Localization of Unmanned Aerial Vehicles in Corridor Environments using Deep Learning

Ram Prasad Padhy,
Shahzad Ahmad,
Sachin Verma,
Sambit Bakshi, and
Pankaj K. Sa*

National Institute of Technology, Rourkela, India

International Conference of Pattern Recognition (ICPR - 2020)

Submission# 2600

Localization of Unmanned Aerial Vehicles in Corridor Environments using Deep Learning

A DNN based model is proposed for safe localization and subsequent navigation of UAV in indoor corridor environments.

Localization of Unmanned Aerial Vehicles in Corridor Environments using Deep Learning

A DNN based model is proposed for safe localization and subsequent navigation of UAV in indoor corridor environments.

Salient features

- Autonomous UAV navigation in indoor corridors

Localization of Unmanned Aerial Vehicles in Corridor Environments using Deep Learning

A DNN based model is proposed for safe localization and subsequent navigation of UAV in indoor corridor environments.

Salient features

- Autonomous UAV navigation in indoor corridors
- Avoid collision with side and front wall

Localization of Unmanned Aerial Vehicles in Corridor Environments using Deep Learning

A DNN based model is proposed for safe localization and subsequent navigation of UAV in indoor corridor environments.

Salient features

- Autonomous UAV navigation in indoor corridors
- Avoid collision with side and front wall
- Exploits corridor's properties:
  the imaginary central bisector line (CBL)

Contributions

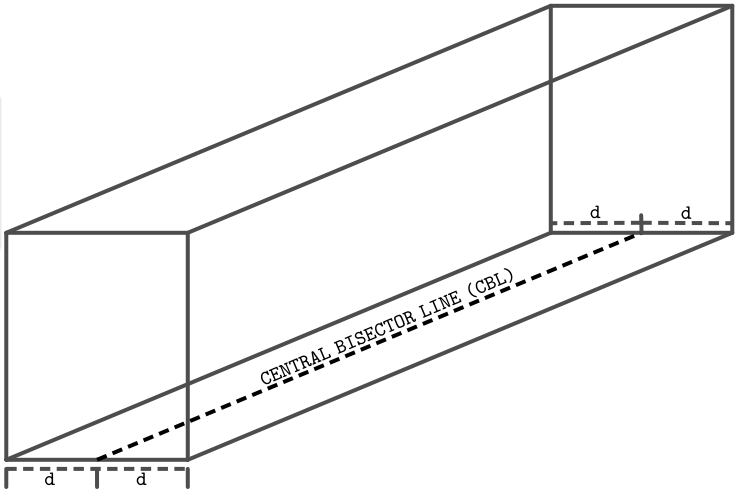
- We propose a method that uses two different DNN models for UAV localization
  1^{st} DNN: UAV's deviation from CBL (translation)
  2^{nd} DNN: UAV's orientation.

- Our algorithm generates necessary control commands to keep the UAV along the CBL and continuously monitors its position to rectify any deviation.

- We propose a new corridor dataset, UAVCorV1 [1], which contains images as captured from the UAV's front camera at different positions in a number of corridors.

Central Bisector Line (CBL)

CBL is an imaginary line that runs at the middle of the corridor's floor and is parallel to the side walls.



Dataset Creation

- 80 different corridors; 59← training and 21← testing.

Dataset Creation

- 80 different corridors; 59← training and 21← testing.
- images are captured at an approx height of 1m from the floor.

Dataset Creation

- 80 different corridors; 59← training and 21← testing.
- images are captured at an approx height of 1m from the floor.
- Three locations (center, left, and right) are selected for one imaginary horizontal line in a corridor ($\perp$ to CBL).

Dataset Creation

- 80 different corridors; 59$\leftarrow$ training and 21$\leftarrow$ testing.
- images are captured at an approx height of 1m from the floor.
- Three locations (center, left, and right) are selected for one imaginary horizontal line in a corridor ($\perp$ to CBL).
- At each location the UAV is oriented towards center, left, and right w.r.t. itself.
- Hence, we have nine images for a given horizontal line.

Dataset Creation

- 80 different corridors; 59← training and 21← testing.
- images are captured at an approx height of 1m from the floor.
- Three locations (center, left, and right) are selected for one imaginary horizontal line in a corridor ($\perp$ to CBL).
- At each location the UAV is oriented towards center, left, and right w.r.t. itself.
- Hence, we have nine images for a given horizontal line.
- Such horizontal lines are chosen at regular interval along the corridor length and the images thus obtained accommodate all possible scenarios.

Figure 1: 9 different possible alignments over a horizontal line perpendicular to the CBL.

