Memory-Based Deep Reinforcement Learning for Obstacle Avoidance in UAV With Limited Environment Knowledge

Abhik Singla, Sindhu Padakandla, and Shalabh Bhatnagar

***Abstract*— This paper presents our method for enabling a UAV quadrotor, equipped with a monocular camera, to autonomously avoid collisions with obstacles in unstructured and unknown indoor environments. When compared to obstacle avoidance in ground vehicular robots, UAV navigation brings in additional challenges because the UAV motion is no more constrained to a well-defined indoor ground or street environment. Unlike ground vehicular robots, a UAV has to navigate across more types of obstacles - for e.g., objects like decorative items, furnishings, ceiling fans, sign-boards, tree branches, etc., are also potential obstacles for a UAV. Thus, methods of obstacle avoidance developed for ground robots are clearly inadequate for UAV navigation. Current control methods using monocular images for UAV obstacle avoidance are heavily dependent on environment information. These controllers do not fully retain and utilize the extensively available information about the ambient environment for decision making. We propose a deep reinforcement learning based method for UAV obstacle avoidance (OA) which is capable of doing exactly the same. The crucial idea in our method is the concept of partial observability and how UAVs can retain relevant information about the environment structure to make better future navigation decisions. Our OA technique uses recurrent neural networks with temporal attention and provides better results compared to prior works in terms of distance covered without collisions. In addition, our technique has a high inference rate and reduces power wastage as it minimizes oscillatory motion of UAV.**

***Index Terms*— Unmanned aerial vehicle (UAV) obstacle avoidance (OA), deep reinforcement learning (DRL), partial observability, deep Q-networks (DQN).**

I. Introduction

UNMANNED aerial vehicles (UAVs) or "drones" are cyber-physical systems that can be operated either by remote control or autonomously using onboard computers. Ranging from crop [1] and infrastructure monitoring [2], rescue operations and disaster management [3], to more popular uses like goods delivery and filming [4], [5], UAVs are increasingly finding application in diverse scenarios. Owing to their small size UAVs can be utilized to penetrate constricted spaces, which may possibly be beyond the reach of humans. However, while navigating through constricted spaces, UAVs face the challenge of avoiding obstacles. Avoiding obstacles is a difficult task, because in constricted spaces, the obstacles might be so positioned that avoiding them requires delicate and dexterous movements. To be able to carry out these dexterous moves, the UAV needs to perceive the distance between itself and the obstacles along with other visual cues such as their shape and height. This crucial visual information enables a UAV to infer traversable spaces and obstacles (see Fig. 1 for an illustration).

Fig. 1. A UAV encountering stationary as well as moving obstacles in an indoor environment. Here, the walking human being is a moving obstacle, whose direction and future intent of motion cannot be predicted.

Classical approaches for inferring visual geometry include techniques like Simultaneous Localization and Mapping (SLAM) and Structure from Motion (SfM). These techniques use measurements from sensors like Kinect [6], Light Detection and Ranging (LIDAR), Sound Navigation and Ranging (SONAR), optical flow, stereo and monocular cameras for computation. SLAM algorithms utilize measurements from a single sensor [7] or a combination of sensors [8] to build and update a map of the environment surrounding the UAV while simultaneously using the same to estimate the UAV's position. The SfM approaches use measurements from sensors like optical flow [9] and/or a moving monocular camera [10] to determine depth map and the 3D structure. SLAM and SfM approaches require the UAV to compute a path and then follow the computed path. The UAV needs to repetitively hover, compute the depth map and then find a suitable path. Thus, path planning on the fly

Manuscript received June 4, 2019; revised September 30, 2019; accepted November 18, 2019. Date of publication November 28, 2019; date of current version December 24, 2020. This work was supported in part by grants from the Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science, as well as the Department of Science and Technology, through the ICPS Program. The Associate Editor for this article was D. Sun. *(Corresponding author: Sindhu Padakandla.)*

A. Singla is with the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bangalore 560012, India (e-mail: abhiksingla@iisc.ac.in).

S. Padakandla and S. Bhatnagar are with the Department of Computer Science and Automation, Indian Institute of Science, Bengaluru 560012, India (e-mail: sindhupr@iisc.ac.in; shalabh@iisc.ac.in).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/TITS.2019.2954952

is not easy in SLAM and SfM approaches. This also means that SLAM and SfM approaches cannot be used for real-time obstacle avoidance based on the visual information gathered about the surroundings. Reference [11] proposes a SLAM technique which computes a path on the fly. However, such an enhancement does not avoid dynamic and non-stationary obstacles whose movements cannot be predicted. Another disadvantage of using SLAM and SfM methods is that these do not detect untextured walls. Untextured walls normally arise in indoor environments and hence being able to distinguish textures on walls is crucial to obstacle avoidance.

Kinect, LIDAR, SONAR, optical flow, and stereo camera sensors are widely used for depth estimation (see [12], [13]) and hence these can be potentially used for obstacle avoidance as well without resorting to computation-intensive approaches like SLAM and SfM. However, these sophisticated sensors are expensive and add unnecessary burden to the UAV in terms of weight as well as consumption of power. Moreover, optical flow and stereo camera are not suited for long-range obstacle avoidance. Other sensors, like for example, the monocular camera, is essential for every UAV application, as it gives visual information. The monocular camera is a low-cost sensor which provides RGB images of the UAV's ambient environment. In comparison to the heavy-weight sensors mentioned earlier, a monocular camera is light-weight. The question then is whether we can use a monocular camera for depth estimation as well and plausibly for obstacle avoidance.

Extracting the range information (i.e., distance between the sensor and the various objects in front of the sensor) from the monocular RGB images is a challenging problem, simply because the camera captures only the 2-D information of the surrounding environment. Some recent works [14]–[23] address the issue of depth prediction using monocular camera RGB images by leveraging deep learning techniques. Supervised and semi-supervised learning approaches ([14]–[18]) collect huge amounts of data consisting of the monocular images and the corresponding depth maps to train a deep learning model. Such models are based on convolutional neural networks (CNN) or their variants (residual networks [14]). Given a single image, the deep network outputs the predicted depth map from the monocular image. The proposed approaches in [14]–[18] however do not tackle the vital problem of UAV obstacle avoidance and navigation, which is the problem that we are interested in this paper.

Varied obstacle avoidance techniques in conjunction with depth prediction are proposed in [19]–[23]. Reference [19] proposes a behavior arbitration scheme to obtain the yaw and pitch angles for the UAV to avoid an obstacle and for navigation in general. Trajectory planning using obstacle bounding boxes and depth estimation is explored in [20]. This work designs a CNN architecture that jointly estimates depth and obstacle bounding boxes. The extracted information is utilized in the RRT-Connect planner to plan trajectories between start and end points. Reference [21] proposes two different CNN architectures - one for depth and surface normal estimation and the other for trajectory prediction. Both the CNNs use a 3D cost function for training and evaluation. Reference [22] follows an unconventional approach, wherein the authors collect a dataset of UAV crashes. This dataset is labeled and then input to a CNN model. Given an image obtained from the monocular camera, the network predicts how the UAV should move in the next instant to avoid a crash. UAV navigation in the presence of obstacles is inherently a sequential decision making problem under uncertainty. This is because an action taken at an instant affects the path of the UAV in the future instants too. Hence, it is appropriate to design obstacle avoidance in UAV as a model-free reinforcement learning (RL) problem (since the system model is typically unknown). CAD2RL [23] proposes a Deep RL (DRL) method for obstacle avoidance in indoor flight. This work trains a UAV for navigation using simulated 3D hallway environments. For this, a large number of 3D hallway environment images with different lighting, wall textures, furniture placement are generated and a deep Q-network learns UAV movement policy on these images. However, this work requires substantial amount of data concerning the images of hallway environments and is not efficient. Moreover, the method proposed in [23] is not intuitive. It does not attempt to mimic how humans learn to avoid obstacles. The basic information which helps the human brain to navigate is the depth information (owing to the binocular vision) and not the RGB information.

