Using Path Planning Algorithm & Digital Twin Simulator to Collect Synthetic Training Dataset for Drone Autonomous Navigation

Ismail Ryad[1], Marwan Zidan[1], Nadien Rashad[1], Dina Bakr[1], Nadeen Bakr[1], Nada Yehia[1] and Yara Ismail[1], Mohamed AbdelSalam [2], Ashraf Salem[2]
[1]Computer and Systems Engineering Department, Ain Shams University, Cairo, Egypt
[2]Siemens EDA, Cairo, Egypt
{mohamed_abdelsalam, ashraf_salem}@mentor.com

***Abstract*— There is a major challenge in collecting real-world data for training the AI Agents of self-driving Cars, Drones and Automated Guided Vehicles (AGVs). The process is slow and expensive, since data must be reprocessed and correctly labeled before use. Furthermore, it is difficult to collect data for corner cases, especially dangerous scenarios that lead to accidents. Another challenge is the distribution of data that we use to train the model to guarantee optimal results without fitting problems or training meaningless or redundant data. In this paper, we present a novel methodology to use path planning algorithms as a means to generate and label the dataset needed for training AI agents. Our methodology is demonstrated with A* path planning algorithm and Microsoft Airsim Drone Simulator which eases the obtainment of the required data for creating the obstacles grid and provides the tools needed for simulating the drone's movement.**

I. Introduction & Related Work

Recent work has investigated training and testing AI Agents of self-driving cars, drones and Automated Guided Vehicles (AGVs) with synthetically generated data in digital twin simulators, which can be produced in bulk with correct labels. Developers can train their models to be robust to rare events, test their performance under different conditions, and debug entire systems to understand the root cause of a failure and eliminate it.

Drones, have gained a huge success in many fields such as rescue, exploration, environment monitoring, map construction or search. There has been much significant research and related work done in making drones fly autonomously, by generating a collision free path from starting point to destination [1, 2, 3, 4, 5, 6].

It was concluded that for the path planning task when determining the most optimal approach to reach the landing whereabouts and at the same time avoiding obstacles, we must take into consideration all the constraints on the drone. For example, as drones move in 3D environments it has the capability of vertical take-off and landing and high maneuverability which increases the complexity of the path planning task as it requires large processing time to find the collision free paths while moving in cluttered 3D environments more than in 2D environments.

Our work primary focuses on simplifying the environment from the drone point of view, which is done through selecting a fixed height for the drone from the moment it takes off till it reaches its destination, by applying this height constraint we are able to convert 3D environments to 2.5D maps which is the same as 2D map with additional height information for the object, this can be applied in indoor environments such as factories, hospitals or residential compounds where the information of the environment is predetermined in order to create a grid containing the location of each object and its dimensions, which is taken as an input to the path planning algorithm to create the minimal path.

Our contribution is to create an automated, efficient and annotated data collection mechanism that is built on top of the path planning algorithm to collect labelled dataset for training the AI agent model that we use for the drone autonomous navigation. We tested our implementation in two different environments in Microsoft Airsim, which is an open-source digital twin simulator for autonomous driving research [7, 8, 9].The same methodology can be applied to other digital twin simulators which have the capabilities of modeling sensors and scenarios, either proprietary [10] or open source.

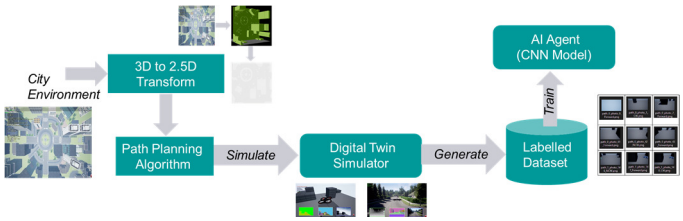


Fig. 1 Our Methodology for Training AI Agent of Drone

This paper is organized as follows, section II explores our proposed methodology including environment setup, path planning algorithm selection and dataset format. Section III describes developed AI agent model for the ego drone. Section IV describes the results of the experiments and discusses different challenges and how we managed to overcome in our implementation and finally section V summarizes the conclusions of the study and directions for future work.