Obtaining CBL

(a) Actual image

(b) Image with markers

(c) CBL on image plane

Figure 2: Process of obtaining the CBL on the image plane using markers.

Dataset Creation

Labeled Data Generation: Translational Shift

- It has been observed that when the UAV is on the left side of the CBL, the red line on the image plane forms an acute angle with the bottom image boundary.
- In case the UAV is on the right side of the CBL, the angle formed is obtuse.
- And when the UAV is at the center, it forms a right angle.

Dataset Creation

Labeled Data Generation: Translational Shift

- It has been observed that when the UAV is on the left side of the CBL, the red line on the image plane forms an acute angle with the bottom image boundary.
- In case the UAV is on the right side of the CBL, the angle formed is obtuse.
- And when the UAV is at the center, it forms a right angle.

- For the corresponding actual image in the dataset, these angles (in radians), are stored as the target values. It may be noted that these angles remain the same irrespective of the UAV tilt.

Dataset Creation

(a) Left side of CBL (b) Right side of CBL (c) On the CBL

Figure 3: Three different positions of the UAV over a horizontal line perpendicular to the CBL.

Dataset Creation (Labeled Data Generation: Orientation)

(a) Aligned with the CBL (b) Left tilted (c) Right tilted

Figure 4: Three different orientations of the UAV, when it is situated on the CBL.

Dataset Creation

For each image, the dataset now contains two target values

- Angle of the CBL (for rectifying the translational deviation)
- Distance of the CBL (for rectifying the rotational deviation).

We have shared this dataset as a public dataset, named UAVCorV1 [1]. This dataset contains 35000 training and 600 testing images for angle, and 21000 training and 300 testing images for distance, and their corresponding target values.

UAV Navigation

UAV navigation is a two-step process

- Rectifying the translational deviation.

UAV Navigation

UAV navigation is a two-step process

- Rectifying the translational deviation.
- Rectifying the rotational deviation.

UAV Navigation

UAV navigation is a two-step process

- Rectifying the translational deviation.
- Rectifying the rotational deviation.

- Both processes are achieved by processing the images from the UAV front camera through pre-trained DNNs to predict the deviations.

UAV Navigation

UAV navigation is a two-step process

- Rectifying the translational deviation.
- Rectifying the rotational deviation.

- Both processes are achieved by processing the images from the UAV front camera through pre-trained DNNs to predict the deviations.
- Although the architecture of both the networks remains the same, the target output differs; one is for angle in radian and the other is for normalized pixel distance.

Network Structure

- Pre-trained models:
  - AlexNet, VGGNet, InceptionNet, ResNet-50,101,152, DenseNet-161,201.

Network Structure

- Pre-trained models:
  - AlexNet, VGGNet, InceptionNet, ResNet-50,101,152, DenseNet-161,201.
- Dataset: Our custom dataset UAVCorV1

Network Structure

- Pre-trained models:
  - AlexNet, VGGNet, InceptionNet, ResNet-50,101,152, DenseNet-161,201.
- Dataset: Our custom dataset UAVCorV1
- The last fully connected layer of the pre-trained model is replaced by a few convolution layers to decrease the output feature map size.
- At the end of the network, one fully connected layer is appended to obtain 1×1 output.

Network Structure

- Pre-trained models:
  - AlexNet, VGGNet, InceptionNet, ResNet-50,101,152, DenseNet-161,201.
- Dataset: Our custom dataset UAVCorV1
- The last fully connected layer of the pre-trained model is replaced by a few convolution layers to decrease the output feature map size.
- At the end of the network, one fully connected layer is appended to obtain 1×1 output.
- The input image resolution is 320×180.

Network Structure

- Pre-trained models:
  - AlexNet, VGGNet, InceptionNet, ResNet-50,101,152, DenseNet-161,201.
- Dataset: Our custom dataset UAVCorV1
- The last fully connected layer of the pre-trained model is replaced by a few convolution layers to decrease the output feature map size.
- At the end of the network, one fully connected layer is appended to obtain 1×1 output.
- The input image resolution is 320×180.
- The angle lies in the range $[0, \pi]$ and the normalized pixel distance lies in the range of [0, 1].