Taking cue from how humans learn to avoid obstacles, we propose a DRL method which enables the UAV controller to collect and store relevant observations gathered over time and utilize the stored observations to avoid obstacles dexterously. We are motivated from how humans decide what to do next given a scenario. Humans have *limited* or *partial access* to the environment, but still are able to solve challenging problems in daily lives. All this is possible, because human brain has memory which is key to summarizing and storing relevant information for tackling problems. This memory is capable of effectively storing and recalling relevant information gathered over time in order to take the next suitable decision in every scenario. UAV obstacle avoidance also presents a similar problem of *partial observability* which requires a notion of memory. For example, while navigating, a UAV may fly towards a corner. When it is approaching the corner, the depth map might indicate more space in the front when compared to the sides. The lack of temporal information coupled with limited field of vision of the monocular camera makes the UAV to move ahead towards the corner and crash onto the wall. Such scenarios are very common in UAV navigation and hence require a controller which can utilize the relevant past information. Our aim is to design a UAV control algorithm which has the capability to combine information obtained over a period of time in order to make better navigation and obstacle avoidance decisions.

We propose a DRL method based on recurrent neural network (RNN) architecture and *Temporal Attention*. This method enables the UAV controller to collect, store relevant observations gathered over time and use them to make better obstacle avoidance (OA) decisions.

A. Organization of the Paper

The next section describes the method we have developed for UAV obstacle avoidance. Section III gives the details of

experimental settings and the simulation environments used for highlighting the performance of our method. Sections IV and V describe the results on a number of simulation settings and also bring out the advantages as well as limitations of our approach. Section VI concludes the paper and points out future improvements for our method.

II. The Method

The objective of our work is to find a suitable policy (a sequence of actions given states of the environment) for UAV movement that avoids obstacles (both stationary and mobile). We propose a general method which can find such suitable policies. Our method can be integrated with a high-level planner which takes as input an overall path objective with a start and a goal position.

A. Problem Definition

In order to safely navigate without colliding against obstacles in an indoor or outdoor environment, the UAV needs to be aware of the state s of the environment. The state of the environment is a tuple of properties of the environment which characterize it and aid the UAV in navigation. Once the state s is known, the UAV selects an appropriate action a. The action the UAV chooses affects the visual information available to the UAV. In the obstacle avoidance problem, this means the UAV chooses to move in some particular direction leading to a change in its position, orientation, and visual feedback. The UAV gets to observe more obstacles or perhaps more free space in front depending on this change in position and/or orientation. As noted in Section I, the UAV needs to choose an action depending on the state at every instant t when it navigates through the environment. Further, each action taken affects future states and hence future decisions of the UAV. Based on the action taken, the realization of the next state is probabilistic implying that the navigation by avoiding obstacles is a sequential decision making problem in the face of uncertainty.

Prior works [23], [24] assume that the monocular image of the environment is a good indicator of the state of the environment. However, since the UAV monocular camera has a limited field of vision, we believe that the UAV controller cannot infer the full state of the environment using the monocular RGB image. Moreover, the RGB images do not provide the depth information. Thus, the UAV controller has only an estimate of the state and it needs to decide on the next action based on the RGB image input. With this reasoning, we propose a partially observable Markov decision process (POMDP) model for the UAV OA task.

B. Model

We propose a POMDP model $\langle S, A, P, R, \Omega, \mathcal{O}, \gamma \rangle$ for the obstacle avoidance problem. Here S is the set of states of the environment, referred to as "state space", while A is the set of *feasible* actions and referred to as the "action space". P is the transition probability function that models the evolution of states based on actions chosen and is defined as $P : S \times A \times S \rightarrow [0, 1]$. R is the *reinforcement* or the *reward* function defined as $R : S \times A \rightarrow \mathbb{R}$. The reward function serves the role of providing a *feedback* signal to the UAV for the action chosen. For instance, in a state s, if the UAV selects an action a which steers it away from an obstacle, the reward for that state-action pair (s, a) is positive, implying that the action a is beneficial in the state s, while picking an action which results in collision will naturally yield a negative reward. Ω is the set of observations and an observation $o \in \Omega$ is an estimate of the true state s. $\mathcal{O} : S \times A \times \Omega \rightarrow [0, 1]$ is a conditional probability distribution over Ω, that corresponds to a distribution on observations, given the state-action tuples and is explained below. $\gamma \in (0, 1)$ is the discount factor. At each time t, the environment state is $s_t \in S$. The UAV takes an action $a_t \in A$ which causes the environment to transition to state s_{t+1} with probability $P(s_{t+1}|s_t, a_t)$. Based on this transition, the UAV receives an observation $o_t \in \Omega$ which depends on s_{t+1} with probability $\mathcal{O}(o_t|s_{t+1}, a_t)$. The aim is to solve the obstacle avoidance problem, which translates to the task of finding an optimal *policy* $\pi^* : \Omega \rightarrow A$. By determining an optimal policy, the UAV controller is able to select an action at each time step t that maximizes the expected sum of discounted rewards, which is denoted as $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$.

We need to define the sets S, Ω, A and the functions $R, P, \mathcal{O}$ in order to find an optimal policy. The state space S for our model is the information representing the ambient environment and which is useful in the OA task. Since it is impossible to summarize all useful information about the ambient environment for the OA task, we can only define the state space S as the collection of RGB-D images of the ambient environment. RGB-D images are RGB images with the associated depth information. These images contain the distance of each object from the UAV along with the RGB information. At every decision instant, the monocular camera mounted on the UAV has a limited field of view and provides only the RGB image and cannot provide the depth information. Thus, the complete state information is not available for decision making. So, we need to design a method which takes as input the monocular RGB image with limited information and outputs a valid action for UAV movement.

The input to our model is the monocular RGB image which is of size $84 \times 84 \times 3$ pixels, without any depth information. Our model extracts the depth map from the RGB image which is an observation $o \in \Omega$ for the UAV controller. The size of the depth image is 84×84 pixels. While ascertaining the shape, height, and distance of the obstacles, the depth map plays an important role, since the depth map indicates the distance between the objects and UAV. Hence we define the depth map as an observation. Thus, the set of observations Ω is all possible depth maps obtained from the RGB images (after extraction). Given an observation, the feasible actions (A) available for the UAV are "go straight", "turn right" and "turn left". For "tunr right" and "turn left" actions, the UAV rotates by 5 deg. It should be noted that, unlike prior works [19], [21], in our model the actions of the UAV are discrete and do not represent low-level actions like motor angles, or yaw and pitch angles. The justification for such a formulation is that our method

intends to provide a general direction in which the UAV must move in the next instant. Based on the general direction, these physical level controls can be appropriately set. The reward function R is designed using the depth information as

$$R(s,a) = \min\left(1, \frac{d_s - r_d}{\sigma - r_d}\right), \tag{1}$$

where d_s is the distance to the nearest obstacle based on the RGB image obtained, r_d is the radius of the drone and σ is the threshold distance. The distance d_s is obtained from the depth map of the monocular RGB image and hence is an inference from the partial state information. The reason for this specific formulation of the reward function is provided in Section III.

