Flying through Gates using a Behavioral Cloning Approach

Erick Rodríguez-Hernández, Juan Irving Vasquez-Gomez, Juan Carlos Herrera-Lozada

Abstract—Drone racing presents a challenge to autonomous micro aerial vehicles (MAV) because usually the track is not known in advance and it is affected by the environment light. In such scenarios, the vehicle has to act quickly depending on the information provided by its sensors. In this work, we want to predict the movement of the drone so that it passes through a gate. Unlike previous approaches where the task is decomposed into perception, estimation, planning, and control, we are proposing a behavioral cloning approach. In this method, a convolutional neural network is trained with the flights of a human operator. So that the output of the trained network is directly the desired MAV state so that it leads the drone through the gate. We have tested the method using a validation set where we obtained a low loss. Furthermore, we have tested the trained network with unseen data obtaining promising results.

I. INTRODUCTION

Drone racing has become a popular sport in recent years. Some competitions are followed by a lot of people in many countries around the world. In these competitions, the discriminant factor is the ability of human pilots. So far, the success of autonomous micro air vehicles is limited to restricted scenarios, due to the complexity of the involved problems [4]. Among the problems are the lack of a track map, variable light conditions [3] and mobile spectators.

In this paper, we are addressing one of the main tasks in autonomous drone racing: fly through a gate using the onboard sensors. See Figure 1. State-of-the-art methods decompose the task into several steps: perception, estimation, planning, and control. In the perception step, the gate is detected using computer vision techniques, for example, in [4] and [3] a convolutional neural network (CNN) is proposed to detect the window. Then, a Kalman Filter [11] is used to eliminate false detections. Next, a trajectory generator and active control are used to pass through the gate [9]. Unlike previous approaches, we propose a behavioral cloning approach were a CNN is trained to directly predict the next state of the micro air vehicle (MAV) using previous flights made by a human expert. See Fig. 2. This approach consolidates the gate detection, estimation, and planning into one single forward pass of the network, so that, the network output will be processed by the onboard controller. To implement the approach, an expert pilot has performed dozens of flights, from them, we have extracted more than 2,000 examples of the desired state that the drone has to follow in order to pass the gate. In addition, we are proposing a custom CNN architecture. Even though the experiments are limited we have observed that the outputs of the network correspond to valid states. This work is proof of concept that behavioral cloning can strengthen the current approaches in order to improve their effectiveness.

The rest of the paper is presented as follows. Section II describes the machine learning technique called behavioral cloning as well as a review of recent approaches. Section III details the proposed approach composed by drone kinematics, learning approach and dataset. Section IV test the approach with a validation and test set. Finally, section V present our conclusions and future research directions.

II. BEHAVIORAL CLONING

The artificial neural networks are a paradigm of machine learning inspired by the neurons of the nervous systems of living beings. It is a system of linked neurons that collaborate with each other to produce an exit stimulus. The connections have numerical weights that adapt according to the experience. In this way, neural networks adapt their selves to an impulse and are able to learn.

An artificial neuron, also known as a simple Perceptron, receives as input a set of values, and performs a linear combination given a set of weights and a bias according
to the following equation,

\[ h = \sum_{i} w_{i} * x_{i} + b \]  

(1)

After the linear combination, \( h \), is passed through an activation function that aims to hide or show the result of the linear combination, namely

\[ \hat{y} = f(h) \]  

(2)

Given the model of an artificial neuron, more complex structures can be realized by interconnecting them in layers. Once the structure of a network is defined, learning is carried out by finding the weights and biases that give the expected output.

A. Deep Learning

Deep learning is a specific type of machine learning. Unlike other learning algorithms, algorithms based on deep learning have the characteristic of constructing hierarchical conceptual structures, for example, in order to recognize a person, the algorithm learns to recognize basic concepts such as edges or color differences, based on them, they create new concepts such as circles or curves, in such a way that concepts such as eyes or faces are learned in higher layers [2]. Another difference of Deep Learning is that it has the capacity to identify the important characteristics to obtain the correct output from the artificial neural network architecture. Although there is no established measure to know when a network is deep, usually proposed architectures use at least two layers for feature extraction by convolutions, followed by fully connected layers for the classification or regression stage.

