Obstacle Avoidance through Deep Networks based Intermediate Perception

Shichao Yang*, Sandeep Konam*, Chen Ma, Stephanie Rosenthal, Manuela Veloso, Sebastian Scherer

***Abstract*— Obstacle avoidance from monocular images is a challenging problem for robots. Though multi-view structure-from-motion could build 3D maps, it is not robust in texture-less environments. Some learning based methods exploit human demonstration to predict a steering command directly from a single image. However, this method is usually biased towards certain tasks or demonstration scenarios and also biased by human understanding. In this paper, we propose a new method to predict a trajectory from images. We train our system on more diverse *NYUv2* dataset. The ground truth trajectory is computed from the designed cost functions automatically. The Convolutional Neural Network perception is divided into two stages: first, predict depth map and surface normal from RGB images, which are two important geometric properties related to 3D obstacle representation. Second, predict the trajectory from the depth and normal. Results show that our intermediate perception increases the accuracy by 20% than the direct prediction. Our model generalizes well to other public indoor datasets and is also demonstrated for robot flights in simulation and experiments.**

I. Introduction

Autonomous vehicles have raised wide interest in recent years with various applications such as inspection, monitoring and mapping. To navigate autonomously, the vehicles need to detect and avoid 3D obstacles in real time. Various range sensors such as laser [1], stereo cameras [2], and RGB-D depth cameras [3] could be used to build 3D map of the environment. In this work, we focus on more challenging monocular obstacle avoidance. A typical approach to vision-based navigation problem is to use geometric 3D reconstruction techniques such as structure from motion (SfM) and simultaneous localization and mapping (SLAM). They usually track visual features to build sparse [4] or semi-dense 3D map [5]. However, monocular vSLAM may not be robust especially in challenging low-texture or degenerated environments. Besides, their map is usually relatively sparse and cannot fully represent 3D obstacles.

On the other hand, for the image in Fig. 1, a human can easily understand the 3D layout with obstacles and generate the best motion commands. Recently, to mimic human's behavior, there are also some learning based algorithms to predict paths directly from raw RGB image especially with the popularity of Convolutional neural networks (CNNs) [6] [7] [8]. To train CNN models, a large dataset with motion command labels is needed. Previous work mainly utilized human demonstration to collect datasets with labels,

The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA. {shichaoy, skonam}@andrew.cmu.edu

* The first two authors contributed equally.

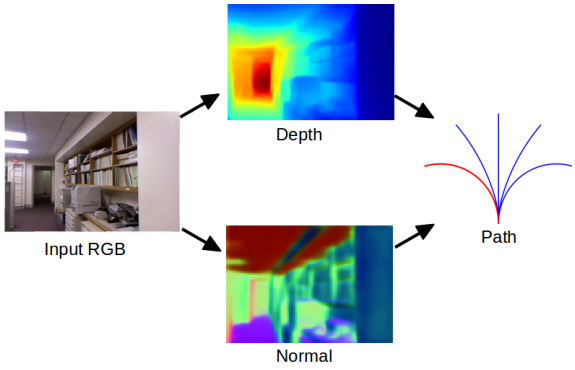


Fig. 1. Method Overview. Instead of directly predicting path from RGB image, we propose intermediate perception: first predict depth and surface normal which are closely related to 3D obstacles, then predict the path from the depth and normal. Both steps utilize CNN.

but it has some problems such as being time-consuming to collect large demonstration data and usually biased to certain scenarios where demonstrations happen. For example, in Fig. 1, people might select 'straight forward' or 'left turning' based on their own analysis of risk and task requirements. They only predict the instantaneous steering command (left, right, etc.) so it requires the learning algorithm to predict frequently. It is also difficult to evaluate whether robots hitting obstacles or not as it depends on the how long the steering command last.

In this paper, we propose to predict trajectories directly from one single image. Trajectories contain more 3D information than instant steering command. To do that, we first propose a new method to get ground truth labels from RGB-D images automatically without human demonstrations. Then we propose a two-stage CNN with intermediate perception. The first stage is to predict the depth and surface normal from images, which are two important geometric properties related to 3D obstacle representation. The second step is to predict a path from the depth and normal maps using another CNN model. In the latter experiments, we demonstrate that our method is more robust compared to direct CNN prediction from raw RGB images. An overview of the proposed perception method is presented in Fig. 1.