Network Structure

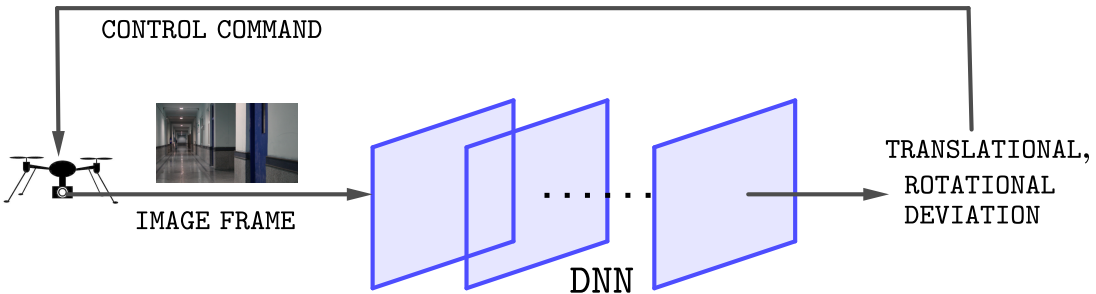


Figure 5: Architectural flow of the proposed DNN based corridor navigation model

Table 1: Various pre-trained models and their augmented convolution layers to train our custom dataset. In the 2D convolution operation, Conv2d(C_{in}, C_{out}, $k \times k$), C_{in}, C_{out}, and k represent the number of input channels, number of output channels, and kernel size respectively.

Input	Pre-trained Models	Augmented Convolution Layers	Augmented last Fully connected layer	Output layer
320 × 180	AlexNet [2]	no augmentation of convolution layer	4096 × 1	1 × 1
	VGG-16 [3]	Conv2d(512, 1024, 1 × 1) Conv2d(1024, 128, 5 × 5) Conv2d(128, 16, 1 × 1)	96 × 1	
	InceptionV3 [4]	**Main:** Conv2d(2048, 1024, 1 × 1) Conv2d(1024, 512, 2 × 2) Conv2d(512, 128, 3 × 3) **Aux:** Conv2d(768, 128, 4 × 4) Conv2d(128, 32, 2 × 2)	**Main:** 256 × 1 **Aux :**640 × 1	
	ResNet-50 [5]	Conv2d(2048, 1024, 1 × 1) Conv2d(1024, 128, 5 × 5) Conv2d(128, 8, 1 × 1)	96 × 1	
	ResNet-101 [5]			
	ResNet-152 [5]			
	DenseNet-201 [6]	Conv2d(1920, 1024, 1 × 1) Conv2d(1024, 128, 5 × 5) Conv2d(128, 16, 1 × 1)	96 × 1	
	DenseNet-161 [6]	Conv2d(2208, 1024, 1 × 1) Conv2d(1024, 128, 5 × 5) Conv2d(128, 16, 1 × 1)		

Loss Function

Mean Absolute Error (MAE) is chosen as the loss function to prevent the learning process of the DNN from being severely affected by large deviations between real and predicted values.

Mean Absolute Error

$$\mathrm{MAE}(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|,$$

where $\hat{Y} = \{\hat{y}_i\}_{i=1}^{n}$ and $Y = \{y_i\}_{i=1}^{n}$ denote the predicted and target values for a mini batch of size n, respectively.

Input Pre-processing

- Before processing, the original BGR pixel values, which lie in the range $[0, 255]$, are first normalized to the range $[0, 1]$.

Input Pre-processing

- Before processing, the original BGR pixel values, which lie in the range $[0, 255]$, are first normalized to the range $[0, 1]$.
- Each of the Blue, Green and Red channels are then normalized by the corresponding mean ($\mu_b = 0.406, \mu_g = 0.456, \mu_r = 0.485$) and standard deviation ($\sigma_b = 0.225, \sigma_g = 0.224, \sigma_r = 0.229$) of the ImageNet classification dataset [7].

Input Pre-processing

- Before processing, the original BGR pixel values, which lie in the range $[0, 255]$, are first normalized to the range $[0, 1]$.
- Each of the Blue, Green and Red channels are then normalized by the corresponding mean ($\mu_b = 0.406, \mu_g = 0.456, \mu_r = 0.485$) and standard deviation ($\sigma_b = 0.225, \sigma_g = 0.224, \sigma_r = 0.229$) of the ImageNet classification dataset [7].
- Prediction of translational deviation requires images from all possible locations of the corridor, however rotational deviation requires images when the UAV is at the center and tilted in all three orientations (left, right or center).

Control Command Generation

Algorithm 1: Control command generation

Input: Image From UAV front camera: *img*
Output: UAV direction: [*pitch*, *roll*, *yaw*]

angle = TrainedModelForAngle(*img*)
if *angle out of bound for continously* 1 *second* **then**
    Land the UAV
if *angle* $\approx 90^\circ$ **then**
    dist = TrainedModelForDistance(*img*)
    if *dist* ≈ 0.5 **then**
        Actuate UAV in Pitch Forward
    else if *dist* $< 0.5 - \delta$ **then**
        Actuate UAV in Yaw Left until *dist* ≈ 0.5
    else
        Actuate UAV in Yaw Right until *dist* ≈ 0.5
else if *angle* $< 90^\circ - \delta$ **then**
    Actuate UAV in Roll Right until *angle* $\approx 90^\circ$
else
    Actuate UAV in Roll Left until *angle* $\approx 90^\circ$
return [*pitch*, *roll*, *yaw*]

System Setup

- We used PyTorch deep learning library to implement our deep networks.