B. Related work

Baomar and Bentley [1] proposed the creation of an intelligent autopilot system (IAS) as a potential solution to the current problem of automatic flight control systems of being unable to handle flight uncertainties, and the need to construct control models manually. The IAS can learn the piloting skills of expert pilots through imitation. In order to carry out their project it was necessary to collect a large dataset, so they made use of simulators in which expert pilots were put to the test in different flight conditions, including emergency situations in which a conventional autopilot system could fail, like in the cases of strong turbulence or adverse weather conditions.

Another interesting project that also uses deep neural networks is the presented by Kim and Chen [5], in which they teach a drone to fly indoors, due to the fact that the flight presents a great challenge because it can not use GPS to be located in a closed environment, and the drone must make decisions to navigate into corridors and rooms. To generate the dataset, it was necessary to relate the image that the drone is observing with a high level action, this action will be the output of the artificial neural network, and can be solved as a classification problem.

The work presented by Jung, S. et al. [3] develops a drone for autonomous drone racing, in this kind of racing it is well known that a drone is required to fly through the gates quickly without any collision. They propose a convolutional neural network to estimate the center of a gate and after that they use the output to apply a line-of-sight guidance algorithm. Finally, they developed a real-time gate detection networks on an embedded computer that can be used for indoor environment navigation.

Recent work has shown that deep networks provide robust perception capabilities to drones and facilitate safe navigation, as the presented by Kaufmann et al. [4], where they developed a deep learning approach to the autonomous drone racing capable of quickly adapting to new tracks, a convolutional network predicts the poses of the closest gate along with its uncertainty. These predictions are incorporated in an extended Kalman filter to maintain optimal estimates of the gate location. Given the estimated gate poses, they use predictive control to navigate on the track.

Finally, another related work is the presented by Loquercio, et al. [7] where they propose a convolutional neural network, whose purpose is to reliably drive an autonomous drone through the streets of a city. Trained with data collected by cars and bicycles, their system learns from them to follow basic traffic rules, e.g., do not go off the road, and to safely avoid other pedestrians or obstacles. They used a dataset that originally was thought for autonomous cars and they do data augmentation to improve it and implement in a drone.

III. FLYING THROUGH THE GATE

In the proposed approach, we consider two reference frames, an image, an autonomous drone racing capable of quickly adapting to new tracks, a convolutional network predicts the poses of the closest gate along with its uncertainty. These predictions are incorporated in an extended Kalman filter to maintain optimal estimates of the gate location. Given the estimated gate poses, they use predictive control to navigate on the track.

Finally, another related work is the presented by Loquercio, et al. [7] where they propose a convolutional neural network, whose purpose is to reliably drive an autonomous drone through the streets of a city. Trained with data collected by cars and bicycles, their system learns from them to follow basic traffic rules, e.g., do not go off the road, and to safely avoid other pedestrians or obstacles. They used a dataset that originally was thought for autonomous cars and they do data augmentation to improve it and implement in a drone.

In the proposed approach, we consider two reference frames, an image, an autonomous drone racing capable of quickly adapting to new tracks, a convolutional network predicts the poses of the closest gate along with its uncertainty. These predictions are incorporated in an extended Kalman filter to maintain optimal estimates of the gate location. Given the estimated gate poses, they use predictive control to navigate on the track.

Finally, another related work is the presented by Loquercio, et al. [7] where they propose a convolutional neural network, whose purpose is to reliably drive an autonomous drone through the streets of a city. Trained with data collected by cars and bicycles, their system learns from them to follow basic traffic rules, e.g., do not go off the road, and to safely avoid other pedestrians or obstacles. They used a dataset that originally was thought for autonomous cars and they do data augmentation to improve it and implement in a drone.
Fig. 2 b. Considering the current drone state as the robot state to be processed by the onboard controller. See Fig. 2a. In this paper, we propose to directly estimate the next planned state, which is processed by the onboard controller, see Fig. 2a. Previous work usually estimates the gate position, filters previous positions, and outputs the next state. Namely, we will train a CNN, \( \Phi \), so that

\[ C_{\text{next}} = \Phi(I_t) \]  

where

\[ I_t = (v_t, C_t) \]  

and 

\[ C_t = \{ s_t, z_t \} \]

Fig. 3: The inertial coordinate frame \( O \) and the body coordinate frame \( V \).