II. Methodology

Fig. 1 shows the overall methodology that we adopted, which starts with environment setup to convert the city into proper format that can be processed by a path planning

algorithm, then selecting the algorithm – in our case study we selected A* Algorithm, and then simulating the results in a digital twin simulator which has the capabilities of modeling different cities, and sensors and animating the behavior of the drone actions, followed by automatically generating labelled data for all possible paths including the optimal path and feeding this dataset to train an AI agent model to finally get the suitable action the drone should take during deployment.

A. *City Environment Setup*

Fig. 2 shows an example of city environment setup in the drone simulator, it shows also the process of converting the environment into 3D grid then converting it to a 2.5D grid that carries the required information needed for our path planning Algorithm.

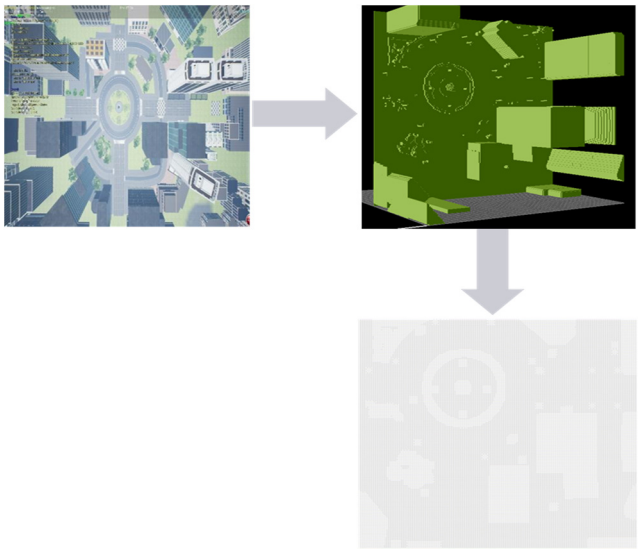


Fig. 2 City Environment Setup & Transformation

First we used *simCreateVoxelGrid()* API in Airsim which requires specifying some parameters.

simCreateVoxelGrid(self, position, x, y, z, res, of)

- position (Vector3r): Global position around which voxel is centered.
- x, y, z (float): Size of each voxel grid dimension.
- res (float): Resolution of voxel grid.
- of (str): Name of output file to save voxel grid

It then outputs a file which contains a 3D representation of the environment. Then we create a class that takes this file and converts it to a 2.5D grid (2d array) in the form of 2 classes: obstacles (1) and non-obstacles (0).

B. *Path Planning Algorithm Selection*

Fig. 3a shows our selection of A* algorithm [11] and how it can process our environment grid construction file, in addition to other specifiers including start, goal, and heuristics and output all possible paths including optimal path. Fig. 3b shows the grid that has the heuristic distance 'H' that helps in prioritizing the nodes to be visited first, adding the cost distance 'C' we get an estimate of the total cost of the plan from the start all the way to the goal. In Fig. 3c, after calculating the total cost and knowing the optimal path that is shown, we automatically start from the current location, go one grid down and stop. Next we send the next waypoint, which is to go one grid cell down again and stop. Next we send a waypoint to go one grid cell right and stop and so forth.