In order to determine the functions P and $\mathcal{O}$, we must be aware of the structure of the environment and the motion dynamics of the UAV. In practice, these are impossible to know. The UAV must be capable of navigating in unknown, unstructured environments in the presence of other factors like wind, turbulence, etc. Thus, we propose a reinforcement learning technique to find an optimal policy for UAV OA. Reinforcement learning is a learning-based approach to solve (PO)MDPs when the model information via P, R (and $\mathcal{O}$) is not available.

C. Algorithm

When model dynamics functions P, R are unavailable (but S is completely observable), one of the well known approaches learns an optimal policy using Q-values. The Q-value $Q^{\pi}(s,a)$ corresponding to the policy π is defined as the expected sum of discounted rewards obtained by taking the action a in state s and following the policy π thereafter. The optimal Q-values are defined as $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$. Once the optimum Q-values in a state s are obtained, the optimal action is picked by finding $\arg\max_{a\in A} Q^*(s,a)$. So, the optimal policy can be computed by finding the optimal action for every state. Q-learning [27] is a model-free iterative algorithm to learn the optimal Q-value of every state-action (s,a) pair. The Q-value update of any such pair is given below:

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)). \tag{2}$$

However, this algorithm suffers from the *curse of dimensionality*, since iteratively learning the Q-values for huge state-action spaces requires maintaining and updating Q-values for all unique state-action pairs. For large state-action spaces, these computations turn out to be infeasible. Deep Q-Networks (DQN) [28] solve this issue by utilizing a neural network parametrized by weights (w) to approximate the Q-value (denoted as $Q(s,a|w)$) for a given state input. DQN also uses an experience replay buffer D which stores experience tuples (s,a,r,s'). The usage of a replay buffer improves the stability of the algorithm. During training, mini-batches of the experience are sampled uniformly and input to the network to calculate the Bellman residual as the loss given by

$$L_i(w_i) = \mathbb{E}_{(s,a,r,s')\sim D}[(r + \gamma \max_{a'} Q(s',a';w_i^-) - Q(s,a;w_i))^2]. \tag{3}$$

Here, w^- represents weights of the target network which is an older copy of network weights lagging behind a few iterations. To achieve a better approximation, the weights are updated using mini-batch gradient descent.

We can adapt the DQN method for POMDP models too. However, it should be noted that in a POMDP model, only observations in Ω are received. Thus, one cannot directly utilize the DQN to solve the OA task. We need a mechanism which can infer the state based on the observations obtained. For this, we augment a recurrency to DQN. The recurrent layer integrates the observations over time to better estimate the underlying state. The memory augmented convolutional neural network architecture to approximate the Q-values from the observations is presented next.

D. Deep Recurrent Q-Network With Temporal Attention

The architecture for approximating Q-values is based on deep recurrent Q-network with attention. This approach essentially keeps track of the past few observations. In the UAV obstacle avoidance application, we keep track of the depth maps obtained from the RGB images. The recurrent network possesses the ability to learn temporal dependencies by using information from an arbitrarily long sequence of observations, while the temporal attention weighs each of the recent observations based on their importance in decision-making.

At time t, the proposed model utilizes a sequence of recent L observations $o_{t-(L-1)}, \ldots, o_t$. Each observation $o_{t-(L-i)}$, $0 \le i \le L$ is a depth map which is processed by convolutional layers of the network, followed by a fully connected layer augmented with LSTM [31] recurrent network layer. The DRQN model with LSTM estimates the Q-value $Q(o_t, h_{t-1}, a_t)$, where h_{t-1} is the hidden state of the recurrent network and is determined as $h_{t-1} = LSTM(h_{t-2}, o_{t-1})$. The hidden state represents the information gathered over time.

Following the LSTM layer, we propose the use of Temporal Attention [30] (see Fig. 2) in our model for evaluating the informativeness of each observation in the sequence. Temporal attention optimizes a weight vector with values depicting the importance of observations at the previous instants. This increases the training speed and provides better generalizability over the training dataset. Let $(\mathbf{v}_{t-(L-1)}, \mathbf{v}_{t-(L-2)}, \ldots, \mathbf{v}_t)^{\top}$ be the vector of feature vectors obtained from the convolutional layers, over the past L observations. For each $1 \le i \le L$, $\mathbf{v}_{t-(L-i)}$ is a feature vector in $\mathbb{R}^{m\times 1}$. The vector of weights $(e_{t-(L-1)}, e_{t-(L-2)}, \ldots, e_t)^{\top}$, for the L feature vectors, is computed using the obtained hidden state values and the feature vector given by:

$$e_{t-(L-i)} = \mathbf{w}^{\top} \tanh\left(W_a h_{t-1} + U_a \mathbf{v}_{t-(L-i)} + b_a\right) \tag{4}$$

in which $\mathbf{w}, b_a \in \mathbb{R}^{a\times 1}, W_a \in \mathbb{R}^{a\times r}, U_a \in \mathbb{R}^{a\times m}$ are all learnable parameters and $h_t \in \mathbb{R}^{r\times 1}$. In (4), $\tanh(\cdot)$ is an activation function which is computed for every element of the vector given by $W_a h_{t-1} + U_a \mathbf{v}_{t-(L-i)} + b_a$. Here, we assume that r is the size of an RNN hidden state, m is the encoding size of CNN and a is the attention matrix size. The tanh activation function is applied pointwise on the vector obtained from $W_a h_{t-1} + U_a \mathbf{v}_{t-(L-i)} + b_a$.

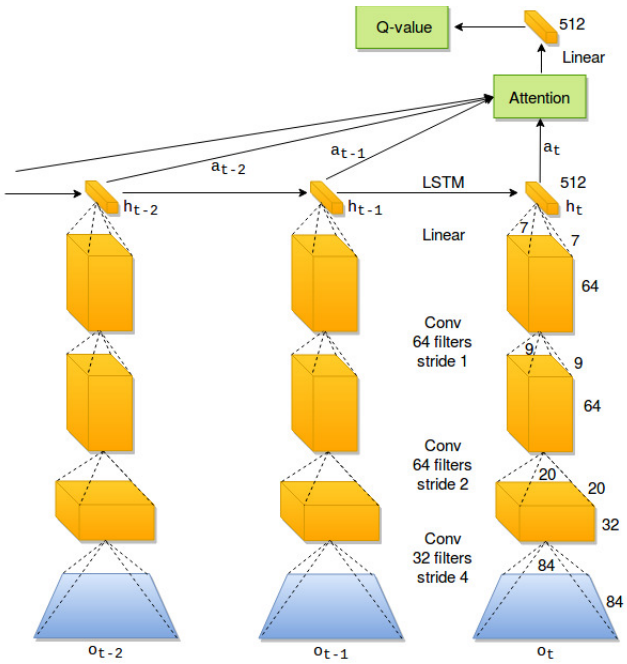

Fig. 2. Control Network: Architecture of Deep Recurrent Q-Network with Temporal Attention. Number of filters, stride and output size are mentioned for each convolutional layer.

