Learning to Capture a Film-Look Video with a Camera Drone

Chong Huang[1], Zhenyu Yang[1], Yan Kong[1], Peng Chen[2], Xin Yang[3], and Kwang-Ting (Tim) Cheng[4]

Abstract— **The development of intelligent drones has simplified aerial filming and provided smarter assistant tools for users to capture a film-look footage. Existing methods of autonomous aerial filming either specify predefined camera movements for a drone to capture a footage, or employ heuristic approaches for camera motion planning. However, both predefined movements and heuristically planned motions are hardly able to provide cinematic footages for various dynamic scenarios. In this paper, we propose a data-driven learning-based approach, which can imitate a professional cameraman's intention for capturing a film-look aerial footage of a single subject in real-time. We model the decision-making process of the cameraman with two steps: 1) we train a network to predict the future image composition and camera position, and 2) our system then generates control commands to achieve the desired shot framing. At the system level, we deploy our algorithm on the limited resources of a drone and demonstrate the feasibility of running automatic filming onboard in real-time. Our experiments show how our data-driven planning approach achieves film-look footages and successfully mimics the work of a professional cameraman.**

I. INTRODUCTION

One of the most common goals for drone enthusiasts is to capture film-like videos via drones. The development of intelligent drones makes it more convenient and efficient to capture high-quality footage, such as the footage complying with a predefined image composition (e.g. image center) and/or following predefined camera movement (e.g. circling). However, these heuristic settings provide few filming styles and cannot always satisfy the user's expectation. Therefore, we expect the drone to be more intelligent to learn filming skills and improvise cinematic videos.

Watching a large number of video clips captured by professional filmmakers is an effective way for beginners to learn video shooting skills and ultimately derive their own creative works. Such a "watching - learning - imitating" strategy has been successfully applied to camera planning in some automated filming tasks and is known as "imitation filming". [1] and [2] learned a recurrent decision tree to automate basketball game broadcasting with a pan-tilt-zoom (PTZ) camera. [3] learned an agent for tracking a salient object through 360° panoramic videos using a recurrent neural network. These successful applications of imitation filming benefit from low-degree-of-freedom control outputs and the manually-labeled data.

Inspired by these works, we aim to extend imitation learning to the more complex drone system to assist inexperienced users to capture cinematic footage. However, there exist several challenges that prohibit direct usage of the existing data-driven approaches in our task.

1) Hard to provide an objective evaluation metric: The goal of our task, i.e. cinematic aerial footage, is subjective. Although [4]–[6] provides several metrics (e.g. lighting, color and composition) to quantify the aesthetic quality of the footage, it is still difficult to use these metrics to drive the drone to capture cinematic videos.

2) Lack of the annotated training data: Imitation learning requires the video and synchronized camera pose as training data. Although existing visual-based camera pose estimation (e.g. ORB-SLAM [7, 8]) can estimate the camera pose (without the absolute scale), the ambiguous scale makes it infeasible to feed the camera pose with different scales into the training network.

In this work, we focus on filming videos containing one subject, the framing of the filmed subject, including the image composition and the relative camera position to the subject, is an important aspect of aesthetic assessment [9]. Based on this intuition, we propose the following techniques to address the above challenges:

1) We imitate the decision-making process of professional cameraman using a supervised learning network. The network learns to predict the next camera movement by minimizing the difference between the image composition and camera position generated from the professional videos and the output from the network.

2) We estimate the relative camera pose to the subject based on the perspective projection of the subject's skeleton. This method makes the scale of the camera pose to be related only to the height of the subject. Therefore, we can fix the subject's height to guarantee the input camera pose with the same scale.

Our system takes the shot framing given by a user as input and automatically records the scene as a professional cameraman. We mount two GPUs (one Manifold [10] and one TX2 [11]) on a DJI Matrix 100 drone [12] to achieve real-time computation. In addition, we develop a user interface (UI) to assist users to conveniently design shots.