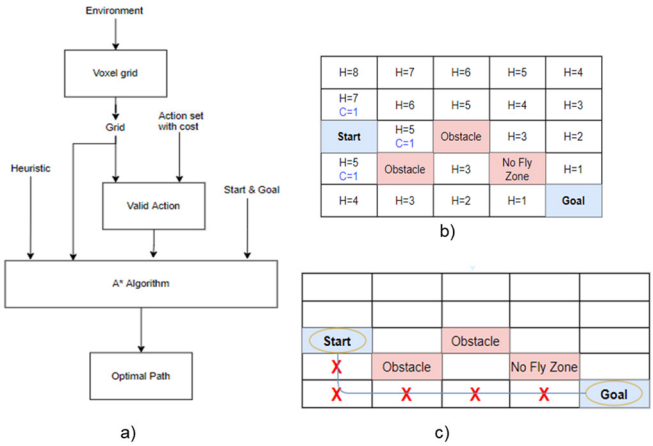


Fig. 3 Using A* Algorithm in Optimal Path Planning

C. *Dataset Format*

For the optimal path and optionally for all possible simulated paths in Airsim the drone takes off and during its movement, we automatically capture an image every time the drone performs an action, and for each image taken the image get labeled automatically, and eventually after the drone reaches its destination, a folder of all captured images is created with each image labeled with its action as Forward[FWD], Clockwise [CW], Counter Clockwise [NCW] as shown in Fig. 4.

Fig. 4 Training images captured in Blocks environment

Different type of cameras could be used to capture images for training model, there are plenty of them supported in AirSim as shown in Fig 5, in our case normal cameras was tested against segmentation camera and found that the model performs better with normal cameras.

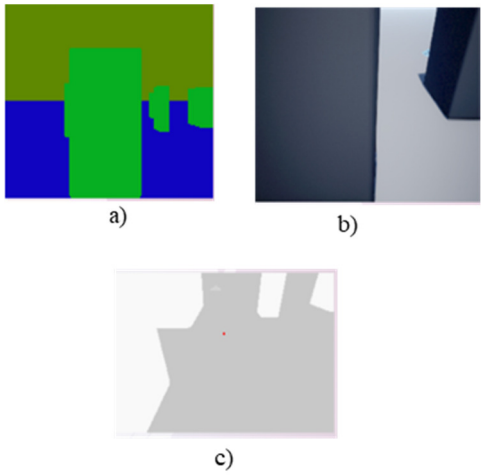


Fig. 5 Different types of cameras. a) segmentation camera, b) normal camera c) infrared camera

Our setup was also to use a normal camera in the front of the drone and the orientation is always the same as the drone itself with angle 30 below to capture not only insights of what is facing the drone but also some insights of what's below the drone which is much better for the model performance.

III. AI AGENT MODEL

After generating the training data set, we developed the drone AI Agent model which is a Convolutional Neural Network (CNN) that attempts to predict the optimal flight command which the drone should execute next. so it is a multi-class classification problem as there are three potential flight commands (Forward, CW, NCW).

The input to the CNN model is an image taken with the drone's camera using the camera setup described in the dataset section above. The model was constructed using Keras [12]. The structure is as shown in Fig 6.

We used input images which are 210*280 RGB images (3 channels) as input to a convolutional layer followed by a max pooling layer, another 2 convolutional layers then max pooling layer, a dropout layer was added to prevent overfitting then flatten layer followed by a dense layer, a batch normalization layer was added to accelerate learning of the CNN model followed by another dropout layer, then another batch normalization layer to boost model performance.

The output layer has 3 nodes (Actions): Forward[FWD], Clockwise [CW], Counter Clockwise[NCW]. The CNN model was trained and deployed in Google Co-lab [13] using Tesla P100-PCIE GPU.

