Towards Online Waypoint Generation for a Quadrotor Using Enhanced Monocular Depth Estimation

Arunava Basu[1] Hirunima Jayasekara[1] Paras Savnani[1] Naitri Rajyaguru[1]

[1]University of Maryland

Abstract

The autonomous navigation in unknown and unstructured environments is very difficult. The depth estimation using monocular camera relies on training set and prediction of waypoints is not feasible with scaled depth. To overcome this, dataset is collected using manually flying quadcopter in AirSim to have self-supervised network for waypoint prediction.Hence,in this paper, we present a monocular vision based waypoint prediction system for quadrotor where estimated depth is being used for navigation.[1]

1. Introduction

Exploring and navigating autonomously in uneven and unstructured environments is an unsolved problem because flights have a high risk of collision and have a battery bottleneck due to limited storage onboard. Also due to environmental uncertainites, an intelligent autonomous aerial system is essential to navigate these environments. Current literature on autonomous flights focuses on mapping the environment first then navigating them. Our approach doesn't assume that the environment is mapped, we take in real time images and try to navigate the environment. Also having a dense depth map of the real-world can be very useful in applications including navigation and scene understanding, augmented reality [1].

Based on our experimental analysis of existing architectures and training strategies we set out with the design goal to develop a simpler architecture that makes training and future modifications easier. Our architecture is a modification of the DenseDepth to compute depth and this feeds into a novel parallized 2-way waypoint prediction network. We rely on transfer learning of the Encoder-Decoder weights for the depth network and train the WPNet from Ground zero. A novel thing about this architecture is that we can train each module seperately and it does not affect the performance of the network. This modular approach is advantageous if in future we want to substitute any network or improve the performance of any submodule.

Our approach uses Depth Estimation with Multi-Task Regression-based Learning to individually learn the position of waypoints in X, Y, Z coordinates within the scene in addition to learning the qw, qz, qy, qz quaternions. The flight controller can change position and speed in real time to perform obstacle avoidance. Also it uses the inertial and visual information to predict waypoints so its a very robust approach in GPS-denied environments.

2. Related Work

The problem of depth estimation from RGB image is a difficult problem as estimated depth does not provide real world coordinate information. Also, current work for navigation can be divided into two parts that is path planning where prior information of the environment is provided or other is simultaneous localization and mapping [5]. Within existing literature, the end-to-end network for navigation of quadcopters using depth estimation is not vastly explored in the literature. Whereas navigation using monocular camera is accomplished using Multi-Task Regression Based end-to-end learning where a waypoint is predicted using RGB images along with ground truth location of drone in AirSim software [9].

Eventhough estimating the depth from a 2D image is an ill-posed problem, there are many work done in the literature [1, 3, 14] exploring the hypothesis of predicting the relative depth by sampling on the training data. Some work was carried out with unsupervised generation of depth maps with the availability of stereo images [6]. Due to unavailability of a waypoint prediction dataset which contains depth data, the utilization of transfer learning methods are essential for a hybrid system. Alhashim *et al.* [1] proposed densedepth part of which utilized pretrained densenet169 [7] as feature extractor for the encoder.

3. Approach

Autonomous Navigation for quadcopters is quite challenging due to the resource constrains and the fast response