In summary, our contributions are three-fold. First, we introduce imitation learning to drone cinematography, significantly enhancing the drone's intelligence. Second, our

[1] Chong Huang, Zhenyu Yang and Yan Kong are with Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA 93106 USA. (chonghuang, zhenyuyang, yankong@ucsb.edu)

[2] Peng Chen is with the College of Information and Engineering, Zhejiang University of Technology, Hangzhou 310023 China. (chenpeng@zjut.edu.cn)

[3] Xin Yang is with the School of Electronics Information and Communications, Huazhong University of Science and Technology, Wuhan, Hubei 430074 China. (xinyang2014@hust.edu.cn)

[4] Kwang-Ting (Tim) Cheng is with School of Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. (timcheng@usk.hk)

proposed system allows a user to focus on the shot design and dramatically reduce the needs for drone control. This simplifies capturing cinematic shots for drone beginners. Third, we implement the entire system, including skeleton detection, viewpoint prediction and trajectory planning, and demonstrate the feasibility of running the system in real-time on a standard drone platform with limited computation resources.

We discuss related work in Sec. II, and describe some preliminary information in Sec. III. We give the problem definition in Sec. IV. The implementation of training and testing are presented in Sec. V and Sec. VI, respectively, followed by the system architecture in Sec. VII. In Sec. VIII, we present the experimental results to evaluate our system. Finally, we give the conclusion in Sec. IX.

II. Related Work

Autonomous Aerial Filming: Commercially available applications are often limited to watching a specified target from a fixed viewpoint, e.g. ActiveTrack [13], or a fixed path, e.g., Quickshot [14]. These modes are hardly able to provide cinematic footage for various dynamic scenarios.

Consequently, several algorithms [15]–[22] regarding human-drone interaction have been proposed. [15]–[18] allow users to specify the subject size, viewing angle and position on the screen to generate quadrotor motion plans automatically. However, their methods consider the camera and drone as the same rigid body in the control optimization and as a result, this model suffers from shaking footage in flight. The systems in [19]–[21] model a camera on a gimbal attached to a quadrotor and apply two independent controllers to guarantee smooth videos. These techniques are used essentially to move the camera to the closest pose in terms of the user's input; therefore, the aesthetic quality of the video highly relies on the user's input. Huang et al. [22] designed an automatic drone filming system without the user input. This method employs an aesthetic objective–maximizing the visibility of the subject–to automate aerial filming in action scenes. But this objective is oversimplified and is not applicable to many real-world scenarios.

Imitation Filming: Imitation filming is essentially a data-driven autonomous camera planning. [1] and [2] directly use the video clips of basketball games to imitate professional filming for team sports. [3] uses images labeled with the object's position for their application of tracking the most salient object in the 360° panoramic video. Our system aims to capture aesthetic footage for human action, yet the definition of "aesthetic" is subjective and ambiguous. Therefore, it is difficult to formulate the problem without predefined heuristics.

III. Preliminary

A. Coordinates Definition

We denote $(\cdot)^w$ as the world frame, which is initialized by the drone's navigation system. $(\cdot)^c$ is the camera frame and $(\cdot)^v$ is the image frame, where the origin is the center of the screen. To describe the relative position between the subject and the camera, we use the subject's 3D skeleton joints to define the subject-oriented coordinate system $(\cdot)^s$. The subject-oriented coordinate system is updated with the subject's movement. More concretely, the origin is the center of the subject's 3D skeleton and three axes are defined as follows:

$$\begin{aligned} z^s &= z^w \\ x^s &= norm(p^w_{ls} - p^w_{rs}) \times z^s \\ y^s &= z^s \times x^s, \end{aligned} \tag{1}$$

where p^w_{ls} and p^w_{rs} denote the 3D positions of the subject's left shoulder and the right shoulder, respectively in the world coordinate system, and z^w denotes the z-axis of the world coordinates.

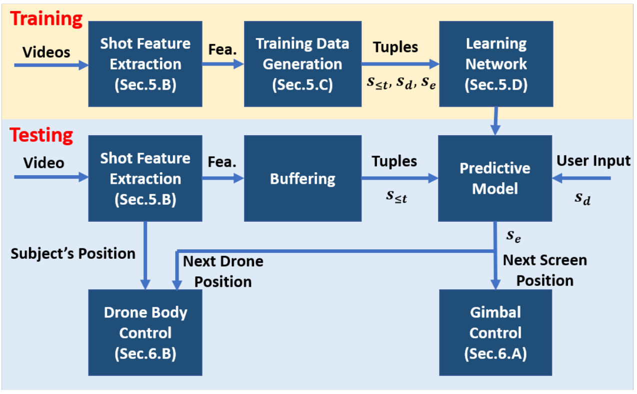

Fig. 1. The framework of imitation filming.