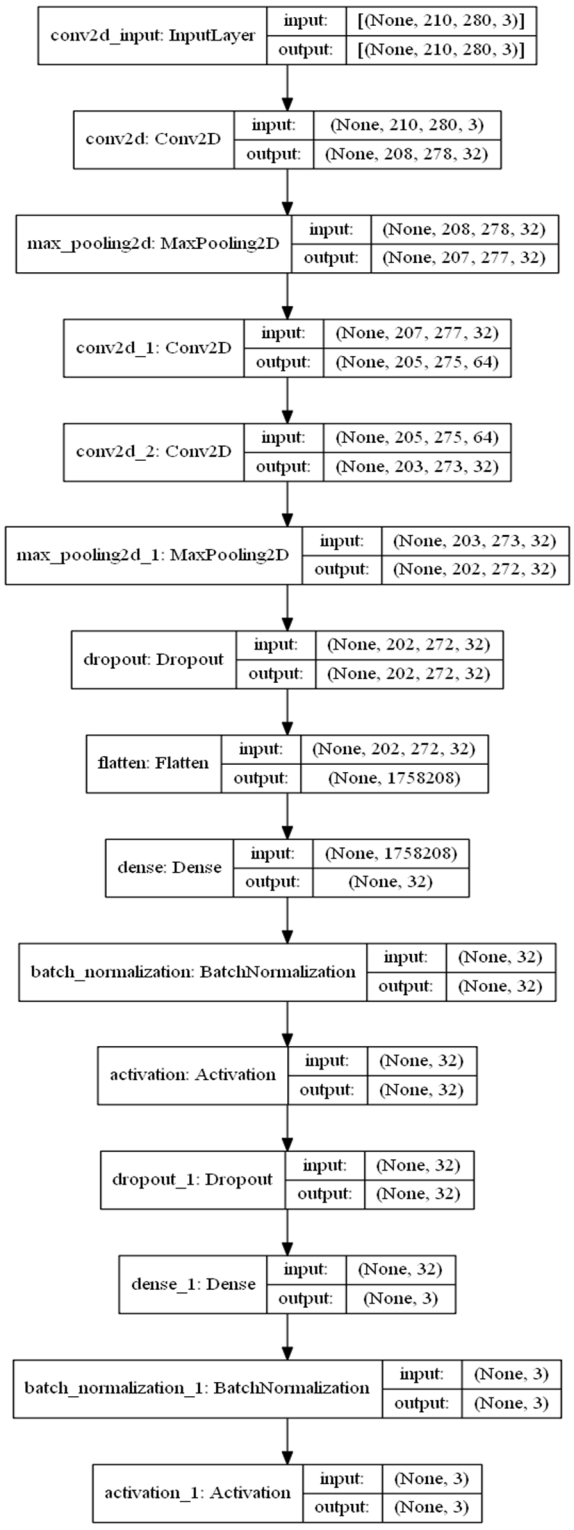


Fig. 6 Drone AI Agent (CNN) Model Architecture

IV. Results & Discussion

In this section we discuss the experimental results, and challenges we faced in collecting the dataset needed for training the AI model.

A. *Safety Margin*

For generalization purposes in our experiments the simulation was done on 2 distinct environments in Airsim (blocks and city) as shown in Fig 7, this separate simulation lead to establishing a reasonable safety margin between the drone and any obstacle. A velocity limit was defined which was selected in a timely manner, also during the testing process on Airsim it was concluded that the z coordinate of the drone and the obstacles were inverted so a negative sign was added to any height while simulating.

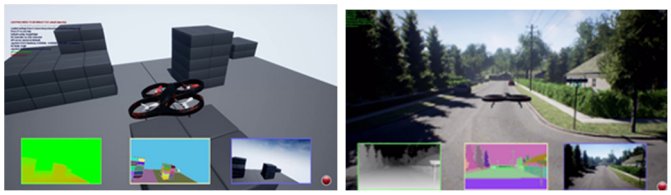


Fig. 7 Environments used in Experiments (Blocks, City)

B. *Drone Orientation*

By default, the main API used in AirSim to move a drone from a position to another is *MoveToPositionAsync()* which takes the new position of the drone (xyz coordinates), but the orientation (pose) of the drone is constant which caused a constraint on the data collection algorithm that uses cameras to capture images, so to solve this problem a function was implemented to set the orientation of the drone whenever it changes.

C. *Camera Instability*

Due to the instability of the drone movement and so as the cameras on it, the cameras by default in Airsim has a fixed orientation which is a problem because it is essential to find the best setting (position, orientation and heading) for the camera used to capture training images. To solve this problem instead of using fixed camera on the drone directly we used camera with *Gimbal* as shown in Fig. 8.

