

REFERENCES

[1] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul A. Beardsley, and Roland Siegwart. 2010. Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots. In *DARS*.

[2] A. Amditis, A. Polychronopoulos, I. Karaseitanidis, G. Katsoulis, and E. Bekiaris. 2002. Multiple sensor collision avoidance system for automotive applications using an IMM approach for obstacle tracking. In *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002. (IEEE Cat.No.02EX5997)*, Vol. 2. 812–817 vol.2. DOI: http://dx.doi.org/10.1109/ICIF.2002.1020890

[3] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee. 2015. Intention-aware online POMDP planning for autonomous driving in a crowd. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 454–460. DOI: http://dx.doi.org/10.1109/ICRA.2015.7139219

[4] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. 2005. Learning Motion Patterns of People for Compliant Robot Motion. *The International Journal of Robotics Research* 24, 1 (2005), 31–48. DOI: http://dx.doi.org/10.1177/0278364904048962 arXiv:https://doi.org/10.1177/0278364904048962

[5] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. 2019. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *ICRA*. IEEE, 6015–6022.

[6] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. 2017. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *ICRA*. IEEE, 285–292.

[7] Michael Everett, Yu Fan Chen, and Jonathan P How. 2018. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *IROS*. IEEE, 3052–3059.

[8] Tingxiang Fan, Xinjing Cheng, Jia Pan, Pinxin Long, Wenxi Liu, Ruigang Yang, and Dinesh Manocha. 2019. Getting Robots Unfrozen and Unlost in Dense Pedestrian Crowds. *IEEE Robotics and Automation Letters* 4, 2 (2019), 1178–1185.

[9] Tingxiang Fan, Xinjing Cheng, Jia Pan, Dinesh Manocha, and Ruigang Yang. 2018. CrowdMove: Autonomous Mapless Navigation in Crowded Scenarios. *arXiv e-prints* (Jul 2018), arXiv:1807.07870. arXiv:cs.RO/1807.07870

[10] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. 2018. Fully Distributed Multi-Robot Collision Avoidance via Deep Reinforcement Learning for Safe and Efficient Navigation in Complex Scenarios. *arXiv e-prints* (Aug 2018), arXiv:1808.03841. arXiv:cs.RO/1808.03841

[11] D. Fox, W. Burgard, and S. Thrun. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine* 4, 1 (March 1997), 23–33. DOI: http://dx.doi.org/10.1109/100.580977

[12] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers. 1998. A hybrid collision avoidance method for mobile robots. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, Vol. 2. 1238–1243 vol.2. DOI: http://dx.doi.org/10.1109/ROBOT.1998.677270

[13] Dirk Helbing and Peter Molnar. 1995. Social force model for pedestrian dynamics. *Physical review E* 51, 5 (1995), 4282.

[14] Y. Kim, J. Jang, and S. Yun. 2018. End-to-end deep learning for autonomous navigation of mobile robot. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*. 1–6. DOI: http://dx.doi.org/10.1109/ICCE.2018.8326229

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (May 2017), 84–90. DOI: http://dx.doi.org/10.1145/3065386

[16] Frédéric Large, Dizan Alejandro Vasquez Govea, Thierry Fraichard, and Christian Laugier. 2004. Avoiding Cars and Pedestrians using V-Obstacles and Motion Prediction. In *Proc. of the IEEE Intelligent Vehicle Symp.* Pisa (IT), France. https://hal.inria.fr/inria-00182054 voir basilic : http://emotion.inrialpes.fr/bibemotion/2004/LVFL04/ note: Poster session address: Pisa (IT).

[17] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. 2017. Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning. *arXiv e-prints* (Sep 2017), arXiv:1709.10082. arXiv:cs.RO/1709.10082

[18] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha. 2018. PORCA: Modeling and Planning for Autonomous Driving Among Many Pedestrians. *IEEE Robotics and Automation Letters* 3, 4 (Oct 2018), 3418–3425. DOI: http://dx.doi.org/10.1109/LRA.2018.2852793

[19] B. Lötjens, M. Everett, and J. P. How. 2019. Safe Reinforcement Learning With Model Uncertainty Estimates. In *2019 International Conference on Robotics and Automation (ICRA)*. 8662–8668. DOI: http://dx.doi.org/10.1109/ICRA.2019.8793611

[20] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. 2016. From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots. *arXiv e-prints* (Sep 2016), arXiv:1609.07910. arXiv:cs.RO/1609.07910

[21] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena. 2017. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 1527–1533. DOI: http://dx.doi.org/10.1109/ICRA.2017.7989182

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv e-prints* (Jul 2017), arXiv:1707.06347. arXiv:cs.LG/1707.06347

[23] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99)*. MIT Press, Cambridge, MA, USA, 1057–1063. http://dl.acm.org/citation.cfm?id=3009657.3009806

[24] L. Tai, J. Zhang, M. Liu, and W. Burgard. 2018. Socially Compliant Navigation Through Raw Depth Inputs with Generative Adversarial Imitation Learning. In *ICRA*. 1111–1117. DOI: http://dx.doi.org/10.1109/ICRA.2018.8460968

[25] S. Thompson, T. Horiuchi, and S. Kagami. 2009. A probabilistic model of human motion and navigation intent for mobile robot path planning. In *2009 4th International Conference on Autonomous Robots and Agents*. 663–668. DOI: http://dx.doi.org/10.1109/ICARA.2000.4803931

[26] R. B. Tilove. 1990. Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Proceedings., IEEE International Conference on Robotics and Automation*. 566–571 vol.1. DOI: http://dx.doi.org/10.1109/ROBOT.1990.126041

[27] Jur P. van den Berg, Stephen J. Guy, M. Chiao Lin, and Dinesh Manocha. 2009. Reciprocal n-Body Collision Avoidance. In *ISRR*.

[28] Linhai Xie, Sen Wang, Andrew Markham, and Niki Trigoni. 2017. Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning. *arXiv e-prints* (Jun 2017), arXiv:1706.09829. arXiv:cs.RO/1706.09829

[29] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2016. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *arXiv e-prints* (Sep 2016), arXiv:1609.05143. arXiv:cs.CV/1609.05143