System Setup

- We used PyTorch deep learning library to implement our deep networks.
- Training is performed with GTX 1080 Ti graphics card, which has a memory of 11GB.

System Setup

- We used PyTorch deep learning library to implement our deep networks.
- Training is performed with GTX 1080 Ti graphics card, which has a memory of 11GB.
- ILSVRC [8] pretrained weights are used for initializing the standard state-of-the-art networks.

System Setup

- We used PyTorch deep learning library to implement our deep networks.
- Training is performed with GTX 1080 Ti graphics card, which has a memory of 11GB.
- ILSVRC [8] pretrained weights are used for initializing the standard state-of-the-art networks.
- Weights for other convolution layers are initialized from a normal distribution $\mathcal{N}(0, \sqrt{2/ck^2})$, where k represents the filter size and c is the number of output channels of the convolution operation.

Evaluation Metrics

Evaluation is done on the UAVCorV1 labeled test set, which contains 600 images for angle and 300 images for distance, respectively.

Three different evaluation metrics

$$\text{Mean Squared Error}: \ \text{MSE}(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2,$$

$$\text{Mean Absolute Error}: \ \text{MAE}(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|,$$

$$\text{Mean Relative Error}: \ \text{MRE}(\hat{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i}.$$

where $\hat{Y} = \{\hat{y}_i\}_{i=1}^{n}$ and $Y = \{y_i\}_{i=1}^{n}$ denote the predicted and target values for a mini batch of size n, respectively.

Experimental Results (Translational Deviation)

Quantitative Comparison

Table 2: Evaluation metrics for the prediction of translational deviation of the UAV using 9 different pretrained networks over 600 test images.

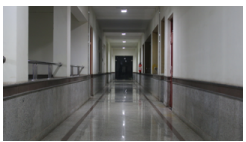
Pre-trained Model	MSE	MAE	MRE
AlexNet	0.21997	1.72831	1.39280
VGG-16	0.47318	2.84597	2.41795
InceptionV3	0.11929	1.44906	1.20959
ResNet-50	0.11253	1.50321	1.18916
ResNet-101	0.11103	1.46875	1.17946
ResNet-152	0.11032	1.41558	1.16529
DenseNet-201	0.12383	1.79144	1.42709
DenseNet-161	**0.05791**	**1.32693**	**1.08712**

Experimental Results (Translational Deviation)

Qualitative Comparison

Figure 6: Qualitative performance evaluation of translational deviation for different corridor locations of National Institute of Technology, Rourkela, India. Ground truth and predicted values are given in degree. GT: Ground truth, PR: Predicted.

TIIR Building



GT: 90.720°
PR: 90.303°

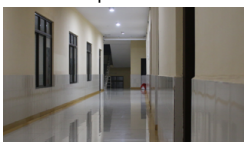
Physics Department

GT: 47.055°
PR: 46.707°

Life Science Department

GT: 90.000°
PR: 91.142°

Computer Science Department



GT: 136.507°
PR: 135.253°

Experimental Results (Rotational Deviation)

Quantitative Comparison

Table 3: Evaluation metrics for the prediction of rotational deviation of the UAV using 9 different pretrained networks over 300 test images.

Pre-trained Model	MSE	MAE	MRE
AlexNet	5.5677	27.0064	54.1467
VGG-16	1.1928	4.6213	13.5329
InceptionV3	0.0687	3.1364	10.4852
ResNet-50	0.0473	2.7485	9.0729
ResNet-101	0.1186	4.2163	14.6806
ResNet-152	0.0662	3.4258	10.8230
DenseNet-201	0.0828	3.6442	12.1421
DenseNet-161	**0.0326**	**2.5060**	**1.557**

Experimental Results (Rotational Deviation)

Qualitative Comparison

Figure 7: Qualitative performance evaluation of rotational deviation for different corridor locations of National Institute of Technology, Rourkela, India. Ground truth and predicted values are in the range $[0, 1]$. GT: Ground truth, PR: Predicted.