[1]https://github.com/savnani5/WPNet

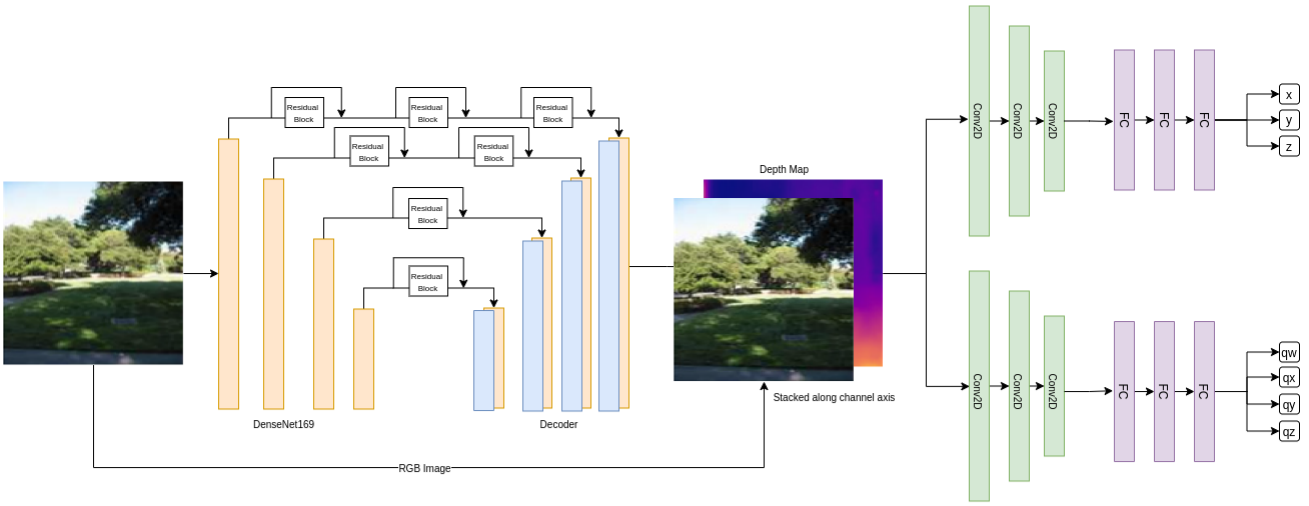


Figure 1. Overview of the system. Full system consists of two modules, namely depth network which produces depth map from a single 2D images and a control module to predict the next best routine for the quadrotor.

time required. While various methodologies have been proposed by far with deep learning based [2, 8], reinforcement learning based control [10, 12] and others. The challenges of the mentioned methodologies are low reliability, and high computational and sensory power. For better optimization, monocular camera is being taken into consideration by Amer et al. [2].

3.1. Depth Network

Estimation of depth from a single image is a simple task for humans, but it is difficult for computational models to compute with high accuracy and low resources. Monocular Depth Estimation is the task of predicting the depth map from a single RGB image. The standard way of approaching this task is to use a large set of RGB images with their corresponding depth pair image and try to model it as a pixel-level continuous regression problem. For our work we initially explored two different lines of models to see the best feasible approach. The two were Transformers, and convolution based architectures.

Vision transformers for dense prediction [13] was explored, specifically dense vision transformers and it was noted that it shows better global coherence and produces finer depth maps than some of the state of the art convolution networks. We were successfully However transformer based models require large amounts of data. The baseline model, DPT was trained with an affine-invariant loss and therefore cannot be directly fine-tuned on the smaller datasets such as KITTI and NYUv2.

Shifting our focus to convolution network based architectures, we considered DenseDepth [1] as our baseline model. This model works on a encoder decoder based architecture and focuses more on design choices which leads to high quality depth maps. The encoder of the network is DenseNet-169 pre-trained on ImageNet. The output of the encoder is then fed into a series of bilinear upsampling layers. These upsampling layers and the corresponding skip-connections forms the decoder. The decoder is hence composed of basic blocks of convolutional layers applied on the concatenation of the bilinear upsampling of the previous block with the skip connection data of the same spatial size after upsampling.

In order to improve the baseline we focused on four major architectural changes :

- Addition of residual blocks in the skip connections: The point of this was to hopefully further emphasize on feature maps from previous stages in the encoder. By adding residual blocks, we hope that the network will learn in which block to follow an identity path and in which to traverse through the convolution block. The overall architecture with residual blocks is illustrated in Fig. 1.

- Adding Optical Flow as an Additional Modality: As the each frame of the dataset has a correlation between it's neighbouring frames similar to that of a video stream, we can deploy specific processing methods inorder to stabilize the depth map progression from one frame to another while extracting more information from previous frames.