In all, our main contributions are:

- Generate ground truth trajectory label from images automatically based on the designed 3D cost functions instead of human demonstration limited to certain scenarios.
- Propose novel idea of mediated perception to instruct CNN to learn depth and surface normal before path

prediction. It is more accurate than direct prediction demonstrated in the experiments.
- Apply to robot simulation and real flight. Robot is able to avoid various configuration of obstacles.

We first provide the related work in Section II. Details of the proposed method including the dataset creation and CNN architecture is presented in Section III. In Section IV, we present experiments on several public datasets, dynamic simulations and real flights. Finally, the paper is concluded in Section V.

II. Related Work

There are two basic categories of methods that utilize monocular images to detect obstacles and generate motion commands. The first category is to analyse the object position and size based on monocular motion cues and low-level features. Mori *et al.* detect SURF features and analyze their size change in consecutive frames to detect front obstacles without geometric assumption or training on the scenes but may not work for lateral and far away obstacles. Optical flow [9], vanishing points direction [10] and wall-floor intersection detection [11] can also be utilized but they are usually suitable for relative easy tasks such as corridor following.

The second category is to directly predict 3D obstacle position or motion commands using learning algorithms especially recent CNNs. 3D scene layout understanding [12] [13] could be an efficient way of representing the environment but it is usually limited to Manhattan environments or corridor with clear boundaries [14] which might not be satisfied in all environments. CNN is widely used in semantic segmentation [15], depth estimation [16], surface normal prediction [17] and so on. Chen *et al.* [6] propose a 'deep driving' system to predict some features such as position on road and distance to car, then generate steering commands (angular velocity) using a feedback controller. Tai *et al* [8] apply CNN to ground robot navigation by directly predicting five steering direction from the input of depth image. Giusti [7] *et al* apply CNN to autonomous trail following using aerial vehicles. In [18], obstacles are directly detected and represented through deep network based depth prediction but it doesn't further apply to robot navigation. Recently, deep reinforcement learning (RL) with Q-learning also shows promising results. [19] use a deep RL approach to directly map raw images to robot motor torques. However, most of RL work is demonstrated in simple or simulated environments such as video games and needs more time for real robotic applications.

III. Visual Perception

We first propose an automatic method of labelling trajectory from image to create a large training dataset. Then CNN is utilized to predict depth and surface normal to form a rough 3D model. This model might be noisy, inaccurate, and even invalid in some cases so we can not use the standard motion planner [20] [21] to generate trajectories. Instead, we utilize the predicted 3D model to predict the trajectory in another CNN model.

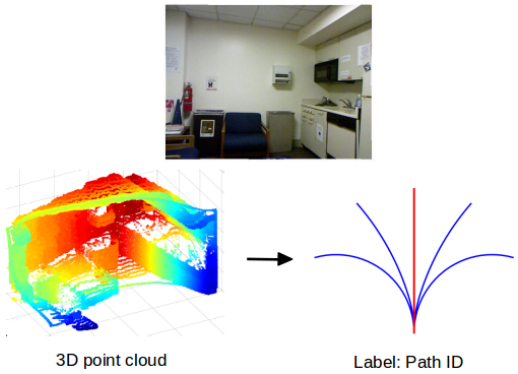


Fig. 2. Generating ground truth path label. Using the provided depth image, a 3D point cloud map is built. Then the best path in red is selected based on the Euclidean obstacle cost and path smoothness cost.