These weights are normalized using the softmax function:

$$a_{t-(L-i)} = \frac{\exp(e_{t-(L-i)})}{\sum_{j=1}^{L} \exp(e_{t-(L-j)})}. \tag{5}$$

Further, to predict the Q-values, a context vector is computed using the above calculated softmaxes and hidden states as:

$$\phi(t) = \sum_{j=1}^{L} (a_{t-(L-j)} \mathbf{v}_{t-(L-j)}). \tag{6}$$

The obtained context vector is input to a single fully connected layer with ReLU [35] activation functions that outputs approximated Q-value for each action. The complete architecture is trained by minimizing a loss function as described in [29]. The importance of appropriate weightage of past observations can be understood from a simple example scenario where the UAV flies towards a corner. When it is approaching the corner, the depth map might indicate more space in the front when compared to the sides. The lack of temporal information coupled with limited field of vision of the monocular camera makes the UAV to move ahead towards the corner and crash onto the wall. Using past temporal relevant information, the UAV can safely avoid crashing into the corner. Since UAV navigation involves many such scenarios, we believe the temporal attention layer will give an additional fillip to the UAV OA method.

E. Obtaining Depth Maps From RGB Images

The UAV on-board sensor is limited to providing monocular RGB image data. Effective depth prediction from an RGB image is essential when operating in the physical world. Learning a mapping for image translation $X \rightarrow Y$, given image pairs $\{x \in X, y \in Y\}$, is a challenging task in the computer vision community. In this work, we propose the use of conditional generative adversarial network (cGAN) [34] for this image-to-image translation. This approach uses two separate ConvNets (called as Generator and Discriminator) with BatchNorm layers and ReLU activation layers. The Generator (G) ConvNet is an encoder-decoder structure with skip connections, designed to generate realistic fake images taking $x \in X$ and a noise vector Z as inputs. The Discriminator (D) network classifies randomly picked images as fake or real with a cross-entropy loss. Let θ_D and θ_G represent the weights of the Discriminator and Generator networks, respectively. The Generator is expected to produce images close to the ground truth, while the discriminator is supposed to distinguish between fake images and the real images. Hence in a sense, the objectives of these two networks are opposed to each other. The loss function $\mathcal{L}_{cGAN}(\theta_G, \theta_D)$ defined below reflects these objectives:

$$\begin{aligned}\mathcal{L}_{cGAN}(\theta_G, \theta_D) &= \mathbb{E}_{x,y\sim p_{data}}[\log D(x, y)] \\ &\quad + \mathbb{E}_{x\sim p_{data}(x), z\sim p_z(z)}[\log(1 - D(x, G(x, z)))]. \end{aligned} \tag{7}$$

In the above equation, the variable x is the RGB image and y is its true depth map. The depth map generated by G is denoted as $G(x, z)$. $D(x, y)$ and $D(x, G(x, z))$ are the probabilities of the image belonging to the real class of images. Training a cGAN involves a few steps. Initially, the discriminator is trained on real and fake depth images with the correct labels for few epochs. Following this, the generator is trained using the real/fake predictions from the trained discriminator as its objective. This procedure is repeated for few epochs until the generated fake depth maps are difficult to distinguish from the real depth maps. The cGAN architecture is illustrated in Fig.3. The approach also incorporates L_1 constraint that is then taken together with the cGAN loss via a Lagrangian formulation to generate better near ground truth images. Thus, let

$$\mathcal{L}_{L_1}(\theta_G) = \mathbb{E}_{x,y\sim p_{data}, z\sim p_z(z)}[\| y - G(x, z) \|_1]. \tag{8}$$

Then, the final objective of the model can be analytically represented as

$$\min_{\theta_G} \max_{\theta_D} \{\mathcal{L}_{cGAN}(\theta_G, \theta_D) + \lambda \mathcal{L}_{L_1}(\theta_G)\}, \tag{9}$$

where λ is the Lagrange parameter that is an adjustable hyperparameter. In contrast to previous methods ([14]–[17]) our approach learns a loss function adaptable to the input data, making it domain independent and suitable for our problem of intermediate depth prediction for obstacle avoidance.

F. Remarks

1) It must be noted that the depth maps generated from cGANs as described above still provide limited information with respect to the visual geometry of the environment surrounding the UAV (a similar problem when monocular camera images are used). This issue of partial information was highlighted in Section I. The limited information obtained in stages from cGAN can be stored

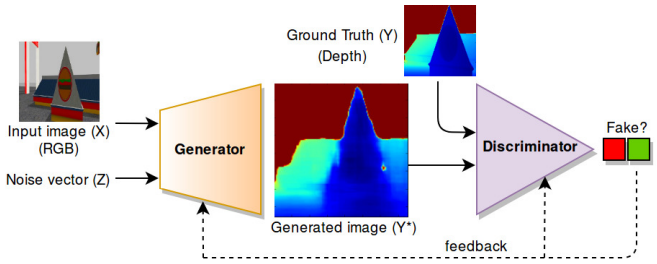

Fig. 3. Depth Network: Conditional GAN architecture.

and collected to make a better navigation decision. The task of using all the relevant partial information obtained in the past is done by the LSTM network architecture as described earlier in this Section.

2) The deep RL method we propose in this section learns optimal Q-values and the optimal policy for the obstacle avoidance task. There are also other policy improvement approaches for learning a good policy. Recently proposed methods like the Asynchronous Advantage Actor-Critic (A3C) [36], deep deterministic policy gradient (DDPG) [37], and dueling network architecture for double deep Q-networks (D3QN) [38] can also be used with our proposed method. For using these methods, one has to change the loss function (3) for the network architecture. Our method involving temporal attention can be easily integrated with A3C, DDPG, and D3QN. However, in this paper, our objective is to highlight the need for using LSTM architecture for partially observable scenarios in UAV obstacle avoidance.

III. Experimental Setup

A. cGAN Training to Obtain Depth Images

The proposed conditional GAN is initially trained on a dataset consisting of 90, 000 RGB-D image pairs. This dataset is collected from simulated environments in Gazebo [39]. We designed 22 different simulated indoor environments in Gazebo, of which few are inspired from [19]. The environments consist of broad and narrow hallways, small and large enclosed areas with floorings ranging from asphalt to artificial turf. The simulated environments also contain structured and unstructured obstacles like humans, traffic cones, tables, etc., placed at random positions and with random orientation. The walls and obstacles with diverse shapes, textures, and colours provide abundant visual information for effective learning. Fig. 4 shows example snapshots of the environment.

The RGB-D image pairs are collected using a Kinect sensor mounted on the flying drone in simulation, covering all possible viewpoints. Further, the dataset is augmented offline by random flipping, adding random jitter and random alteration to the brightness, saturation, contrast, and sharpness. The cGAN network is trained on the entire collected dataset for 20 epochs in batches of size 4 on an NVIDIA Titan X machine. We require the depth network (trained on the simulated images) to predict depth from the unseen real-world images. Predicting depth from either simulated images or real-world images are similar tasks. Thus, it is intuitive to leverage

Fig. 4. Screenshots of some environments designed in Gazebo. We cover a large range of colors, textures, sizes, and shapes for obstacles and walls.

the low-level features learned during training in one task for a different, yet similar task. The basic idea in fine-tuning of depth architecture is exactly this. Once a neural network has been trained on simulated images, the lower layers of the neural network are frozen (so that features learned are kept intact). Then, using the real images, one can just re-train the output layer. By freezing the lower layers, we are using the same features learned earlier to predict depth on the real-world images. The major benefit of this approach is that the network works effectively on similar tasks without the need for training from scratch and also requires substantially low data. In our problem, the network is fine-tuned using 8, 000 and 16, 000 augmented pairs from RGBD-human-explore data [33] and NYU2 dataset [32], respectively.

