Autonomous Navigation in Complex Environments with Deep Multimodal Fusion Network

Anh Nguyen[1], Ngoc Nguyen[2], Kim Tran[2], Erman Tjiputra[2], Quang D. Tran[2]

***Abstract*— Autonomous navigation in complex environments is a crucial task in time-sensitive scenarios such as disaster response or search and rescue. However, complex environments pose significant challenges for autonomous platforms to navigate due to their challenging properties: constrained narrow passages, unstable pathway with debris and obstacles, or irregular geological structures and poor lighting conditions. In this work, we propose a multimodal fusion approach to address the problem of autonomous navigation in complex environments such as collapsed cites, or natural caves. We first simulate the complex environments in a physics-based simulation engine and collect a large-scale dataset for training. We then propose a Navigation Multimodal Fusion Network (NMFNet) which has three branches to effectively handle three visual modalities: laser, RGB images, and point cloud data. The extensively experimental results show that our NMFNet outperforms recent state of the art by a fair margin while achieving real-time performance. We further show that the use of multiple modalities is essential for autonomous navigation in complex environments. Finally, we successfully deploy our network to both simulated and real mobile robots.**

I. INTRODUCTION

Autonomous navigation is a long-standing field of robotics research, which provides an essential capability for mobile robot to execute a series of tasks on the same environments performed by human everyday. In general, the task of autonomous navigation is to control a robot navigate around the environment without colliding with obstacles. It can be seen that navigation is an elementary skill for intelligent agents, which requires decision-making across a diverse range of scales in time and space. In practice, autonomous navigation is not a trivial task since the robot needs to close the perception-control loop under the uncertainty in order to obtain the autonomy.

Recently, the learning-based approaches (e.g., deep learning models, etc.) have demonstrated the ability to directly derive end-to-end policies which map raw sensor data to control commands [1], [2]. This end-to-end approach also reduces the complexity of the implementation and effectively utilizes input data from different sensors (e.g., depth camera, laser) thereby reducing cost, power and computational time. One more advantage is that the end-to-end relationship between input data and control outputs can result in an arbitrarily nonlinear complex model (i.e., sensor to actuation) which has yielded surprisingly encouraging results in different control problems such as lane following [3], autonomous driving [4], and Unmanned Aerial Vehicles (UAV) control [5]. However, most of the previous works are only tested in man-made normal environments, while navigating in complex environments such as collapsed houses/cities suffered from a disaster (e.g. an earthquake) or a natural caves still remains as an open problem.

Fig. 1. A visualization of a collapsed city, a complex environment from our simulation. **Top:** The collapsed city from a top view; **Bottom:** A mobile robot's view.

Unlike normal environments (e.g., man-made road) which have clear visual clues in normal condition, complex environments such as collapsed cities or natural caves pose significant challenges for autonomous navigation [6] [7]. The main reason is that complex environments usually have very challenging visual and physical properties. For example, the collapsed cities may have constrained narrow passages, vertical shafts, unstable pathways with debris and broken objects (Fig. 1); or the natural caves often have irregular geological structures, narrow passages, and poor lighting condition. Autonomous navigation with intelligent robots in complex environments, however, is a crucial task, especially in time-sensitive scenarios such as disaster response, search and rescue, etc. Recently, the DARPA Subterranean Challenge [8] was organized to explore novel methods for quickly mapping, navigating, and searching in complex underground environments such as human-made tunnel systems, urban underground, and natural cave networks.

Inspired by the DARPA Subterranean Challenge, in this work, we propose a learning-based system for end-to-end mobile robot navigation in complex environments such as collapsed cities, collapsed houses, and natural caves. To

[1]Department of Computing, Imperial College London, UK a.nguyen@imperial.ac.uk

[2]AIOZ Pte Ltd, Singapore quang.tran@aioz.io

overcome the difficulty in data collection, we build a large-scale simulation environment which allows us to collect the training data and deploy the learned control policy. We then propose a new Navigation Multimodal Fusion Network (NMFNet) that effectively learns the visual perception from sensor fusion and allows the robot to autonomously navigate in complex environments. To summarize, our main contributions are as follows:

- We introduce new simulation models that can be used to record large-scale datasets for autonomous navigation in complex environments.
- We present a new deep learning method that fuses both laser data, 2D images, and 3D point cloud to improve the navigation ability of the robot in complex environments.
- We show that the use of multiple visual modalities is essential to learn a robust robot control policy for autonomous navigation in complex environments in order to deploy in real-world scenarios.