A. Create Dataset

We use the *NYUv2* RGB-D dataset which contains many diverse indoor scenes. For trajectory prediction, in theory, robots could execute numerous continuous 3D trajectories which will be a difficult high-dimensional regression problem. To simplify the problem, we change it into a classification problem by discretizing the continuous trajectory space into five categories of trajectories: left turn, left forward, straight forward, right forward, right turn, shown in the right of Fig. 1. Then for each image, the best one is selected as the ground truth label. These five trajectories are generated by the maximum dispersion theory of Green *et al* [22]. Each path is around 2.5m long. It is difficult for people to tell whether a trajectory hits obstacles or not from 2D images so we design a cost function to automatically select the best path. We first project the provided depth image into 3D point cloud then build a 3D Euclidean distance map. The cost function is commonly used in optimization based motion planner [21] [3]. An example of the 3D point cloud and selected label is shown in Fig. 2. In more detail, the cost function of a trajectory ξ is computed by:

$$J(\xi) = f_{obst}(\xi) + w f_{smooth}(\xi) \tag{1}$$

where w is the weighting parameter. $f_{obst}(\xi)$ is the obstacle cost function [21] penalizing closeness to obstacles:

$$f_{obst}(\xi) = \int_0^1 c_{obs}(\xi(t)) \| \frac{d}{dt} \xi(t) \| dt \tag{2}$$

where $c_{obs}(\xi(t)) = \| \max(0, d_{max} - d(\xi(t)) \|^2$. d_{max} is the maximum distance upon which obstacle cost is available. For NYU dataset, we set $d_{max} = 3.5$m. $d(\xi(t))$ is the distance to obstacles from the distance map. $f_{smooth}(\xi)$ measures the smoothness of the path and penalizes the high derivatives:

$$f_{smooth}(\xi) = \frac{1}{2} \int_0^1 \| \frac{d}{dt} \xi(t) \|^2 dt \tag{3}$$

The final ground truth label distribution for *NYUv2* is shown in Table I. We can see that five categories are nearly

TABLE I

TRAJECTORY LABEL DISTRIBUTION ON *NYUv2* DATASET.

Class ID	Distribution
Left turn	20.70%
Left forward	17.08%
Straight forward	22.15%
Right forward	18.01%
Right turn	22.08%

equally distributed and thus fair for the latter evaluation and comparison of different methods.

B. Intermediate Perception - depth and surface normal

Different from existing CNN based navigation methods [7] [8] to predict the command directly from RGB image, we first predict depth and surface normal. As explained in Section II, there are many 3D geometry understanding methods that could help obstacle avoidance and we choose depth estimation due to its generality for various environments. Surface normals are also useful for navigation for example in Fig. 1, the predicted normal on the right half of the image point to the left, it will give some information to the robot that there is a right wall and it is better to turn left.

There has been lots of work in depth and normal estimation from a single image [16] [17]. We utilize the CNN model from Eigen [16], which is a multi-scale fully convolutional network. It first predicts a coarse global output based on the entire image area, then refines it using finer-scale local networks. The cost functions for depth training are defined as follows: suppose d is the log difference between predicted and ground depth, then depth loss is:

$$L_{depth} = \frac{1}{n}\sum_i d_i^2 - \frac{1}{2n^2}\left(\sum_i d_i\right)^2 + \frac{1}{n}\sum_i \nabla d_i^2$$

where ∇d_i is the gradient magnitude of d. If the ground truth and predicted normal vector are v and v^*, then the dot product between vectors is used to define the normal loss function:

$$L_{normal} = -\frac{1}{n}\sum_i v_i \cdot v_i^*$$

C. Trajectory prediction

We design another CNN network in Fig. 3 to utilize the predicted depth and normal to get the final path classification. It is a modification of standard Alexnet [23] with two inputs. For symmetry, we replicate the depth image (one channel) into three channels. More complicated HHA feature [24] from depth image could also be used. Then depth and normal images learn convolution filters separately and are fused at the fourth layer. By doing this, the final prediction will merge two sources of information. For training, we minimize the standard classification cross-entropy loss:

$$L(C, C^*) = -\frac{1}{n}\sum_i C_i^* \log(C_i) \tag{4}$$

where $C_i = e^{z_i} / \sum_c e^{z_{i,c}}$ is the softmax class probability given the CNN convolution output z.