B. DQN Training With LSTM and Temporal Attention

For RL algorithms to learn an effective collision avoidance policy, the UAV learning agent must have enough experience of undesirable events like collision. Training a learning algorithm on a fragile drone in a physical environment is expensive and hence the performance of DRL algorithms is usually demonstrated on simulated environments. In this work, we build and test our UAV collision avoidance algorithms on the aforementioned simulated environments. Our method initially trains the UAV by starting off with simple hallway environments free of obstacles. Gradually the environment complexity is increased by narrowing down the pathways, enclosing the free space and increasing the density of obstacles.

The proposed control network is trained to learn the observation-action value over the last L observations. As explained in Section II, observations are the depth images received from the simulated Kinect sensor aboard the UAV. The action-value function is learnt for all states and corresponds to the three actions viz., *"go straight"*, *"turn left"*, and *"turn right"*, respectively. It should be noted that the window size L used in the experiments is also a hyperparameter, which needs to be fixed correctly.

TABLE I
HYPER-PARAMETER VALUES OF THE PROPOSED CONTROL NETWORK

Entity	Value
Discount Factor (γ)	0.99
Mini-batch size	32
Learning rate	0.0001
Target network update frequency	400
Input observation size	84×84
Conv1 layer filter size	8×8
Conv2 layer filter size	4×4
Conv3 layer filter size	3×3

In Section IV, we show results for different values of L, which provides us a guideline to fix the value of L.

The agent receives a reward according to the function defined in (1). We set the radius of drone, r_d to 0.292m and σ is set to 1.5m. The reward function (1) penalizes the action of the controller when it is at a distance less than $\sigma - r_d$ from the obstacle. If the agent collides, the episode ends with a penalty of -10. Otherwise, the episode continues until it reaches the maximum number (1000) of steps and terminates with no penalty. The agent also receives an additional $+0.5$ reward if it chooses the "go straight" action. The bias for the "go straight" action helps the UAV to always move forward and turn only when there are obstacles in its clear view. Additionally, to cope with the exploration-exploitation tradeoff, a linear annealed policy is utilized during training with initially chosen value of $\epsilon = 1$ that drops eventually to 0.05 as the final value. The network hyper-parameter values are as shown in Table I.

For the proposed control network to be applicable for robotic applications, the learned policy should be effectively transferable to the real-physical systems. However, this is highly challenging because of the huge gap in visual information available in the real and simulated worlds. Moreover, the depth maps produced by the proposed depth network are too noisy when compared to depth images obtained from the simulated kinect sensor. To overcome this, we degrade the sensor images with Gaussian blurring, random jitter, and superpixel replace (replacement probability 0.5) at the time of training. This additional noise is crucial for non-linear function approximators like neural networks to learn and generalize well, making them robust and transferable to real-world systems.

IV. SIMULATION RESULTS

A. Depth Network Performance on Monocular RGB Images

The depth network is trained as mentioned in the previous section. Once trained, we evaluate the performance of the depth network for two measures - the inference speed and the depth prediction quality, respectively. The inference rate of deep learning models is critical when applied to robotic applications, especially when solving for effective collision avoidance models in flying robots. We tested our model on an NVIDIA GeForce GTX 1050 mobile GPU with 8 GB RAM and Intel core i7 processor machine and observed a sufficient enough data rate of 20Hz on average. In addition, we also implemented previously used depth network in robotic applications [14] and noted an inference rate of 1.4Hz on the same machine configuration.

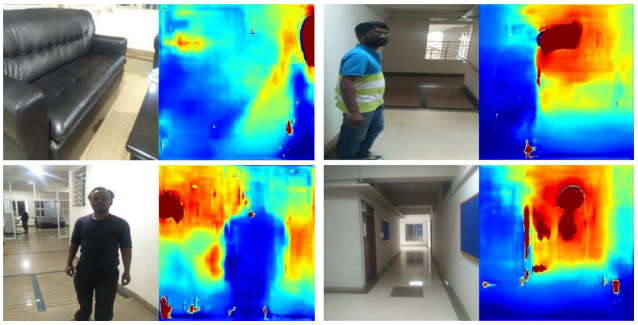

Fig. 5. Example of depth maps generated by the proposed network (trained on simulated data) for completely unseen real world data with variable illumination, color and texture (Red: far, Blue: near).

To assess depth prediction quality of the cGAN architecture, we evaluate the network on unseen simulated data and the fine-tuned data (real-world images) (5, 000 and 2, 500 samples respectively). For evaluation, we compute the L_1 and cGAN loss as in (9) which has been demonstrated to be a better loss function to generate near ground truth images [34]. Table II depicts the network performance in various scenarios.

The first row of values depicts the training and testing loss on manually collected data (data collection is explained in Section III-A). The second row depicts training loss on our simulated dataset, while the testing loss is on a mix of images from the NYU2 [32] and RGBD-human-explore [33] datasets. The third row of values corresponds to the case where the network was trained entirely on the simulated data with fine-tuning. The results in the third row show that such a trained network possesses the ability to generalize well on real-world data. Fig. 5 showcases some samples of the depth maps generated by the cGAN network. The sample images have been taken at the Department of Computer Science and Automation, Indian Institute of Science (IISc) and consist of humans (imitating obstacles) and hallways with varying illumination, colour, and texture which the network has never seen before. The quantitative and qualitative evaluation depicts that the proposed model provides a remarkable boost to the data cycle rate which is essential in robotic applications and can be effectively transferred to real-world systems.

B. Control Network Evaluation

We evaluate the performance of the proposed control network, i.e., Deep Recurrent Q-network with Temporal Attention, and compare it with the baseline DQN previously proposed [23]. We also implement two other policies - random and straight. The random policy picks an action with equal probability for each observation, while the straight policy always picks the "go straight" action. The metric used for performance evaluation is the average number of steps taken until collision with an obstacle. Both the DQN and our proposed model are trained in 12 different simulated indoor environments comprising of hallways and rooms with obstacles of varying structures and sizes. Some snapshots of these environments were illustrated in the earlier section.

As mentioned in Section III-B, we need to select the size of L, which is important for the control network evaluation.

TABLE II
QUANTITATIVE ANALYSIS OF THE DEPTH NETWORK

Scenario		Training loss		Testing loss	
Training set	Testing set	L_1	cGAN	L_1	cGAN
simulated	simulated	0.106	0.666	1.114	0.711
simulated (same training as previous case)	real-world	0.106	0.666	2.779	0.738
simulated + real-world	real-world	0.135	0.692	1.792	0.695