The remainder of this paper is organized as follows: Section II discusses the related background. Section III presents the visual multimodal input used in our method. Section IV introduces our new multimodal fusion network and its architecture. In section V, we present our extensively experimental results. Section VI concludes the paper and discusses the future work.

II. Related Work

Multiple sensor fusion for autonomous robot navigation is a popular research topic in robotics [9]. Traditional methods tackle this problem using algorithms based on Kalman Filter [10]. The advantage of this approach is the ability to fuse data from different sensors and sensor types such as visual, inertial, GPS, or pressure sensors. Lynen et al. [11] proposed a method based on Extended Kalman Filter (EKF) for Micro Aerial Vehicle (MAV) navigation. In [12], the authors developed an algorithm based on EKF to estimated the state of an UAV in multi-environments in real-time. Mascaro et al. [13] proposed a graph-optimization method to fuse data from multi sensors for UAV pose estimation. Apart from the traditional localization and navigation task, multimodal fusion is also used other applications such as object detection [14] or semantic segmentation [15], [16] in changing environments. In both [14], [16] multimodal data from visual sensors are combined and learned in a deep learning framework to deal with challenging lighting conditions.

More recently, many methods have been proposed to directly learn control policies from raw sensory data. These methods can be divided into two main categories: reinforcement learning [17] and supervised learning [18]–[20]. With the rise of deep learning, Convolution Neural Networks (CNN) was widely used to train an end-to-end perception system [21]–[26]. In [27], Bojarski et al. proposed the first end-to-end navigation system for autonomous car using 2D images. Smolyanskiy et al. [28] extended this idea for flying robots using three cameras as the input. Similarly, the authors of DroNet [29] used CNN to learn the steering angle and predict the collision probability given the RGB image as the input. Gandhi et al. [30] introduced a navigation method for UAV by learning from negative and positive crashing trials. In [31] [32], CNN and Variational Autoencoder were combined to estimate the steering control signal. Monajjemi et al. [5] proposed a new method for agile UAV control. More recently, the authors in [33] proposed to combine the navigation map with visual input to learn a deterministic control signal.

Reinforcement learning algorithms have been widely used to learn general policies from robot experiences [17], [34], [35]. In [36], the authors introduced a continuous control framework using deep reinforcement learning. Zhu et al. [37] addressed the target-driven navigation problem given an input picture of a target object. Wortsman et al. [38] introduced a self-adaptive visual navigation system using meta-learning. The authors in [39] used semantic information and spatial relationships to let a robot navigate to target objects. In [40], an end-to-end regression system was introduced for UAV racing in simulation. The authors in [41] [42] proposed to train the reinforcement policy in simulation environments, then transfer the learned policy to the real-world. In [43] [44], the authors combined deep reinforcement learning with CNN in an effort to leverage the advantages of both techniques. Piotr et al. [7] proposed a method with augmented memory to train autonomous agents to navigate within large and visually rich environments (complicated 3D mazes).

While reinforcement learning methods learn the general control policies with nice mathematical formulation, they require many trial and error experiments which are dangerous and not realistic in real safety-critical robotic platforms [39] [40]. On the order hand, supervised learning methods use pre-collected data to learn the control policies. The supervision data can be obtained from the real human expert trajectories [29], [30] or traditional controllers [45]. This is time-consuming and costly but doable with the real robots. Therefore, the supervised learning approach is usually more favorable over the reinforcement learning method when working with real robot platforms. However, it is not trivial to handle the domain-shift between expert guidance and the real robot trajectories in supervised learning methods.

In this work, we choose the end-to-end supervised learning approach for the ease of deploying and testing in real robot systems. We first simulate the complex environments in physics-based simulation engine and collect a large-scale for supervised learning. We then proposed NMFNet, an effective deep learning framework to fuse visual input and allow the robots to navigate autonomously in complex environments.