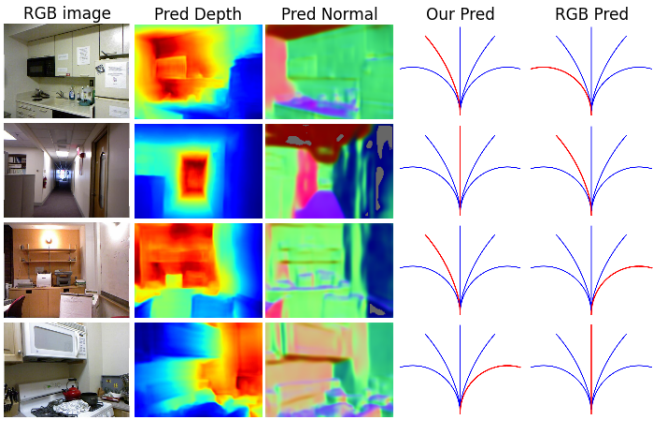


Fig. 4. Example of path prediction on *NYUv2* dataset. The input is only RGB image. Our intermediate prediction and other direct prediction are shown in the last two columns in red color. From left to right: RGB image, predicted depth image, predicted surface normal, and predicted paths. In the top image, two predictions are similar. In the bottom images, our method performs better.

IV. EXPERIMENTS

A. Training and testing

In the training phase, we use the created dataset in Section III-A to train our two-stage CNN separately. In the first stage, the ground truth surface normal is computed by fitting local least-squares planes in the point cloud based on [25]. Actually, we can also directly use the public available CNN model of [16]. In the second stage, we use the *ground truth* depth and surface normal as inputs to train CNN for path classification. We also augment the training data by flipping images horizontally. Note that for surface normal flipping, we need to reverse the horizontal normal component.

In the testing phase, we only use the raw RGB image as inputs of our two-stage CNN.

B. NYUv2 dataset Evaluation

The baseline for comparison is to directly train Alexnet CNN model to predict path label from raw RGB images, which has been adopted in many other CNN navigation works [6] [7] [8]. We use the same settings of parameters for CNN weight initialization and training for comparison.

1) Qualitative Results: A qualitative comparison between our method and baseline method is shown in Fig. 4. We can see that our method generates safer and more reasonable trajectory most of the time.

2) Quantitative Results: The result of the two methods is presented in Table II. The '*Accuracy*' column represents the standard classification accuracy. We also note that there is no clear distinction between different trajectories, for example, in some open space environments, going forward and turning both works. Therefore, to make the evaluation more meaningful, we also check whether the predicted path lies in the top two best ground truth paths shown as '*Top-2 accuracy*' column. From the table, our method with intermediate perception performs much better compared to direct

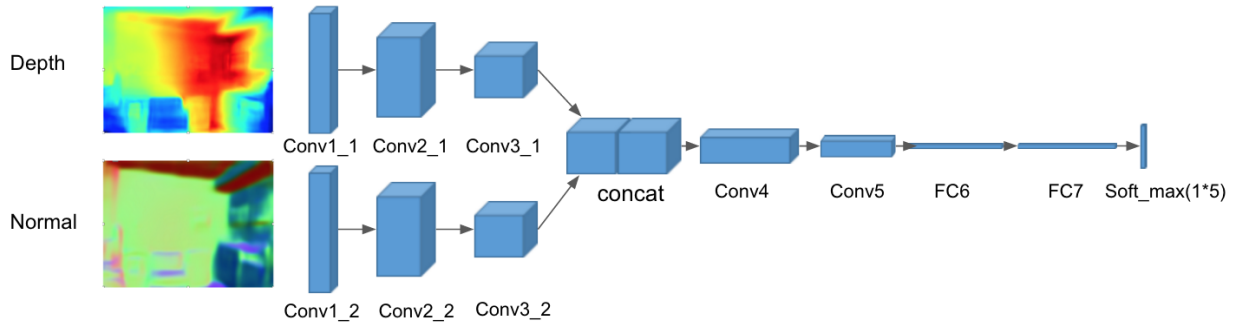


Fig. 3. Proposed model architecture to predict path from depth and surface normal. It has two branches at the beginning to receive two input information. The prediction result is a label within five classes. For predicting depth and normal images, we use the model of [16].

TABLE II

COMPARISON OF PATH PREDICTION ACCURACY ON *NYUv2* DATASET (%).