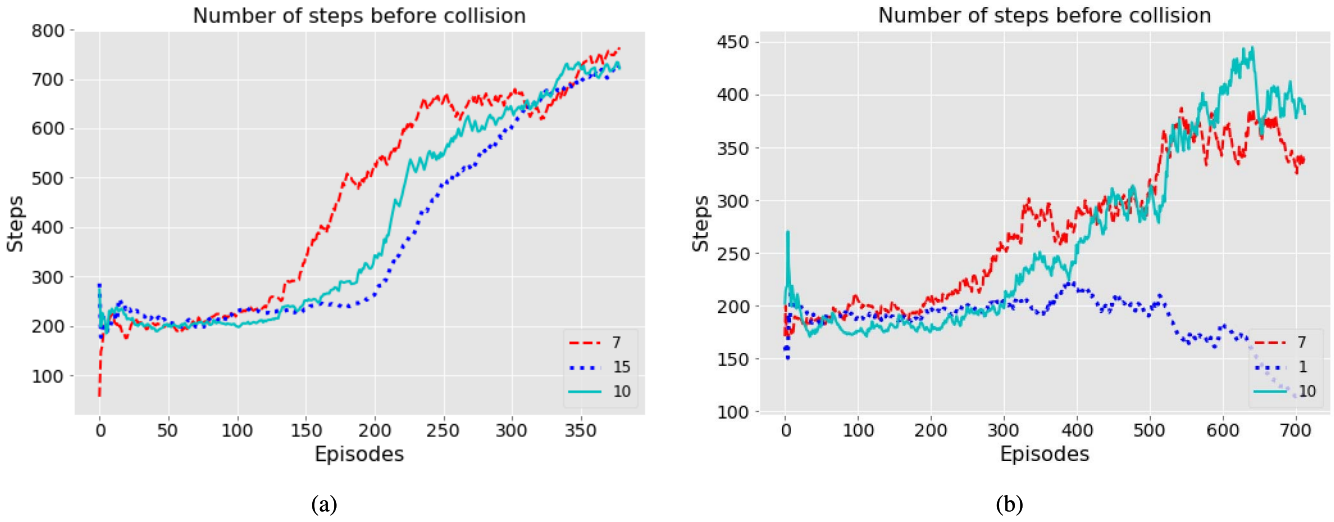

Fig. 6. Performance of DRQN+Attention with different values of L for (a) a straight hallway indoor environment with no obstacles (b) a caffe environment with mobile obstacles.

L is a hyperparameter which needs to be chosen appropriately, such that the relevant information for OA can be utilized. The plots in Fig. 6 show the performance in simulated environments for different values of L. The first plot shows performance for an environment consisting of long straight hallways. As can be observed, for this environment, $L = 7$, $L = 10$ or $L = 15$, does not make a huge difference in the performance. This is because, the ambient environment surrounding the UAV does not provide much information for OA, since it is just a straight hallway which the UAV needs to navigate. However, for the indoor environment filled with mobile obstacles, we see that with $L = 10$, the performance is greatly enhanced (see Fig. 6(b)), when compared to smaller values of L. Hence, based on these observations, in our experiments, we fix $L = 10$.

Figures 7 and 8 show the learning curves during training for both the algorithms for three different environments. These graphs depict the number of steps the UAV takes until collision. Fig. 8 also shows the performance of DRQN for one such environment. As can be observed, partial observability of the environment hinders the performance of DQN in the obstacle avoidance problem. However, the graph shows that augmenting a memory network with attention is beneficial as it retains crucial information gathered over time and this gives an additional fillip to the learning when compared to the no-attention counterpart.

1) Testing in Simulated Environments: The trained models are tested on six randomly selected simulated environments out of the twelve environments used for training. The network takes the noisy depth map and outputs the UAV control signal. The output control signal is expected to safely navigate the UAV within the environment for longer duration. Out of the six environments used for testing, three comprise of enclosed areas with randomly scattered static obstacles of varying sizes and structures (named as Env-1, Env-2, and Env-3 in Table III). The fourth environment (Env-4) is a maze like structure with narrow pathways and no scattered obstacles. The fifth environment (Env-5) is a small enclosed area having poles in between. The sixth environment (Env-6) simulates a cafe-like environment and has 7 human actors randomly walking inside the cafe. The actors are not programmed to avoid the moving UAV and their movement paths are completely random. For this cafe-like environment, the model is initially trained with 3 human actors (randomly moving, not designed to avoid the UAV), but tested with 7 moving actors. We analyze the model performance for 200 episodes in each environment and Table III indicates the average number of steps the UAV takes until collision as well as the standard error. From Table III, it can be seen that using our approach, the UAV flies for the maximum number of time instants until collision.

2) Results: A snapshot of the testing setup is demonstrated in Fig. 9, depicting the learned UAV model maneuvering in Env-6, effectively avoiding the randomly moving human actors inside a cafe. The proposed DRL model also observes a notable inference rate of 60 Hz on NVIDIA GeForce GTX 1050 mobile GPU, essential for robotic applications.

Fig. 10 illustrates the weights attributed to a sequence of 10 images over the recent past used to find the next move of the UAV. It can be analyzed from the images that in an environment consisting of non-stationary obstacles, predicting the

TABLE III

RESULTS INDICATING THE AVERAGE NUMBER OF STEPS TAKEN BY UAV (ALONG WITH STANDARD DEVIATION) UNTIL COLLISION

	Env-1	Env-2	Env-3	Env-4	Env-5	Env-6
Straight	61±16	58±14	76±23	65±12	42±12	27±9
Random	125±84	176±121	113±83	162±88	121±76	42±19
DQN	207±103	229±95	286±142	634±241	384±126	162±83
D3QN	248±109	271±104	297±133	658±253	414±146	177±85
Our Approach	323±134	342±131	326±156	764±273	652±243	247±77

(a) (b)

Fig. 7. Training learning curve of the proposed network and DQN for two different environment settings: (a) An open area with scattered static obstacles of varying sizes and structures. (b) Maze like environment with narrow pathways and no scattered obstacles.

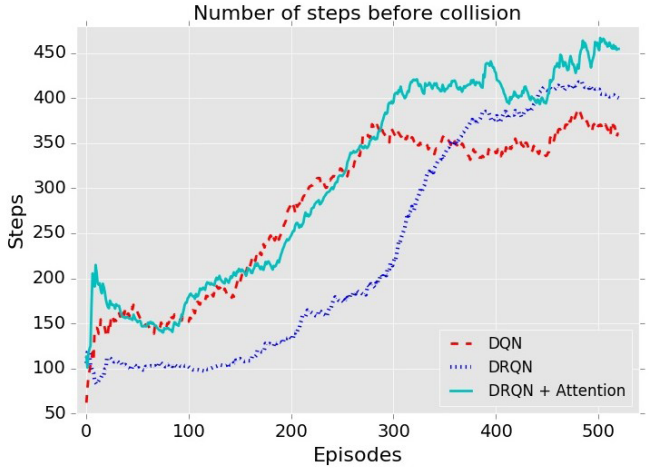

Fig. 8. Training learning curve of the proposed network and DQN for an environment consisting of an enclosed area with scattered static obstacles of varying sizes and structures.

direction of the next step based only on the recent observation (for instance Frame (i) in Fig. 10) is a complicated task. Possessing a memory facilitates an agent to infer the direction of the moving obstacle (such as a human actor walking right) and thereby performing an appropriate action ("turn left") to avoid collision. It is important to note that our proposed algorithm outperforms DQN on different environments. The advantages of the policy learnt by our method are: (i) the UAV smoothly follows a path while avoiding static obstacles, and (ii) in the presence of dynamic obstacles which obstruct the view of the UAV, the UAV skillfully chooses actions to avoid collisions with the dynamic obstacles as well. Video results from these experiments can be seen at https://bit.ly/2PNgWsk.