III. Multimodal Input

Complex environments such as natural cave networks or collapsed cities pose significant challenges for autonomous navigation due to their irregular structures, unexpected obstacles, and the poor lighting condition inside the environments. To overcome these natural difficulties, we use three visual input data in our method: RGB image $\mathcal{I}$, point cloud $\mathcal{P}$, and

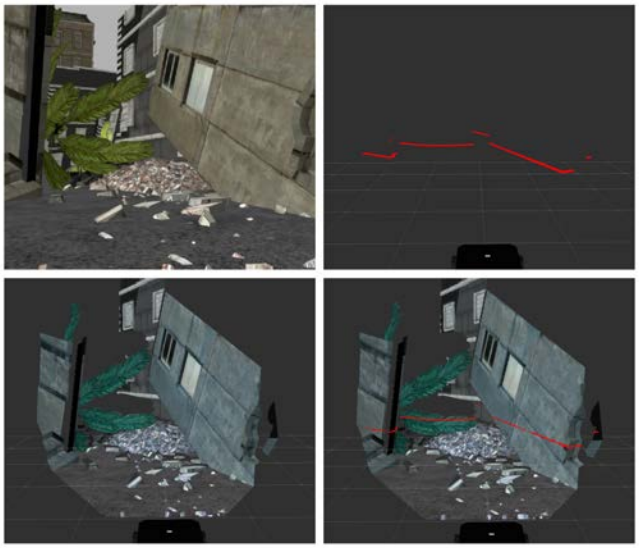


Fig. 2. An illustration of three visual modalities used in our network. **Top row:** The RGB image (left) and the distance map from laser scanner (right). **Bottom row:** The 3D point cloud (left) and the overlay of the distance map in 3D point cloud (right).

distance map $\mathcal{D}$ obtaining from the laser sensor. Intuitively, the use of all three visual modalities ensures that the robot's perception system has meaningful information from at least one modality during the navigation under challenging conditions such as lighting changes, sensor noise in depth channels due to reflective materials, or motion blur, etc.

In practice, the RGB images and point clouds are captured using a depth camera mounted in front of the robot while the distance map is reconstructed from the laser data. In complex environments, while the RGB images and point clouds can provide the visual semantic information for the robot, the robot may need more useful information such as the distance map due to the presence of various obstacles. The distance map is reconstructed from the laser data as follows:

$$\begin{aligned} x_i &= x_0 + d * cos(\pi - \phi * i) \\ y_i &= y_0 - d * sin(\phi * i) \end{aligned} \tag{1}$$

where x_i, y_i is the coordinate of i^{th} point on 2D distance map. x_0, y_0 is the coordinate of robot. d is the distance from the laser sensor to the obstacle, and ϕ is the incremental angle of the laser sensor.

To keep the low latency between three visual modalities, we use only one laser scan to reconstruct the distance map. The scanning angle of the laser sensor is set to $180°$ to cover the front view of the robot. This will help the robots aware of the obstacles from its far left/right hand side, since these obstacles may not be captured in the front camera which provides the RGB images and point cloud data. We notice that all three modalities are synchronized at each timestamp to ensure the robot is observing the same viewpoint at each control state. Fig. 2 shows a visualization of three visual modalities used in our method.

IV. Methodology

As motivated by the recent trend in autonomous driving [28], [29], [33], our goal is to build a framework that can directly map the input sensory data $\mathbf{X} = (\mathcal{D}, \mathcal{P}, \mathcal{I})$, to the output steering commands $\mathbf{Y}$. To this end, we design NMFNet with three branches to handle three visual modalities. The architecture of our network is illustrated in Fig. 4.

A. 2D Features

Learning meaningful features from 2D images is the key to success in many vision tasks. In this work, we use ResNet8 to extract deep features from the input RGB image and laser distance map. The ResNet8 has 3 residual blocks, each block consists of a convolutional layer, ReLU, skip links and batch normalization operations. A detailed visualization of ResNet8 architecture can be found in Fig. 3. As in [29], we choose ResNet8 to extract deep features from the 2D images since it is a light weight architecture and can achieve competitive performance while being robust again the vanishing/exploding gradient problems during training.