We define a body frame velocity as 

\[ s = (\dot{x}^v, \dot{y}^v, \dot{z}^v) \]  

where \( \dot{x}^v \) is the forward or backward speed, \( \dot{y}^v \) the right or left speed, and \( \dot{z}^v \) the upwards speed.

According to the previous model, we define a reduced state as the current body frame velocity and altitude:

\[ C = \{ s, z \} \]  

Note that, a reduced state is a simplification of the full robot state because we are not including the current position or rotational speeds. This reduced state will be used as the output of the network. From now on, we will call state to the reduced state.

A. Behavioral Cloning Approach

Previous work usually estimates the gate position, filters the observations and plans a desired next state so that the drone flies through the gate, in some cases, a full trajectory (sequence of states) is planned [3], once the next state is planned it is processed by the onboard controller, see Fig. 2a. In this paper, we propose to directly estimate the next robot state to be processed by the onboard controller. See Fig. 2 b. Considering the current drone state as \( C_t \), the state to be estimated is written as \( C_{\text{next}} \), this next state should lead the drone through the gate.

In this approach, we avoid the gate position estimation and planning and we use a data-driven approach. The idea is to use the knowledge of a human pilot for training a CNN that receives as input the current camera frame \( I_t \) and outputs the next state. Namely, we will train a CNN, \( \Phi \), so that

\[ C_{\text{next}} = C_{t+n} \]

For this paper, we have set \( n \) equal to 1, it means that we assign to the current image the immediate following state. A more detailed experimentation on the impact of \( n \) over the performance is left for future work.

B. Dataset

Within Deep Learning, an important thing to obtain good results is to have a large amount of data to train the CNN. In our method, we build the dataset by extracting the log data for several tries of a human expert pilot. See Fig. 4.

The data that is stored from each flight corresponds to a vector conformed by the image coming from the camera at time \( t \), \( I_t \), the current inertial speed, \( \dot{s}_t \), the current orientation in yaw, \( \alpha_t \), and the current altitude, \( z \). Note that we want to predict the body state, therefore, we convert the inertial readings to body frame readings, according to:

\[ \dot{s}^B = \dot{Q}^B R(\alpha_t, \beta_t, \gamma_t) \cdot \dot{s}_t \]  

where \( \dot{Q}^B R \) is the inverse of eq. 3, and \( \beta_t \) and \( \gamma_t \) are the angles that relate the coordinate systems. Once the conversion is done, the raw dataset is stored as:

\[ \text{RawData}_t = (I_t, C_t) \]  

Given that we want to predict the next state and the raw dataset stores the current state, we convert the raw dataset. In this conversion, we match the image at time \( t \), \( I_t \), with a future state \( C_{\text{next}} \). It is necessary to train the CNN with the future state because the output of the Network has to predict the movement that pass the gate. For this reason, the training dataset examples are given as follows:

\[ \text{Dataset}_t = (I_t, C_{\text{next}}) \]  

given that the next state must be one of the stored frames, we define the next state as

\[ C_{\text{next}} = C_{t+n} \]
Fig. 5: Examples of images stored in the dataset.

To collect the data, the drone was positioned on the ground at different random distances from the gate, in addition, the lateral position of the drone with respect to the gate was changed. In all cases, the gate was visible by the drone at the initial state. By using several initial locations, we expect to generalize the behavior of the drone. Once the drone is on the ground the pilot rises the drone and command it until it passes the gate, once the gate is passed the drone is landed. It is worth to mention that all the data after the gate is passed is discarded. Such removal is done because we have found that including examples where the gate is not present avoids the learning (loss is not decreased). As a result, a total number of 150 manual flights were recorded. From these flights we generate 2,039 examples where a the next state is assigned to the current image. In Fig. 5 some examples of the stored images are displayed.