TIIR Building

GT: 0.488
PR: 0.491

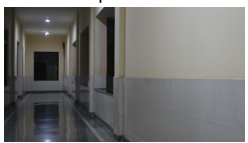
Physics Department

GT: 0.805
PR: 0.808

Life Science Department

GT: 0.547
PR: 0.528

Computer Science Department



GT: 0.167
PR: 0.163

Real World Navigation Experiments

- Parrot AR Drone quadcopter is used for validation purpose.

Real World Navigation Experiments

- Parrot AR Drone quadcopter is used for validation purpose.
- Image transmission delay through ROS is about 0.21s. Our DNN model prediction takes about 0.08s for measuring both translational and rotational deviation simultaneously. Also, the time required for communicating a control command through ROS is about 0.21s.

Real World Navigation Experiments

- Parrot AR Drone quadcopter is used for validation purpose.
- Image transmission delay through ROS is about 0.21s. Our DNN model prediction takes about 0.08s for measuring both translational and rotational deviation simultaneously. Also, the time required for communicating a control command through ROS is about 0.21s.
- Hence, our algorithm can process at most two frames in one second, which is sufficient to generate safe control commands and navigate without collision.

Real World Navigation Experiments

- Parrot AR Drone quadcopter is used for validation purpose.
- Image transmission delay through ROS is about 0.21s. Our DNN model prediction takes about 0.08s for measuring both translational and rotational deviation simultaneously. Also, the time required for communicating a control command through ROS is about 0.21s.
- Hence, our algorithm can process at most two frames in one second, which is sufficient to generate safe control commands and navigate without collision.
- It may be noted that apart from our network prediction, different factors, such as control and state estimation affect the actual UAV flight in real-world scenarios.

Real World Navigation Experiments

- Parrot AR Drone quadcopter is used for validation purpose.
- Image transmission delay through ROS is about 0.21s. Our DNN model prediction takes about 0.08s for measuring both translational and rotational deviation simultaneously. Also, the time required for communicating a control command through ROS is about 0.21s.
- Hence, our algorithm can process at most two frames in one second, which is sufficient to generate safe control commands and navigate without collision.
- It may be noted that apart from our network prediction, different factors, such as control and state estimation affect the actual UAV flight in real-world scenarios.
- We tested our algorithm across 50 trials in 10 different corridors, out of which 43 trails were found to be successful.

Summary

- We have proposed a DNN based UAV localization algorithm, which enables the UAV to navigate safely inside corridors environments.

Summary

- We have proposed a DNN based UAV localization algorithm, which enables the UAV to navigate safely inside corridors environments.
- Unlike previous methods, where the DNN models were designed to predict flight commands directly, our proposed method makes use of an important characteristic of a corridor, the CBL, to generate commands.

Summary

- We have proposed a DNN based UAV localization algorithm, which enables the UAV to navigate safely inside corridors environments.
- Unlike previous methods, where the DNN models were designed to predict flight commands directly, our proposed method makes use of an important characteristic of a corridor, the CBL, to generate commands.
- We designed a single DNN and trained it separately for two tasks; (a) prediction of translational deviation of the UAV from the CBL, and (b) prediction of rotational deviation of the UAV with respect to the CBL.

Summary

- We have proposed a DNN based UAV localization algorithm, which enables the UAV to navigate safely inside corridors environments.
- Unlike previous methods, where the DNN models were designed to predict flight commands directly, our proposed method makes use of an important characteristic of a corridor, the CBL, to generate commands.
- We designed a single DNN and trained it separately for two tasks; (a) prediction of translational deviation of the UAV from the CBL, and (b) prediction of rotational deviation of the UAV with respect to the CBL.
- We also provide a corridor dataset, UAVCorV1, which has the corridor images labeled against translational and rotational deviations.

Summary

- We have proposed a DNN based UAV localization algorithm, which enables the UAV to navigate safely inside corridors environments.
- Unlike previous methods, where the DNN models were designed to predict flight commands directly, our proposed method makes use of an important characteristic of a corridor, the CBL, to generate commands.
- We designed a single DNN and trained it separately for two tasks; (a) prediction of translational deviation of the UAV from the CBL, and (b) prediction of rotational deviation of the UAV with respect to the CBL.
- We also provide a corridor dataset, UAVCorV1, which has the corridor images labeled against translational and rotational deviations.
- The proposed vision based navigation algorithm can be implemented in many industrial fields, such as inspection of disaster-hit buildings, autonomous monitoring of industrial plants, inspection of hazardous places inside buildings, and many more.

Bibliography

[1] "Corridor Dataset for UAV Navigation,"
https://www.nitrkl.ac.in/docs/CS/Database/Windows/NitrUAVCorridorV1.zip.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[6] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, 2016.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

Thank you!!