Method	Accuracy	Top-2 accuracy	Safe prediction
Ours	**64.07**	**82.11**	**92.08**
Our depth only	60.34	78.09	90.68
Direct predict	39.20	60.19	78.73
Random predict	20.23	40.00	62.38

TABLE III

CONFUSION MATRIX FOR PATHS PREDICTION ON *NYUv2*.

Class	Left	Left+	Straight	Right+	Right
Left	0.65	0.11	0.11	0.03	0.10
Left+	0.14	0.64	0.12	0.04	0.06
Straight	0.07	0.14	0.60	0.10	0.08
Right+	0.07	0.05	0.21	0.56	0.11
Right	0.09	0.02	0.06	0.10	0.73

perception from RGB image, with an accuracy increase of 20%. It is also more accurate (42% improvement) compared to random prediction.

We also report the percentage of predicted paths hitting the 3D obstacles shown in the '*Safe prediction*' column, which is an important metric related to robot's safety. This metric is not evaluated in other works because they usually predict instant steering command which cannot tell safety or not precisely. 92.08% of the predicted paths using our proposed methods are safe. We also need to know that during actual robot applications, robots will continuously re-predict trajectories from new coming images instead of following one trajectory till ends. So we can consider obstacles only within a certain distance (eg. $2m$) to the robots, the safe prediction ratio increases to 95.55%. Note that since images are usually taken at some distance from obstacles, random prediction will not necessarily generate unsafe trajectory hitting obstacles all the time.

Results of our staged CNN model with only depth prediction can be seen in Table II. It performs slightly worse compared to combining depth and surface normal. The confusion matrix of our method is shown in Table III. We can see that most of the times, the prediction is correct and only a few times, the prediction will generate totally opposite direction. The reason why our network performs better is that we add ground truth depth information in the first stage's CNN model training which will guide the network to learn 3D scene information related to obstacle avoidance. However, the baseline method might not be able to learn this 3D information just from a single image without ground truth depth instruction.

TABLE IV

COMPARISON OF PATH PREDICTION ON *Ram-lab* DATASETS (%).

Method	Accuracy	Top-2 accuracy	Safe prediction
Ours	**50.05**	**73.21**	**91.10**
Direct predict	42.68	65.67	87.92
Random predict	20.00	40.00	72.37

C. Other Public Indoor dataset Evaluation

Recently, Tai *et al* [8] propose the *Ram-lab* RGB-D corridor dataset for CNN based autonomous navigation. We cannot directly compare with their method as they predict steering command from true depth image instead of RGB image. Besides, their dataset labels come from human demonstration, which is different from ours and also not even distributed. So we generate a new path label for their dataset using the same method and parameters in Section III-A. Depth image is pre-processed by the cross-bilateral filter of Paris [26] to fill the missing regions in order to project to 3D space. Note that we directly apply the trained model from *NYUv2* on this dataset without tuning any parameters. Some prediction examples are shown in Fig. 5. A metric comparison on all the images is presented in Table IV.

We can find that our method still outperforms the baseline methods in terms of classification *accuracy* and *Safe prediction* ratio. However, the improvement is not as large as the *NYUv2* dataset mainly because *Ram-lab* dataset contains only corridor images with similar scenes structures and appearance.

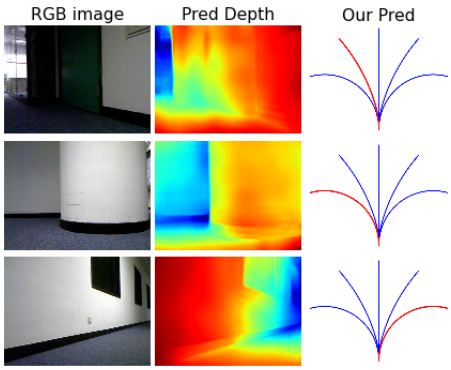


Fig. 5. Some prediction examples in *Ram-lab* dataset images.