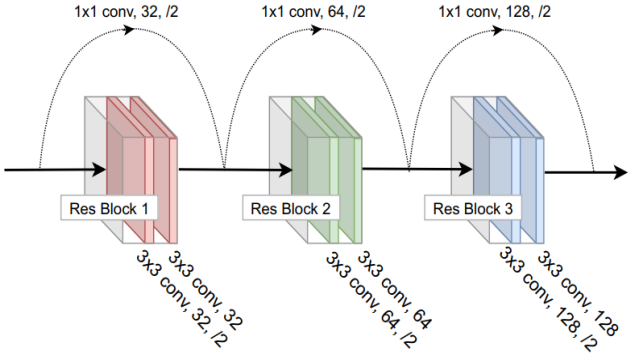


Fig. 3. A detailed visualization of ResNet8.

B. 3D Point Cloud

While the robot is navigating in complex environments, relying on 2D visual information maybe not enough. For example, the RGB images from the front camera of the robot are widely used in many end-to-end visual navigation systems [29], [33], however, in environments such as natural caves, the lighting condition can be a challenge to obtain clear visual images. Therefore, we propose to use point cloud as another visual input for autonomous navigation in complex environments.

Specifically, we use the point cloud associated with the RGB images from the front camera of the robot. Although the point cloud from depth camera is ordered, it contains many points (e.g., in our camera setting, we have $640 \times 480 = 307,200$ points in each cloud), including many missing points. In practice, due to the memory constraints, it is impractical to learn the geometric information from the huge point clouds [46]. Therefore, we remove all the missing points and randomly select $20,480$ points to represent the cloud, hence the point cloud becomes unordered. The point

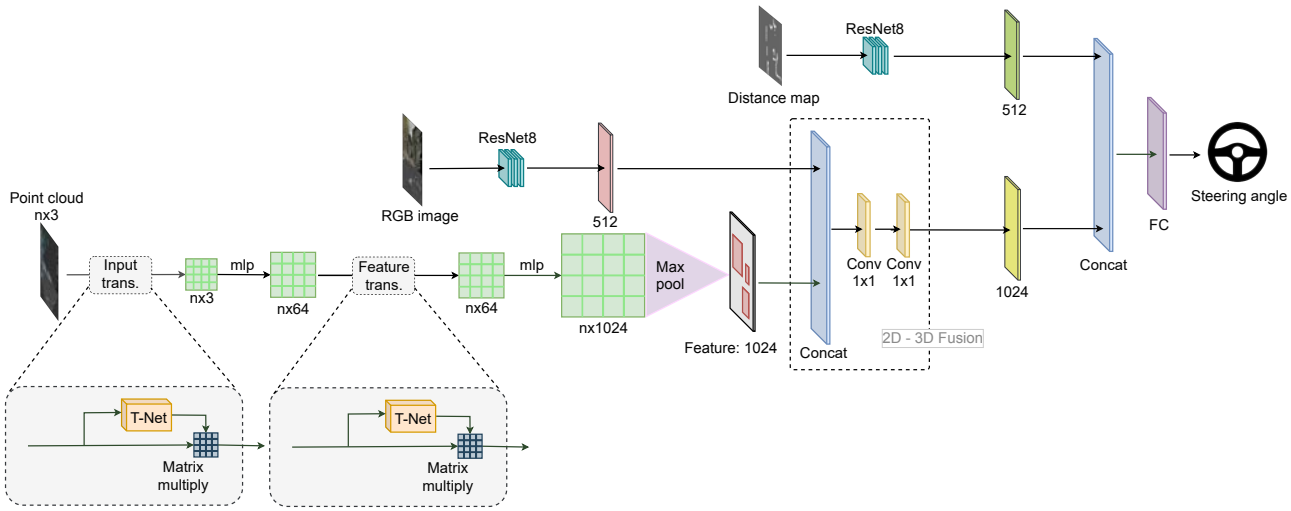


Fig. 4. An overview of our NMFNet architecture. The network consists of three branches: The first branch learns the depth features from the distance map. The second branch extracts the features from RGB images, and the third branch handles the point cloud data. We then employ the 2D-3D fusion to predict the steering angle.

cloud is expected to provide more geometry information of the environment for the network. However, extracting features from the unordered cloud is not a trivial task since the network needs to be invariant to all permutations of the input set.

To extract the point cloud feature vector, our network has to learn a model that is invariant to input permutation. As motivated by [46], we extract the features from the unordered point cloud by learning a symmetric function on transformed elements. Given an unordered point set $\{x_1, x_2, ..., x_n\}$ with $x_i \in \mathbb{R}^3$, we can define a symmetric function $f : X \rightarrow \mathbb{R}$ that maps a set of unordered points to a feature vector as follow:

$$f(x_1, x_2, ..., x_n) = \delta(\underset{i=1,...,n}{MAX}\{\gamma(x_i)\}) \tag{2}$$

where MAX is a vector max operator that takes n input vectors and returns a new vector of the element-wise maximum; δ and γ are usually presented by neural networks.