A UAV is a power-constrained system. Thus, a navigation and obstacle avoidance method must be designed in such a manner that it uses the available battery power judiciously. We say that a UAV wobbles when it takes a long sequence of consecutive left and right turns which do not lead to displacement in its position. Thus, the UAV does not cover any distance when it wobbles, but still, power is consumed in this sequence of right-left movements. This motion without displacement is minimized by our method, which naturally leads to a reduction in power wastage. In order to test for energy efficiency, we designed a simulation environment and tested the proposed method as well as the previously proposed algorithm D3QN [24] over it. The simulated environment consists of straight hallway with two 45◦ turns in between. The navigation task considered is episodic, wherein the UAV starts at a pre-specified initial position. An achievable destination point after the second turn is also specified and the episode terminates when the UAV reaches this destination point. Based on the drop in the battery level and the distance covered, we compute the energy consumption per meter values for both methods by using the power rating of the battery. We observed that for this simulated environment, the average energy consumption over several runs is 0.0571 Wh/m for our approach and 0.0743 Wh/m for D3QN. Thus, this shows that our method achieves a lower value of energy consumption per unit distance traveled when compared to the D3QN method.

Fig. 9. Snapshots of UAV avoiding randomly moving human actors. The yellow arrows show the path the UAV selects in order to avoid the obstacle.

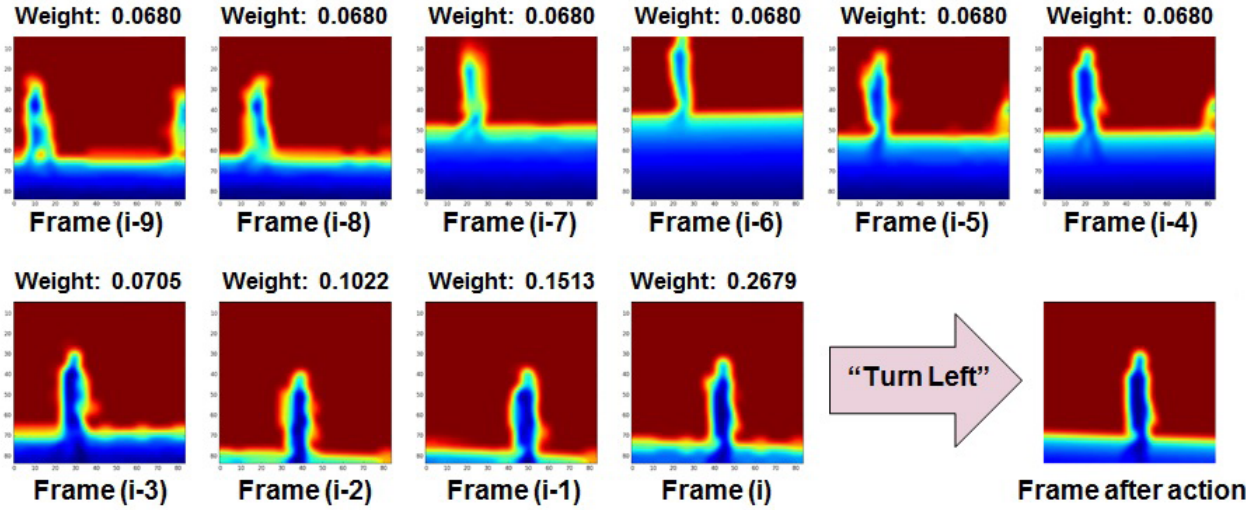

Fig. 10. Temporal Attention weights over the most recent $L = 10$ observations.

V. Discussion

Our method has multiple advantages as well as some limitations that we list below:

- Our approach of adopting cGAN architecture for depth prediction in autonomous aerial systems is novel. Notably, the proposed approach is trained entirely on simulated data and with little fine-tuning on the NYU2 and RGBD-human-explore dataset. The results validate that the model is highly generalizable and qualifies to be adopted in real world applications. As demonstrated by our results, the remarkably high inference rate and transferability of the approach makes it a suitable candidate for intelligent robotic applications.
- We show in our experiments that augmenting DRL with memory networks and temporal attention facilitates the agent to retain vital information gathered from the past observations. This aids the agent towards making better and informed decisions. This learning ability benefits the autonomous agent to maneuver safely in environments without prior knowledge of the surroundings, as well as in environments with moving obstacles. Furthermore, the agent is competent to move deftly near corners (refer supplementary video) which has been found to be a challenging task for the previously proposed controllers ([19], [23]).
- The reward function is designed by considering the energy constraints on aerial systems and time factor in navigation tasks. The bias towards the "go straight" action in the reward function ensures that the UAV maintains its course except when avoiding obstacles in its field of view. In addition, when compared to the D3QN approach, the proposed controller gives smoother trajectories and UAV wobbling is minimized that would otherwise cause a lot of energy to be wasted which is highly undesirable in UAV applications. Our control method minimizes this power wastage and yields considerable power savings. The bias towards the "go straight" action might be problematic at intersections, where the UAV has to turn right or left. However, we would like to emphasize that our proposed method handles only obstacle avoidance and can be easily integrated with a high-level path planner that handles the computation of the path from start to goal position.
- Although the proposed depth prediction network learnt to predict depth maps from the unseen physical world images, the results are noisy. The control network trained with the manually-added noise generalizes and adapts to the noise. However, there is scope for improvement as far as the depth network is concerned. Training depth network on visually high-fidelity simulated data can yield smoother depth predictions.
- For Env-1 and Env-2 which consist of open spaces with scattered obstacles, the number of steps covered before collision is as shown in Table III of the EV. In the evaluation of learnt policy, we observed that both DQN and our approach accrue total reward equal to the number of steps these controllers cover in every simulation run.

When averaged across multiple simulation runs, this then gives same mean and standard deviation as shown in Table III. From this it can be concluded for these simulation environments that both policies seem to be conservative, i.e., at each step, the respective controllers dont let the UAV move very near the obstacle. More extensive experiments and simulations will require to be carried out in order to validate this observation for other environments as well.

VI. Conclusions and Future Work

In this paper, we design and analyze the performance of a Deep Recurrent Q-Network with Temporal Attention which is utilized by a deep RL robotic controller for effective obstacle avoidance of UAV in cluttered and unseen environments. The proposed method first utilizes the cGAN network to predict the depth map from a monocular RGB image which is then used to decide the optimal action. The method addresses the problem of partial observability in obstacle avoidance by retaining crucial information over the long sequence of observations. Experimental results over various settings exhibit significant improvements over Deep Q-Network (DQN) and D3QN algorithms. A potential future direction for our work would be to improve the visual quality of images generated by the cGAN architecture. In GAN architectures, the discriminator block captures the class-specific content from images without imposing constraints on the visual quality of the generated images. The cGAN architecture can be made to generate good quality images by suitably modifying the loss function. Some similarity indices which guarantee structural integrity (e.g., multiscale structural-similarity MS-SIM) can be used for this purpose (see [40]). Another future enhancement would be to use different GAN architectures for depth prediction (see [41], [42]).

The proposed obstacle avoidance method is seen to work well in avoiding obstacles in indoor environments (see Section IV). However, we would also like to test its performance in real outdoor environments. For this, it will be fruitful to include altitude control as an action space variable in the POMDP formulation. This is because, controlling altitude will also help in OA, since some obstacles can be avoided by just changing the height of the UAV from the ground (for e.g., signboards, traffic signs etc.). However, since the action space in our formulation is discrete, the altitude variable should also be suitably discretized to fit into the same POMDP formulation. Further, the reward function needs to be also redesigned to encourage the UAV controller to change altitude while avoiding obstacles. This extension will require more experimentation in real outdoor environments.

In the experiments, we fix start and destination points for the movement of the UAV. However, the path that the UAV takes is not pre-fixed using SLAM or other path following methods. Thus, if the OA algorithm is used along with these methods, then the UAV will avoid obstacles along the pre-fixed path. An extension of our work in this direction will be to intergrate SLAM with our OA method.