- Replacing DenseNet-169 with ResNet50: We hope that replacing the existing architecture with a residual based features extractor such as ResNet50 will improve the results [4].

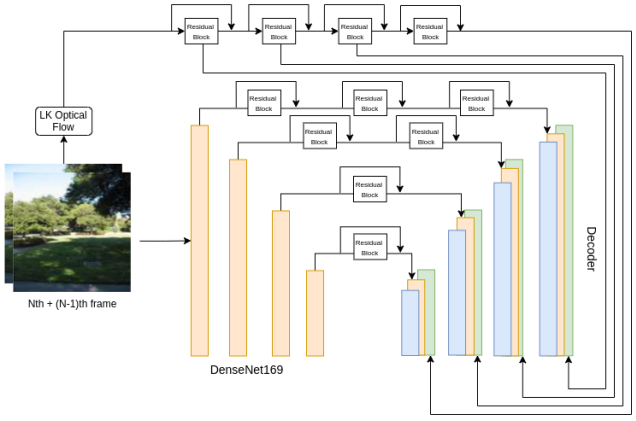


Figure 2. Adding optical flow as a modality for the depth network

All the modified architectures were trained based on hybrid loss defined by Eq 1

$$Loss_{total}(y, y') = 0.1 \times Loss_{depth} + L_{grad} + Loss_{DSSIM} \tag{1}$$

The standalone losses are defined below in Eq. 2, Eq.3 and Eq. 4

$$Loss_{depth}(y, y') = \sum_{i=1}^{n} \|y - y'\| \tag{2}$$

$$Loss_{grad}(y, y') = \sum_{i=1}^{n} \|g_x(y, y')\| + \|g_y(y, y')\| \tag{3}$$

Even though L_{grad} was reported in the original paper, it was not included in the official pytorch implementation. therefore as a support to the community, we made our modifications readily available with the branch we created for the original implementation.

$$\begin{aligned} \text{SSIM}(\text{y}, \text{y}') &= \frac{(2\mu_y\mu_{y'} + C_1)(2\sigma_{yy'} + C_2)}{\left(\mu_y^2 + \mu_{y'}^2 + C_1\right)\left(\sigma_y^2 + \sigma_{y'}^2 + C_2\right)} \\ \text{Loss}_{\text{DSSIM}}(\text{y}, \text{y}') &= \frac{1}{N}\sum_{i=1}^{n} 1 - SSIM(y_n, y'_n) \end{aligned} \tag{4}$$

3.2. WPnet

We develop a WayPoint Prediction Network to predict quadrotor waypoints for the obstacle avoidance task. The objective is to identify the clear flight areas and predict the flight behaviour while exploring an unknown environment. Our approach tries to utilize the depth information about the nearby objects and combine it with the rich input information from original RGB image to make waypoint predictions. This network is a multi-task regressive model which can regress cartesian coordinates and quaternion coordinates for the quadrotor. Furthermore quaternions are more preferred for the regression task as compared to Euler angles because they don't have singularities.

For the input of the network we concatenate RGB images with the depth map at the channel dimension as illustrated in Fig. 1. We parallize this network into two separate Convolutional feature extractors followed by linear layers to regress the waypoints. We tried using a single network for WPNet but due to different scaling of the cartesian and quaternion waypoints the gradient was not backpropogating properly to the input, therefore we developed two parallel architectures.

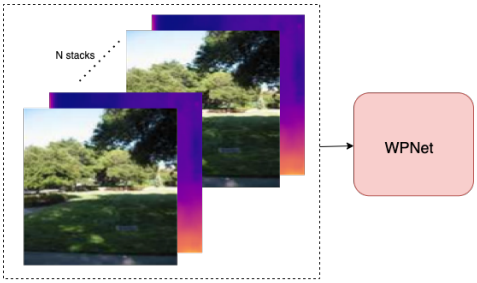


Figure 3. Multiple Frame Stacking Across Channel Dimension for WPNet

We tried multiple frame stacking across the channel dimension as shown in Fig. 3 and tried to predict n waypoints in future but the limited GPU storage capacity constrained the network from learning. But this will be tried in a future iteration of this network. We use per waypoint Mean squared error(4) loss for the regression task. Currently we use Airsim's inbuilt controller to manipulate the drone, but we plan to add a custom LQR controller on top of the predicted trajectories to simulate them in airsim.