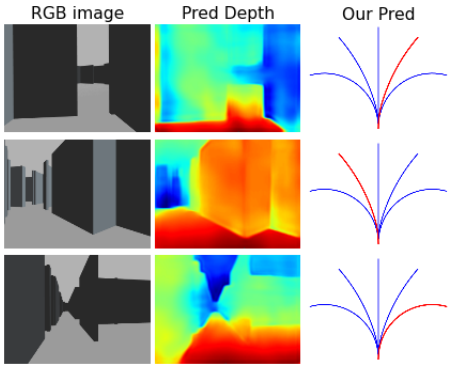


Fig. 7. Some prediction examples in gazebo simulations.

D. Quadrotor simulation flight

We also evaluate our perception algorithm for continuous robot flight in simulations shown in Fig. 7. Our CNN model predicts a trajectory in real-time, then the robot will follow the trajectory until a new predicted trajectory comes. Yaw heading of quadrotor is set as the tangent direction of the trajectory. We use the ardrone quadrotor dynamic simulator developed by Engel *et al* [27] in the default gazebo willowgarage world model without texturing it. A simple PID controller is used for trajectory tracking. The final performance is affected by both the CNN prediction error and also the uncertainty in dynamic modelling and trajectory tracking. In the simulation, there are various kinds of door and wall configurations to test the robustness of our system. Some of the prediction examples are shown in Fig. 7 and the top-view trajectory prediction is shown in Fig. 8. The whole history is shown in Fig .6. We can see that even the scene is quite different from *NYUv2* dataset, our CNN model still works well and the quadrotor is able to travel long distance and cross very narrow doors.

We also provide quantitative analysis of the simulation flight performance shown in Table V. The ground truth pose is provided by gazebo. The mean robot to obstacle distance is 0.98m. The narrowest door is only 0.78m width while quadrotor width is 0.52m. The average prediction time of our whole system is 38.5ms on a GeForce GTX 980 Ti GPU so it is able to run in real time over 25Hz.

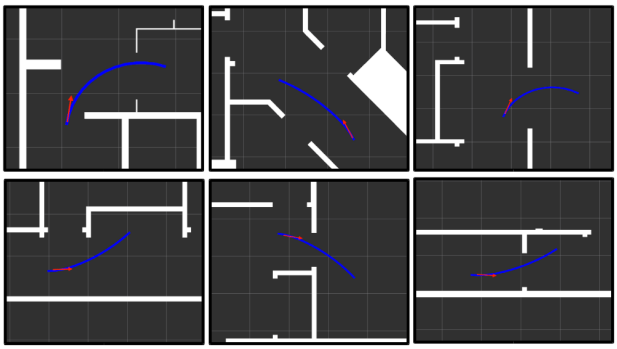


Fig. 8. Top view of path prediction in gazebo simulations. The red arrow represents the robot pose, the blue line is the predicted trajectory. The white areas in the image represents the obstacles.

TABLE V

STATISTIC ANALYSIS OF QUADROTOR SIMULATION.

Method	Scene I
Traveling distance	264.8 m
Distance to obstacles	0.98 m
Prediction time	38.5 ms

We need to note that our CNN model can only be used for local obstacle avoidance so the quadrotor can easily come to a dead end if there is no high level planning shown at the top scene of Fig. 6. Moreover, we currently only use five primitive paths which might not be enough for real world requirements. For example, if there is sharp turning, U-turn trajectory or stop primitives to select, robot flight could perform better. These techniques have been used in some actual flight system [3].

E. Real quadrotor flight

We also test our algorithm on real quadrobot flight. The platform is Parrot bebop quadrotor shown in Fig. 9(a). It can send the forward-facing camera image via WiFi to our host laptop which predicts the trajectory and sends back velocity command to bebop. The laptop is equipped with GeForce GTX 960M GPU for CNN prediction which takes about 0.143s per frame. There is also image transmission delay about 0.2s. We still use the trained model from *NYUv2* dataset. We need to note that there are some other important factors affecting the actual flight performance such as state estimation and control. Due to instable and sometimes jumping state estimation from bebop drone, we cannot use the position feedback in the trajectory tracking which deteriorates the whole performance. We test our vehicle in three indoor environments: straight corridor, turning, front obstacles shown in Fig. 9(a). Some prediction images on the fly are shown in Fig. 9(b). More results could be found in the supplementary video. We can see the robots can predict a reasonable trajectory in these tasks most of the time.