References

[1] M. Reinecke and T. Prinsloo, "The influence of drone monitoring on crop health and harvest size," in *Proc. 1st Int. Conf. Next Gener. Comput. Appl. (NextComp)*, Jul. 2017, pp. 5–10.

[2] Y. Ham, K. K. Han, J. J. Lin, and M. Golparvar-Fard, "Visual monitoring of civil infrastructure systems via camera-equipped Unmanned Aerial Vehicles (UAVs): A review of related works," *Visualizat. Eng.*, vol. 4, no. 1, p. 1, Dec. 2016.

[3] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging UAVs for disaster management," *IEEE Pervasive Comput.*, vol. 16, no. 1, pp. 24–32, Jan. 2017.

[4] P. Grippa, D. Behrens, C. Bettstetter, and F. Wall, "Job selection in a network of autonomous UAVs for delivery of goods," in *Proc. 13th Robot., Sci. Syst.* Cambridge, MA, USA: Massachusetts Institute of Technology, Jul. 2017, pp. 1–14. [Online]. Available: http://www.roboticsproceedings.org/rss13/p18.html

[5] M. Funaki and N. Hirasawa, "Outline of a small unmanned aerial vehicle (Ant-Plane) designed for Antarctic research," *Polar Sci.*, vol. 2, no. 2, pp. 129–142, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1873965208000236

[6] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012.

[7] W. G. Aguilar, G. A. Rodríguez, L. Álvarez, S. Sandoval, F. Quisaguano, and A. Limaico, "Visual SLAM with a RGB-D camera on a quadrotor UAV using on-board processing," in *Advances in Computational Intelligence*, I. Rojas, G. Joya, and A. Catala, Eds. Cham, Switzerland: Springer, 2017, pp. 596–606.

[8] T. Gee, J. James, W. Van Der Mark, P. Delmas, and G. Gimel'farb, "Lidar guided stereo simultaneous localization and mapping (SLAM) for UAV outdoor 3-D scene reconstruction," in *Proc. Int. Conf. Image Vis. Comput. New Zealand (IVCNZ)*, Nov. 2016, pp. 1–6.

[9] D.-J. Lee, P. Merrell, Z. Wei, and B. E. Nelson, "Two-frame structure from motion using optical flow probability distributions for unmanned air vehicle obstacle avoidance," *Mach. Vis. Appl.*, vol. 21, no. 3, pp. 229–240, Apr. 2010, doi: 10.1007/s00138-008-0148-9.

[10] H. Alvarez, L. M. Paz, J. Sturm, and D. Cremers, "Collision avoidance for quadrotors with a monocular camera," in *Experimental Robotics*. Cham, Switzerland: Springer, 2016, pp. 195–209.

[11] J. Li *et al.*, "Real-time simultaneous localization and mapping for UAV: A survey," in *Proc. Int. Micro Air Vehicle Conf. Competition (IMAV)*, 2010, pp. 237–242.

[12] N. Eric and J.-W. Jang, "Kinect depth sensor for computer vision applications in autonomous vehicles," in *Proc. 9th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2017, pp. 531–535.

[13] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "MAV navigation through indoor corridors using optical flow," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 3361–3368.

[14] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 239–248.

[15] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, "Fast robust monocular depth estimation for Obstacle Detection with fully convolutional networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4296–4303.

[16] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2024–2039, Oct. 2016.

[17] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6647–6655.

[18] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6612–6619.

[19] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. V. Eycken, "CNN-based single image obstacle avoidance on a quadrotor," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May/Jun. 2017, pp. 6369–6374.

[20] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, "J-MOD^2: Joint monocular obstacle detection and depth estimation," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1490–1497, Jul. 2018.

[21] S. Yang, S. Konam, C. Ma, S. Rosenthal, M. Veloso, and S. Scherer, "Obstacle avoidance through deep networks based intermediate perception," 2017, *arXiv:1704.08759*. [Online]. Available: http://arxiv.org/abs/1704.08759

[22] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3948–3955.
[23] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Proc. 13th Robot., Sci. Syst.* Cambridge, MA, USA: Massachusetts Institute of Technology, Jul. 2017, pp. 1–12. [Online]. Available: http://www.roboticsproceedings.org/rss13/p34.html
[24] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," in *Proc. Workshop New Frontiers Deep Learn. Robot. (RSS)*, 2017, pp. 1–6.
[25] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2017, pp. 2140–2146. [Online]. Available: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14456
[26] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 2, 4th ed. Belmont, MA, USA: Athena Scientific, 2013.
[27] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[28] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[29] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Proc. AAAI Fall Symp. Ser.*, 2015, pp. 1–9
[30] W. Pei, T. Baltrušaitis, D. M. Tax, and L.-P. Morency, "Temporal attention-gated model for robust sequence classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 820–829.
[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[32] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. 12th Eur. Conf. Comput. Vis. (ECCV)*. Berlin, Germany: Springer-Verlag, 2012, pp. 746–760, doi: 10.1007/978-3-642-33715-4_54.
[33] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 2759–2764. [Online]. Available: https://ram-lab.com/dataset/rgbd-human-explore.tar.gz
[34] P. Isola *et al.*, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. CVPR*, Jul. 2017, pp. 5967–5976.
[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn. (ICML)*. Madison, WI, USA: Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425
[36] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, Jun. 2016, pp. 1928–1937.
[37] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971
[38] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, Phoenix, AZ, USA, vol. 2, 2016, p. 5.
[39] *Gazebo Simulator*. Accessed: Nov. 19, 2019. [Online]. Available: http://www.gazebosim.org
[40] J. Snell, K. Ridgeway, R. Liao, B. D. Roads, M. C. Mozer, and R. S. Zemel, "Learning to generate images with perceptual similarity metrics," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 4277–4281.
[41] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," 2017, *arXiv:1703.10717*. [Online]. Available: https://arxiv.org/abs/1703.10717
[42] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

Abhik Singla received the B.Tech. degree in electronics and communications from the National Institute of Technology (NIT), Kurukshetra, in 2017. He is currently pursuing the master's degree in computer science with the University of Massachusetts, Amherst. He worked as a Research Assistant with the Indian Institute of Science (IISc), Bengaluru, for the period of 2017–2019. During this period, he was affiliated with the Department of Computer Science and Automation and the Robert Bosch Centre for Cyber-Physical Systems. His research interests include reinforcement learning, robotics, and deep learning.

Sindhu Padakandla received the B.E. degree in computer science and the M.Sc.(Eng.) degree in computer science from the Indian Institute of Science, Bengaluru, India, in 2008 and 2015, respectively, where she is currently pursuing the Ph.D. degree with the Department of Computer Science and Automation. She has worked with Nokia India Pvt., Ltd., and Symbian Software Pvt., Ltd., prior to her Master's program. Her research interest includes reinforcement learning and its applications to networked intelligent systems.

Shalabh Bhatnagar is currently a Professor with the Department of Computer Science and Automation with a joint appointment with the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bengaluru. His research interests are in stochastic approximation algorithms, simulation-based/data-driven optimization, and reinforcement learning. He is also interested in the applications of these algorithms and techniques to autonomous systems, control of micro-grids, as well as vehicular, communication, and wireless networks. He was a Senior Associate for the International Centre for Theoretical Physics, Trieste, Italy. He is a fellow of the Indian National Science Academy, the Indian Academy of Sciences, and the Indian National Academy of Engineering.