Table 1. WPNet parameters

Layer	Size
Conv2D + Relu	(1×16×3)
Max Pooling	(2,2)
Conv2D + Relu	(16×32×3)
Max Pooling	(2×2)
Conv2D + Relu	(32×64×3)
Max Pooling	(2×2)
Linear	(73600×500)
Dropout	0.2
Linear	(500×100)
Dropout	0.2
Linear	(100×20)
Linear	(20×3)
Linear	(20×4)

The waypoint prediction implementation details are given in Table.1. And the rsme loss used in the network is shown in Eq. 5

$$Loss_{MSE} = \sum_{i=1}^{n} \|x - x'\|^2 \tag{5}$$

4. Experiments & Results

4.1. WPnet

We trained this network on Custom Regression dataset [9] of 45k samples for 30 epochs with a mini-batch size of 16 and trained it on NVIDIA RTX 2060 with 6GB of Memory. Batchsize constraint is due to the NVIDIA graphics card. We used Adam optimizer with a learning rate of 0.001 and weight decay of 1e-6.

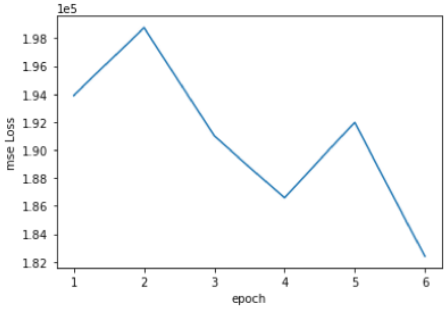


Figure 4. Mean Squared Error for depth input

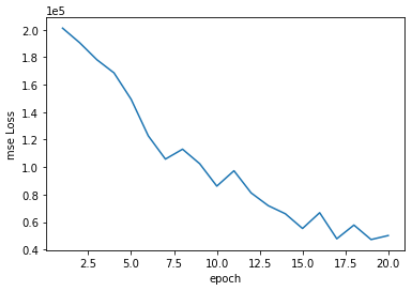


Figure 5. Mean Squared Error for RGBD input

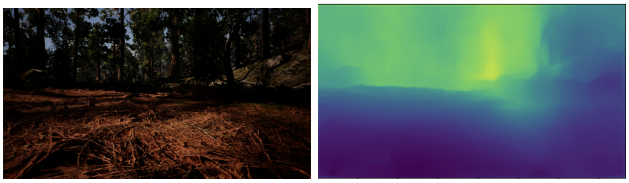
4.1.1 Issues Faced

- Network Convergence is a predominant issue for the WPNet. Primary reason would be that the single image infomation density (low resoluion simulation images) is not enough for the network to learn the regression waypoints.
- The custom multi-regression dataset does not have the ground truth depth images. So we cannot perform end to end learning with the combined model.
- Current Literature does not have a standard metric to evaluate the Waypoint prediction trajectories and therfore we donot have a baseline MSE loss to compare our data with. Although we can genarate collision free distance moved as a metric to determine the perfomance of the algorithm.

4.2. Depth Network

We used the PyTorch library for the development of the model. We trained the network for 287,500 iterations with a batch size of 4 on a NVIDIA V100 with 16G of memory on a 50K subset of the NYUv2 dataset [11] provided by the authors of Densedepth [1]. The optimizer used was Adam with a learning rate of 0.0001 with a exponential weight decay factor of 0.1 per epoch.

After training, the inference results are shown below. We can see that the depth map produced has a decent about of global coherence (for example, the forest floor shown in the image).