Fig. 8 Illustration of a Gimbal holding a Camera

D. *Data Labeling*

Another challenge, the action set of the path planning algorithm during training our model with the annotated data were initially selected to be global (East, West, North, South) directions, this caused us a problem as in some scenarios the drone was not able to differentiate between south and north, and east and west so the solution was to convert this global action system to a drone-centered action system: Forward[FWD], Clockwise [CW], Counter Clockwise[NCW].

E. *Data Weighting*

One of the major challenges we also encountered was the fact that the dataset collected was imbalanced which means that more than 95% of the images was labeled "forward" and the other 2 labels "CW, NCW" together was less than 5% and this is a natural problem in our case because of the fact that the number of time you would change your direction in a given path is always less than the number of times you would go forward.

To solve this major problem we tried a solution such as under sampling (Fig 9(a)), over sampling (Fig 9(b)) and weight balancing which was the most accurate and achieved high performance, this was implemented in Keras using class weights which is basically telling the model to give more importance to "CW, NCW" labels more than "forward" label as shown in the confusion matrices in Fig 10. We can see that after applying weight balancing the model performance on the 2 labels "CW, NCW" was improved.

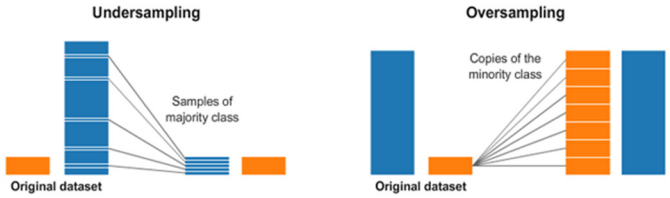
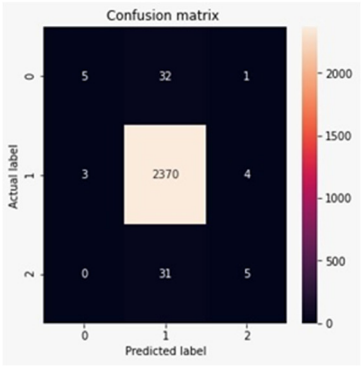
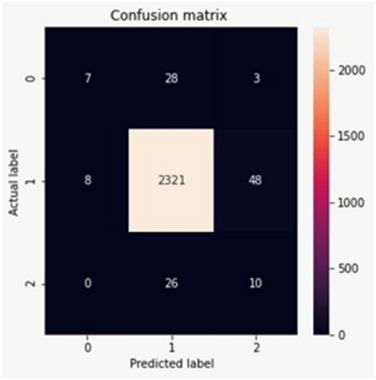


Fig. 9 Sampling Mechanism. a) Undersampling. b) Oversampling



a)

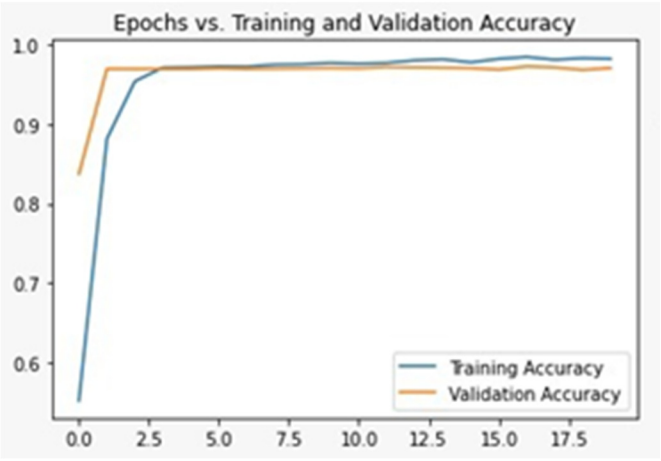


b)