C. Architecture of the Deep Neural Network

To predict the next state, we have adapted the Convolutional Neural Network (CNN) called LeNet-5 proposed by LeCun et al. [7]. Our custom Lenet-5 network is implemented as we describe next. It receives as input a 3-channel image with resolution $320 \times 180 \times 3$. Then, 16 convolutional filters of size $10 \times 10$ are applied with stride one. Followed by a hyperbolic tangent activation function. Next, a max pooling operation is performed using a kernel of $4 \times 4$ and stride four. Then, a new set of 32 convolutional filters with size $5 \times 5$ and stride one is applied. Followed by a hyperbolic tangent activation function. Again, a max pooling operation is performed, using a kernel of $2 \times 2$ and stride two. The last convolutional layer applies 64 convolutional filters of size $5 \times 5$ and stride one. Followed by a hyperbolic tangent activation function and a max pooling operation is performed, using a kernel of $2 \times 2$ and stride two. Then, features are flattened to a one-dimension vector of size 12,160. Finally, one fully connected layer is applied with output length equal to the next state length.

This architecture is one of the simplest deep networks available in the literature, that’s why decided to implement it as a first approach in order to characterize the behavior of the drone.

IV. EXPERIMENTS

This section describes the experiments carried out in order to validate the proposed method. The first part describes the drone hardware. Then, we present the training of the CNN as well as the parameters that were used. Finally, the last experiment test the trained CNN with a set that was not seen by the CNN during training.

A. Hardware

The drone used is a Bebop 2 from Parrot, some of the characteristics of this drone are:
- Flight autonomy 25 minutes
- Weight 500 grams
- 14 Mpx camera with wide-angle lens
- 1080p FULL HD video
- Compatible with ROS

It was decided to use this drone because it has compatibility with ROS and it controlled by bebop autonomy driver based on Parrots official ARDroneSDK3 [8].

B. Training

The implementation of the Deep Neural Network was done in PyTorch, which is an open source machine learning library for Python, based on Torch, it is designed to perform numerical calculations using the programming of tensors. In addition, it allows its execution in GPU to accelerate the calculations.

The deep network takes as input a $320 \times 180$ RGB image and regresses the vector $\hat{C}_{\text{next}}$ that contains the next state to fly through the gate. Part of the parameters used for training the neural network are:
- Learning rate: 0.0001
- Batch size: 100
- Optimizer: Adam [6].
- Loss function: Mean Square Error
- Epochs: 163

To improve the network performance we carry out data augmentation, indeed, we flip each image and we change the contrast. Note that for each flipped image the next lateral speed is also flipped, namely $y^v_{\text{flip}} = (-1)y^v_{\text{next}}$.

Figure 8 shows the loss for each epoch of during the training of the artificial neural network. As we can observe in the figure, the loss is decreased in the first fifty epochs.

C. Test with a Unseen Flight

In this experiment, we test the trained CNN on unseen data (it was not observed during training nor validation). The objective is to simulate the use of the network in a real flight and validate the predictions before implementing it in the real drone.
In Figure 9 we present some examples of current images and the corresponding network predicted output. The left side column shows the input images ($I_t$), the central column shows with a bar graph the values of the CNN output ($C_{t+1}$) and the right column shows the difference between the values of the predictions and the values of the current state ($\hat{C}_{t+1} - C_t$).

In the first row, for the given image, the predicted state (draw with the blue bars) indicates that the velocity on the body frame $x$ axis is slightly positive (0.8 m/s), producing a slight forward movement (See Fig. 3), the velocity on the $y$ axis is positive indicating a movement to the left, the velocity on the $z$ axis is positive indicating an upwards movement, the last element indicates the desired altitude. As we can see, the observed output is coherent with the given input image. A similar behavior is observed on the remaining examples. The green graphs are showing the difference with respect to the current state, in a future work, such differences can be used as input to the control system.

As future work we will have to add a filtering stage that allows us to make smooth movements of flight, it will be necessary to have knowledge of the present control states and the predictions given by the artificial neural network, so that we can avoid abrupt movements by sudden changes in speed.

V. CONCLUSIONS

In this article, we presented an approach for guiding a drone to pass a gate. The approach uses a set of human flights to train a convolutional neural network. Once the network has been trained, it is capable of predicting the next state so that the drone is lead through the gate. In our proposition, the gate detection, estimation, and planning are replaced by a single forward pass of the network. In our experiments, we have observed that the outputs are coherent with respect to the input image. Our next step is to integrate the method into the drone to perform drone racing.

REFERENCES

Fig. 9: Input image with its corresponding network output. Left column shows the current input image. Center column shows the predicted next state ($\hat{C}_{t+1}$). Right image shows the difference with respect of the current drone state ($\epsilon = \hat{C}_{t+1} - C_t$).