In practice, δ and γ function are approximated by an affine transformation matrix with a mini multi-layer preception network (i.e., T-net) and a matrix multiplication operation. Given the $n \times 3$ unordered input points, we apply this transformation twice to learn the geometric feature from the pount cloud: input transformation and feature transformation. The input transformation uses raw point cloud as input and regresses to a 3×3 matrix. It consists of a three multi-layer perceptron network with layer output sizes are 64, 128, 1024, respectively. The output matrix is first initialized as an identity matrix and all layers have ReLU and batch normalization (except the last layer). We then feed the output of the first transformation to the second transformation network which has the same architecture and generates a 64×64 matrix as output. This matrix is also initialized as an identity and presents the learned features from the point cloud.

C. Multimodal Fusion

Given the features from the point cloud branch and the RGB image branch, we first do an early fusion by concatenating the features extracted from the input cloud with the deep features extracted from the RGB image. The intuition is that since both the RGB image and the point cloud are captured using a camera with the same viewpoint, fusing their features will let the robot aware of both visual information from RGB image and geometry clue from point cloud data. This concatenated feature is then fed through two 1×1 convolutional layers. Finally, we combine the features from 2D-3D fusion with the extracted features from the distance map branch. The steering angle is predicted from a final fully connected layer keeping all the features from the multimodal fusion network.

D. Training

To train an end-to-end navigation network, two popular approaches are used: classification loss [32] and regression loss [29]. Methods use classification loss first bin the ground-truth steering angles into small and discrete groups, then learn possible controls as a classification problem using a softmax loss function. In practice, we have observed the instability during training due to the highly imbalanced statistic in the dataset. Therefore, we employ the regression loss to train the network end-to-end using the mean squared error (MSE) L_2 loss function between the ground-truth human actuated control, y_i, and the predicted control from the network $\hat{y}$:

$$L(y, \hat{y}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{3}$$

Apart from training with normal data, we also employ the training method using domain randomisation [47]. As shown in [47], this simple technique can effectively improve the generalization of the network when only simulation data are available for training.

V. EXPERIMENTS

A. Dataset

Data Collection Unlike the traditional autonomous navigation problem for autonomous car or UAVs that can collect data in real-world setting [28]–[30], it is not a trivial task to build complex environments such as collapsed cities or a collapsed houses in real life. Therefore, we create the simulation models of these environments in Gazebo and collect the visual data from simulation. In particular, we collect the data from three types of complex environment:

- Collapsed house: The house suffered from an accident or a disaster (e.g. an earthquake) with random objects on the ground.
- Collapsed city: Similar to the collapsed house, but for the outdoor environment. In this scenario, the road has many debris from the collapsed house/wall.
- Natural cave: A long natural tunnel in a poor brightness condition with irregular geological structures.

To build the simulation environments, we first create the 3D model of normal daily objects in indoor and outdoor environments (e.g. beds, tables, lamps, computers, tools, trees, cars, rocks, etc.), including broken objects (e.g. broken vases, broken dishes, and debris). These objects are then manually chosen and placed in each environment to create the entire simulated environment.

For each environment, we use a mobile robot model equipped with a laser sensor and a depth camera mounted on top of the robot to collect the visual data. The robot is controlled manually to navigate around each environment. We collect the visual data when the robot is moving. All the visual data $(\mathcal{D}, \mathcal{P}, \mathcal{I})$ are synchronized with a current steering signal of the robot at each timestamp.

Data Statistic In particular, we create 539 3D object models to build the complex environments. These objects are used to build 30 environments in total (i.e., 10 instances for each environment). In average, the collapsed house environments are built with approximately 130 objects in an area of $400m^2$. The collapsed city has 275 objects and spread in $3,000m^2$ while the natural cave environments are built with 60 objects in approximately $4,000m^2$ area. We manually control the robot in 40 hours to collect the data.