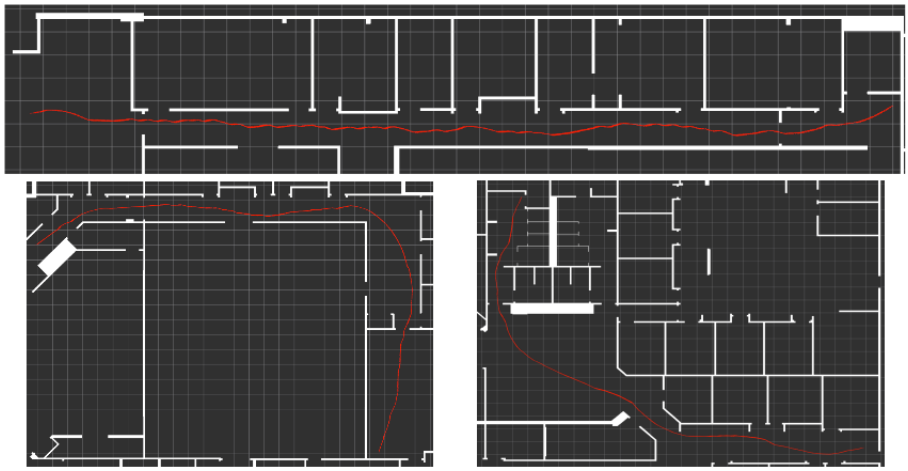
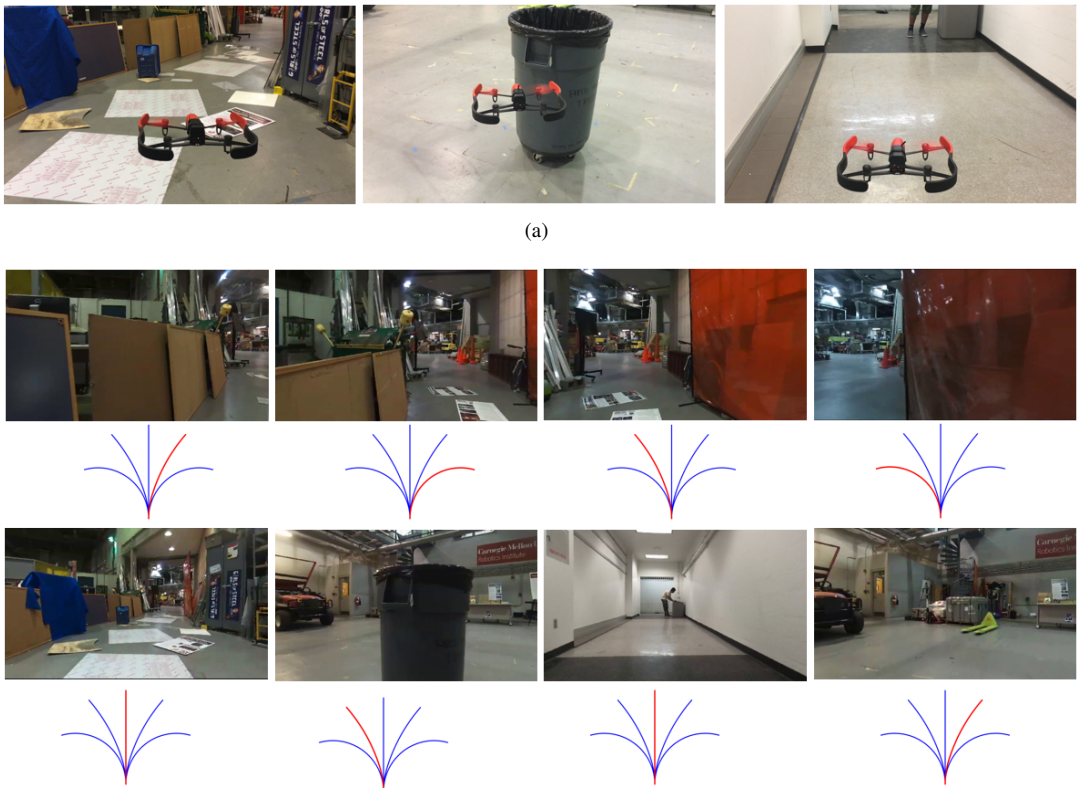


Fig. 6. Some scenarios of the dynamic simulation. White area represents obstacles and red curve is the quadrotor history pose.