(a) Sample Frame (b) Predicted Depth map

Figure 6. Example Output from Depth Network

For evaluating the depth network we use six quantitative metrics provided in [1]. These metrics are defined below. root mean sqaured error(rmse) :

$$\sqrt{\frac{1}{n}\sum_{p}^{n}(y_p - \hat{y}_p)^2} \tag{6}$$

The rmse loss curve between the ground truth and the prediction can be seen in Fig.7. average relative error(rel) :

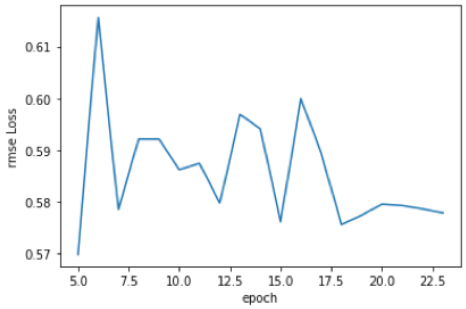
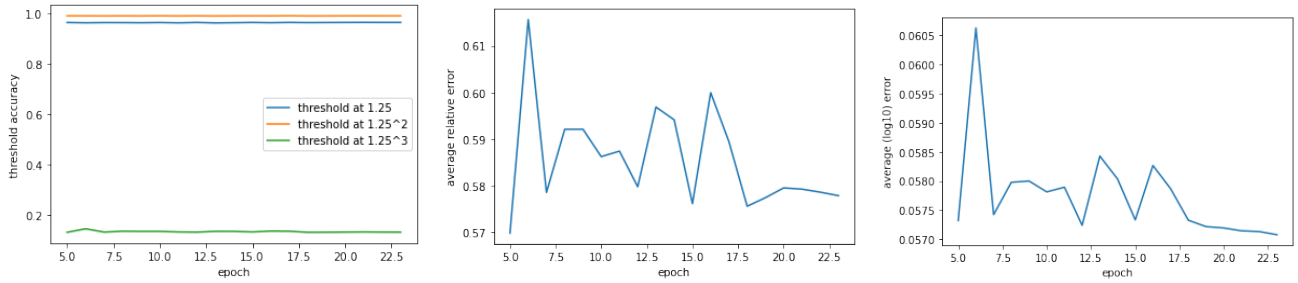


Figure 7. Root Mean Squared Error between Ground Truth and the Prediction for the Depth Network



(a) Threshold accuracy for threshold values at 1.25,1.25^2,1.25^3

(b) Average relative error

(c) Average (log_{10}) error:

Figure 8. Other Evaluation Metrics for Depth Network

$$\frac{1}{n}\sum_{p}^{n}\frac{|y_p - \hat{y}_p|}{y} \tag{7}$$

average log10 error :

$$\frac{1}{n}\sum_{p}^{n}|\log_{10}(y_p) - \log_{10}(\hat{y}_p)| \tag{8}$$

threshold accuracy (δ_i) : % of y_p s.t. $\max\left(\frac{y_p}{\hat{y}_p}, \frac{y_p}{y_p}\right) = \delta < thr$ for $thr = 1.25, 1.25^2, 1.25^3$;

The other metrics are plotted in Fig.8. In the figure illustrating the rmse graph, we can see that with just about 20 epochs we are able to reach the baseline rmse of the weights provided. With more training we are optimistic that we will be able to reach a lower rmse value.

4.2.1 Issues Faced

We faced a number of challenges in trying to beat the baseline of 0.465 rmse provided by the authors of [1].

- The baseline weights in Keras for the original network yielded a rmse of 0.55 on the NYUv2 test set. When the weights were converted and used in the equivalent PyTorch baseline model, the rmse further dropped to 0.588.
- All of the loss functions were not implemented in PyTorch and the training pipeline for the base model has to be heavily reconstructed from the bottom up. Additionally, evaluation methods in PyTorch were not functional and had to be built from scratch.
- Iterations with all the layers unfrozen was taking too long to train on a V100, hence a lot of layers had to be frozen to facilitate training.
- Appreciable training could not be performed on the ResNet50 variant as time taken for 1 epoch on NYUv2 was 23 hours using NVIDIA V100.