In total, we collect around $40,000$ visual data triples $(\mathcal{D}, \mathcal{P}, \mathcal{I})$ for each environment type, resulting a large-scale dataset with $120,000$ records of synchronized RGB image, point cloud, laser distance map, and ground-truth steering angle. Around 45% of the dataset are collected when we use domain randomisation by apply random texture to the environments (Fig. 5). For each environment, we use 70% data for training and 30% data for testing. All the 3D environments and our dataset will be made publicly available to encourage further research.

B. Implementation

We implement our network using Tensorflow framework [48]. The network is optimized using stochastic gradient descent with the fix 0.01 learning rate and 0.9 momentum. The input RGB image and distance map size are (480×640) and (320×640), respectively, while the point cloud data are sampled to $20,480$ points. We train the network with the batch size of 8 and the training time is approximately 30 hours on an NVIDIA 2080 GPU.

Fig. 5. Different robot's views in our simulation of complex environments: collapsed city, natural cave, and collapsed house. **Top row:** RGB images from robot viewpoint in normal setting. **Other rows:** The same viewpoint when applying domain randomisation to the simulated environments.

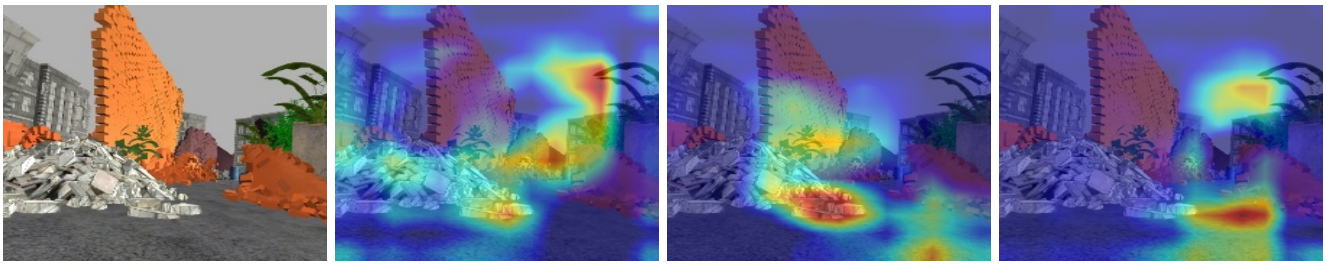
C. Baseline

We compare our method with the following recent state-of-the-art methods in autonomous navigation: DroNet [29], VariationNet [31]. We also present the result for Inception-V3 to serve as a baseline of deep architecture. We note that DroNet uses ResNet8 as the backbone to predict the steering angle and collision probability from RGB input. Since our dataset does not have collision ground-truth, we disable the collision probability branch of DroNet, and only use the regression branch to predict the result. Intuitively, DroNet architecture is similar to our RGB branch. All the baselines are trained with the data from domain randomisation. We show the results of our NMFNet under two settings: with domain randomisation (NMFNet with DR), and without using training data from domain randomisation (NMFNet without DR).

D. Results

Table I summarizes the regression results using Root Mean Square Error (RMSE) of our NMFNet and other state-of-the-art methods. From the table, we can see that our NMFNet outperforms other methods by a significant margin. In particular, our NMFNet trained with domain randomisation data achieves 0.389 RMSE which is a clear improvement over other methods using only RGB images such as DroNet [29]. This also confirms that using multi visual modalities input as in our fusion network is the key to successfully navigate in complex environments.

Within three complex environment types, we observe that the RMSE of the collapsed house results are higher than the collapsed city and the natural cave. A possible reason is that the collapsed house environment is much smaller than others while having more objects. Therefore, it would be



(a) Input Image (b) RGB (c) RGB + Point Cloud (d) Fusion

Fig. 6. The activation map when different modalities are used to train the network. From left to right: **(a)** The input RGB image. **(b)** The activation map of the network uses only the RGB image as input. **(c)** The activation map of the network uses both RGB image and point cloud as input. **(d)** The activation map of the network uses fusion input (both RGB, point cloud and distance map). Overall, the network uses fusion input with all three modalities produces the most reliable heat map for navigation.