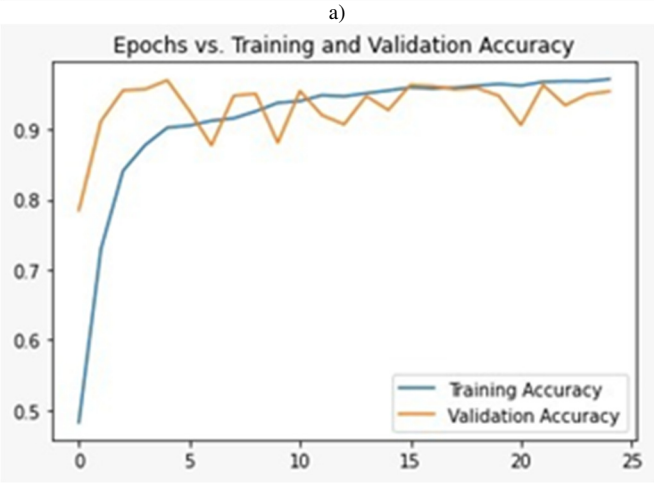
Fig. 10 Weight Balancing. a) Before. b) After.

We have achieved a very good accuracy in the 2 scenarios before and after applying weight balance as shown in Fig 11.

It was obvious that the training process in b) was more rough and the accuracy was a little bit less than in a) but this due to the weight balancing and despite the fact that the accuracy was less but the model behaved much better due to the accurate prediction for the labels "CW, NCW" after applying weight balancing.



a)



b)

Fig. 11. Epochs vs. Training & Validation Accuracy. a) Before. b) After.

V. Conclusions

The analysis done in this paper showed that our methodology of using Path Planning algorithm (e.g. A*) in a drone digital twin simulator (e.g. Microsoft Airsim) can be used to generate synthetic data set for training the AI Agent model of the drone and achieve excellent results. We also presented different challenges we faced in our implementation mainly, camera instability, drone orientation and unbalanced data and how we managed to overcome.

Our future work, includes extending this methodology to other simulators and experiment with different use-cases in area of Automated Guided Vehicles (AGVs) and Self-Driving Cars.

VI. References

[1] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield. Toward low flying autonomous MAV trail navigation using deep neural networks for environmental awareness. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[2] K. Amer, M. Samy, M. Shaker, and M. ElHelw. Deep Convolutional neural network-based autonomous drone navigation. In *arXiv:1905.01657*, 2019.

[3] Max Christl. Vision-Based Autonomous Drone Control using Supervised Learning in Simulation. *Available at: deepai.org/publication/vision*, September 2020.

[4] Ram Prasad, Sachin Verma, Shahzad Ahmad, Suman Kumar, Pankaj Kumar. Deep Neural Network for Autonomous UAV Navigation in Indoor Corridor Environments, In *International Conference on Robotics and Smart Manufacturing (RoSMa),* 2018.

[5] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference*, 2017.

[6] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. Relative camera pose estimation using convolutional neural networks. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, 2017.

[7] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *In Field and service robotics*, pp. 621–635. Springer, 2018.

[8] E. Bondi, D. Dey, A. Kapoor, J. Piavis, S. Shah, F. Fang, B. Dilkina, R. Hannaford, A. Iyer, L. Joppa, M. Tambe. AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs. In *1st ACM SIGCAS Conference*, 2018.

[9] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In *Conference on Robot Learning, CoRL*, 2017.

[10] TASS A Siemens Business, PreScan. Available at: https://tass.plm.automation.siemens.com/prescan.

[11] Akshay Kumar Guruji, Himansh Agarwal, D. K. Parsediya, Time-Efficient A* Algorithm for Robot Path Planning. In *ICIAME,* 2016.

[12] Karan Jakhar, Nishtha Hooda. Big Data Deep Learning Framework using Keras: A Case Study of Pneumonia Prediction, In *ICCCA*, 2018.

[13] T. Carneiro, R. V. Medeiros Da N´oBrega, T. Nepomuceno, G. Bian, V. H. C. De Albuquerque and P. P. R. Filho, Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. In *IEEE Access*, vol. 6, pp. 61677-61685, 2018.