(a)

(b)

Fig. 9. (a) Real flight scenes including curved corridor, front obstacles and corridor following (b) Eight prediction examples from quadrotor's view on the fly. For each image, the path image below it shows the predicted path. More results could be found in the supplementary video.

V. Conclusion

In this paper, we propose a CNN based navigation system applicable to various environments. We start by training our system on public *NYUv2* dataset instead of human demonstration datasets limited to certain scenarios in existing works. By designing 3D cost functions, we are able to automatically choose the best trajectory from each RGB-D image. Previous methods usually predict steering commands for robots which is difficult to evaluate whether it hits obstacles or not. Instead, we choose to predict a 3D trajectory which contains more meaningful 3D information.

Our perception system is also different from existing methods. We first instruct CNN to predict depth and surface normals which give the robot a sense of 3D obstacle position and scene layout and thus it can predict the path more precisely. Results on the *NYUv2* dataset and other public datasets show that this intermediate perception performs much better and robustly than direct path prediction from raw RGB image.

We also apply our CNN model for quadrotor simulation and actual flight in different environments without retraining. The robot is able to avoid various obstacles in real time. Some future works include integrating it with high level motion planning, accurate state estimation and also adding more trajectories into it for example U-turn primitives. Integrating multiple frame's prediction should also improve the performance.

References

[1] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, 2009.

[2] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-based state estimation for autonomous rotorcraft mavs in complex environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1758–1764. IEEE, 2013.

[3] Zheng Fang, Shichao Yang, Sezal Jain, Geetesh Dubey, Silvio Maeta, Stephan Roth, Sebastian Scherer, Yu Zhang, and Stephen Nuske. Robust autonomous flight in constrained and visually degraded environments. In *Field and Service Robotics*, pages 411–425. Springer, 2016.

[4] Raul Mur-Artal, JMM Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *Robotics, IEEE Transactions on*, 31(5):1147–1163, 2015.

[5] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, pages 834–849. Springer, 2014.

[6] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[7] Alessandro Giusti, Jerome Guzzi, Dan Ciresan, Fang-Lin He, Juan Pablo Rodriguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jurgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letter*, 2016.

[8] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[9] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, 4(1):13–16, 2007.

[10] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 5776–5783. IEEE, 2011.

[11] Kyel Ok, Duy-Nguyen Ta, and Frank Dellaert. Vistas and wall-floor intersection features: Enabling autonomous flight in man-made environments. In *2nd Workshop on Visual Control of Mobile Robots (ViCoMoR): IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pages 7–12, 2012.

[12] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *Computer vision, 2009 IEEE 12th international conference on*, pages 1849–1856. IEEE, 2009.

[13] Shichao Yang, Daniel Maturana, and Sebastian Scherer. Real-time 3D scene layout from a single image using convolutional neural networks. In *Robotics and automation (ICRA), IEEE international conference on*, pages 2183 – 2189. IEEE, 2016.

[14] Shichao Yang, Yu Song, Michael Kaess, and Sebastian Scherer. Pop-up SLAM: a semantic monocular plane slam for low-texture environments. In *Intelligent Robots and Systems (IROS), 2016 IEEE international conference on*. IEEE, 2016.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[16] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[17] Xiaolong Wang, David Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015.

[18] Michele Mancini, Gabriele Costante, Paolo Valigi, and Thomas A Ciarfuglia. Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4296–4303. IEEE, 2016.

[19] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[20] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[21] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[22] Colin Green and Alonzo Kelly. Toward optimal sampling in the space of paths. In *13th International Symposium of Robotics Research*, 2007.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[24] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[25] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[26] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. In *European conference on computer vision*, pages 568–580. Springer, 2006.

[27] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems*, 62(11):1646–1656, 2014.