TABLE I

RMSE SCORES ON THE TEST SET

	Input	House	City	Cave	Average
DroNet [29]	RGB	0.938	0.664	0.666	0.756
Inception-V3 [49]	RGB	1.245	1.115	1.121	1.16
VariationNet [31]	RGB	1.510	1.290	1.507	1.436
NMFNet without DR	Fusion	0.482	**0.365**	0.367	0.405
NMFNet with DR	Fusion	**0.480**	**0.365**	**0.321**	**0.389**

TABLE II

RMSE SCORES OF NETWORKS USING DIFFERENT MODALITY

Input	House	City	Cave	Average
RGB	0.938	0.664	0.666	0.756
Distance Map	0.730	0.579	0.602	0.637
Point Cloud	-	-	-	-
RGB + Point Cloud	0.718	0.499	0.783	0.667
RGB + Distance Map	0.568	0.452	0.503	0.508
Distance Map + Point Cloud	0.631	0.474	0.592	0.566
Fusion	0.480	0.365	0.321	0.389

more challenging for the robot to navigate in the collapsed house without colliding with the objects. From Table I, we also notice that by employing domain randomisation, our NMFNet with DR shows a good improvement in comparison with the setting without domain randomisation (NMFNet without DR). On the other hand, the setup with VariationNet approach [31] has the highest error in all three complex environments while the Inception-V3 shows reasonable results.

E. Contribution of Visual Modalities

To understand the contribution of each modality to the results, we perform the following experiment: We first train a network that uses only a single modality (either RGB, distance map, or point cloud) as the input. Technically, each network in this experiment is a branch of our NMFNet. We then perform similar experiments using networks with two branches (i.e., RGB + distance map, RGB + point cloud, and distance map + point cloud) as the input. All the networks use the training set with extra data from domain randomisation.

Table II shows the RMSE scores when different modalities are used to train the system. We first notice that the network that uses only point cloud data as the input does not converge. This confirms that learning meaningful features from point cloud data is challenging, especially in complex environments. On the other hand, we achieve a surprisingly good result when the distance map modality is used as the input. The other combinations between two modalities show reasonable accuracy, however, we achieve the best result when the network is trained end-to-end using the fusion from all three modalities: rgb, distance map from laser camera, and point cloud from the depth camera. To further verify the contribution of each modality, we employ Grad-CAM [50] to visualize the activation map of the network when different modality is used. Fig. 6 shows the qualitative visualization under three input settings: RGB, RGB + point cloud, and fusion. From Fig. 6, we can see that from a same viewpoint, the network that uses fusion data makes the most logical decision since its attention lays on feasible regions for navigation, while other networks trained with only RGB image or RGB + point cloud show more noisy attention.

We also note that the inference time of our NMFNet is approximately 100ms on an NVIDIA 2080 GPU. This allows our method to be used in a wide range of robotic applications. More qualitative results including the deployment of NMFNet on BeetleBot [51] can be found at https://sites.google.com/site/multimodalnavigation/.

VI. CONCLUSIONS AND FUTURE WORK

We propose NMFNet, an end-to-end and real-time deep learning framework for autonomous navigation in complex environments. Our network has three branches and effectively learns the visual fusion input data. Furthermore, we show that the use of mutilmodal sensor data is essential for autonomous navigation in complex environments. The extensively experimental results show that our NMFNet outperforms recent state-of-the-art methods by a fair margin while achieving real-time performance and generalizing well on unseen environments.

Currently, our NMFNet shows limitation on scenarios when the robot has to cross small debris or obstacles. In the future, we would like to quantitatively evaluate and address this problem. Another interesting direction is to combine our method with uncertainty estimation [25] or a goal-driven navigation task [26] for more wide-range of applications.

REFERENCES

[1] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end

[2] A. Nguyen, T.-T. Do, I. Reid, D. G. Caldwell, and N. G. Tsagarakis, "V2cnet: A deep learning framework to translate videos to commands for robotic manipulation," *arXiv preprint arXiv:1903.10869*, 2019.

[3] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, "Deeplanes: End-to-end lane position estimation using deep neural networks," in *CVPR*, 2016.

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars." *CoRR*, 2016.

[5] M. Monajjemi, S. Mohaimenianpour, and R. Vaughan, "Uav, come to me: End-to-end, multi-scale situated hri with an uninstrumented human and a distant uav," in *IROS*, 2016.

[6] T. Howard, C. Green, D. Ferguson, and A. Kelly, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *Journal of Field Robotics*, 2008.

[7] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv:1611.03673*, 2017.