Table 2. Summary of Losses

Network	rmse loss	rel	$\log_{10}$
Depth Network	0.5732	0.577	0.0570
	mse-depth	mse-rgbd	mse-rgbd+fs
WPNet	1.89e+5	0.38e+5	-

5. Future Work

For more recent future work we would like to code another evaluation metric for the network output like the collision free distance moved additionally controlling the outputs with a traditional controller like LQR and simulating the results in airsim. We can code this based on the depth information and the check if the predicted waypoint is colliding with the nearby obstacle or not. Also we would like to change the Encoder-Decoder part of the Densedepth to a more parameter efficient network like MobileNet without compromising the performance. Furthermore for a longer timeline we are planning to make this network totally end-to-end where it will also learn the control policies after the waypoint trajectory prediction, although generating such a custom dataset for this approach might be a big problem.

6. Conclusion

In this network we try to improve the performance of the exiting Densedepth pipeline by adding residual blocks to the skip connections of the Encoder-Decoder pipeline. Furthermore we propose a depth estimation based multi-task regression network to learn the waypoints which can navigate indoor/GPS denied environments. This method can also be generalized to previously unseen domains too. Additionally, these techniques are particularly suitable for search and rescue operations in any above the ground environments. Also we can leverage this technique to perform SLAM type Navigation if we using datasets like TUM, Kitti-EigenSplit etc.

References

[1] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*, 2018. 1, 2, 4, 5

[2] Karim Amer, Mohamed Samy, Mahmoud Shaker, and Mohamed ElHelw. Deep convolutional neural network based autonomous drone navigation. In *Thirteenth International Conference on Machine Vision*, volume 11605, page 1160503. International Society for Optics and Photonics, 2021. 2

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021. 1

[4] Keno K Bressem, Lisa C Adams, Christoph Erxleben, Bernd Hamm, Stefan M Niehues, and Janis L Vahldiek. Comparing different deep learning architectures for classification of chest radiographs. *Scientific reports*, 10(1):1–16, 2020. 2

[5] Prateek Chhikara, Rajkumar Tekchandani, Neeraj Kumar, Vinay Chamola, and Mohsen Guizani. Dcnn-ga: a deep neural net architecture for navigation of uav in indoor environment. *IEEE Internet of Things Journal*, 8(6):4448–4460, 2020. 1

[6] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017. 1

[7] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1

[8] Hyeon Kim, Jaechan Cho, Yongchul Jung, Seongjoo Lee, and Yunho Jung. Area-efficient vision-based feature tracker for autonomous hovering of unmanned aerial vehicle. *Electronics*, 9(10):1591, 2020. 2

[9] Bruna G Maciel-Pearson, Samet Akçay, Amir Atapour-Abarghouei, Christopher Holder, and Toby P Breckon. Multi-task regression-based learning for autonomous unmanned aerial vehicle flight control within unstructured outdoor environments. *IEEE Robotics and Automation Letters*, 4(4):4116–4123, 2019. 1, 4

[10] Bruna G Maciel-Pearson, Letizia Marchegiani, Samet Akcay, Amir Atapour-Abarghouei, James Garforth, and Toby P Breckon. Online deep reinforcement learning for autonomous uav navigation and exploration of outdoor environments. *arXiv preprint arXiv:1912.05684*, 2019. 2

[11] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 4

[12] Huy Xuan Pham, Hung Manh La, David Feil-Seifer, and Luan Van Nguyen. Reinforcement learning for autonomous uav navigation using function approximation. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE, 2018. 2

[13] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *CoRR*, abs/2103.13413, 2021. 2

[14] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv preprint arXiv:1907.01341*, 2019. 1