[8] "Darpa subterranean challenge," https://www.subtchallenge.com/.

[9] M. Kam, X. Zhu, and P. Kalata, "Sensor fusion for mobile robot navigation," *Proceedings of the IEEE*, 1997.

[10] Y. Dobrev, S. Flores, and M. Vossiek, "Multi-modal sensor fusion for indoor mobile robot pose estimation," in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2016.

[11] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *IROS*, 2013.

[12] H. Du, W. Wang, C. Xu, R. Xiao, and C. Sun, "Real-time onboard 3d state estimation of an unmanned aerial vehicle in multi-environments using multi-sensor data fusion," *Sensors*, 2020.

[13] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, and M. Chli, "Gomsf: Graph-optimization based multi-sensor fusion for robust uav pose estimation," in *ICRA*, 2018.

[14] O. Mees, A. Eitel, and W. Burgard, "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments," in *IROS*, 2016.

[15] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[16] A. Valada, J. Vertens, A. Dhall, and W. Burgard, "Adapnet: Adaptive semantic segmentation in adverse environmental conditions," in *ICRA*, 2017.

[17] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *ICML*, 2016.

[18] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *ICRA*, 2013.

[19] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *CoRR*, 2016.

[20] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *RA-L*, 2015.

[21] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *CVPR*, 2017.

[22] A. Nguyen, D. Kundrat, G. Dagnino, W. Chi, M. E. Abdelaziz, Y. Guo, Y. Ma, T. M. Kwok, C. Riga, and G.-Z. Yang, "End-to-end real-time catheter segmentation with optical flow-guided warping during endovascular intervention," *arXiv preprint arXiv:2006.09117*, 2020.

[23] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," in *ICCV*, 2017.

[24] A. Nguyen, Q. D. Tran, T.-T. Do, I. Reid, D. G. Caldwell, and N. G. Tsagarakis, "Object captioning and retrieval with natural language," in *CVPRW*, 2019.

motion planning for autonomous ground robots," in *ICRA*, 2017.

[25] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," in *RSS*, 2017.

[26] W. Gao, D. Hsu, W. S. Lee, S. Shen, and K. Subramanian, "Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation," *arXiv:1710.05627*, 2017.

[27] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.

[28] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness," in *IROS*, 2017.

[29] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *RA-L*, 2018.

[30] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *IROS*, 2017.

[31] A. Amini, W. Schwarting, G. Rosman, B. Araki, S. Karaman, and D. Rus, "Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing," in *IROS*, 2018.

[32] A. Amini, L. Paull, T. Balch, S. Karaman, and D. Rus, "Learning steering bounds for parallel autonomous systems," in *ICRA*, 2018.

[33] S. K. Alexander Amini, Guy Rosman and D. Rus, "Variational end-to-end navigation and localization," *arXiv:1811.10119v2*, 2019.

[34] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *IROS*, 2017.

[35] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.

[36] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, 2015.

[37] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA*, 2017.

[38] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning," *arXiv:1812.00971*, 2018.

[39] J.-B. Delbrouck and S. Dupont, "Object-oriented targets for visual navigation using rich semantic representations," *arXiv:1811.09178*, 2018.

[40] M. Muller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "Teaching uavs to race: End-to-end regression of agile controls in simulation," in *ECCV*, 2018.

[41] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv:1611.04201*, 2016.

[42] M. Mancini, G. Costante, P. Valigi, T. A. Ciarfuglia, J. Delmerico, and D. Scaramuzza, "Toward domain independence for learning-based monocular depth estimation," *RA-L*, 2017.

[43] O. Andersson, M. Wzorek, and P. Doherty, "Deep learning quadcopter control via risk-aware active learning," in *AAAI*, 2017.

[44] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *arXiv:1611.01779*, 2016.

[45] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "Plato: Policy learning using adaptive trajectory optimization," in *ICRA*, 2017.

[46] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[47] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," *arXiv:1707.02267*, 2017.

[48] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *Symposium on Operating Systems Design and Implementation*, 2016.

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2015.

[50] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017.

[51] A. Nguyen, E. Tjiputra, and Q. D. Tran, "Beetlebot: A multi-purpose ai-driven mobile robot for realistic environments," in *Proceedings of UKRAS20 Conference: Robots into the real world*, 2020.