B. Shot Definition

We define the subject's appearance on the screen as "shot" [16], which is related to three important aspects: 1) camera-subject distance, 2) relative viewing angle, and 3) the screen position of the filmed target. Therefore, we represent the shot with two features:

$$s = \{p^v, T^s\} \in \mathbb{R}^5, \tag{2}$$

where T^s is the camera's relative position in the subject-oriented coordinated system, and p^v is the center of the subject's 2D projection on the camera screen.

IV. Problem Definition

Our system takes the desired shot s_d given by the user as the input and automatically records the video, where the evolution of the subject's appearance over time is close to professional filming. Considering that aerial filming is a continuous process, the future camera movement is determined by not only the current shot s_t but also the previous shots $\{s_{t-K}, ..., s_{t-2}, s_{t-1}\}$. The set consisting of the current and previous shots is denoted by $s_{\leq t}$.

Our task is to model the conditional probability of the next camera movement based on $s_{\leq t}$ and s_d. Because the camera pose directly corresponds to how the subject appears on the screen, we divide the motion prediction into two steps: 1) predict the next shot s_e based on the previous shots $s_{\leq t}$ and the desired shot s_d, and 2) estimate the next camera pose

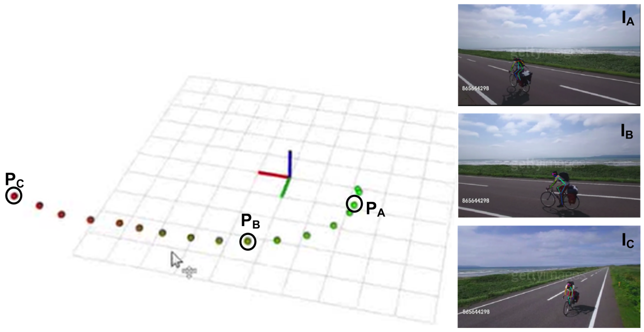

Fig. 2. The camera poses (P_A, P_B and P_C) are estimated from a sequence of frames (I_A, I_B and I_C). The estimated camera positions are represented in the subject-oriented coordinate system.

based on the predicted shot. Finally, the next camera pose will be published to the flight control to guide the drone.

In the following, we introduce the imitation filming method in terms of training phase and testing phase.

V. Training

We illustrate the training part of our method in Fig. 1(top). First, we extract the shot features from the collected professional videos. Second, we use the sliding windows to create the training data set, where each tuple consists of $s_e, s_{\leq t}$ and s_d. Finally, we use the supervised learning network to train a prediction model $p(s_e|s_{\leq t}, s_d)$.

A. Data Collection

We start by collecting a set of demonstrations for our task. To obtain continuous video clips with good image composition and smooth camera movement, we download videos containing only one person from *www.gettyimages.com*, which offers professional photography and videography. Specifically, we use the keywords ”aerial view, one man only, sport” to obtain 1,641 videos clips, each of which is around 15 seconds long. Because some videos are captured in poor lighting conditions, from a long distance, and/or include occlusions, which affect the 2D skeleton detection, we feed these videos to a 2D skeleton detection network based on OpenPose to remove the videos where the subject cannot be identified in more than 4/5 of the sequence. Because we resize the input video by 304x176 pixels to guarantee real-time computation in the testing phase, we also resize the training data to achieve the same scale. Finally, we obtain 298 feasible videos, from which 200 videos are randomly selected as the training set and the remaining videos are test set.

B. Shot Feature Extraction

In this subsection, we present how to extract the feature Eq. 2 from the aerial video. We divide feature extraction into five main steps:

1) We use Openpose [23] to detect the 2D skeleton in the image. To address incomplete 2D joint estimation caused by occlusion, we use the value in the previous frame to compensate the missing space of the current frame. The center of the 2D skeleton joints is set as p^v.

2) We use a seq2seq model [24] to estimate the 3D skeleton from the 2D skeleton. The estimated 3D skeleton is without global position information.

3) We follow [25] [26] and use the predefined subject's height to estimate the subject's relative position to the camera, where the scale of the camera-subject distance is related to the subject's height. Considering that the network requires the input camera pose to maintain the same scale, we set the height of the subject to be the same in all the videos. In fact, the height setting has no impact on learning because the input will be normalized before being fed into the network. We set the height as $1.8m$ in training phase.

4) We then transform the subject's relative position in the camera coordinate system to the camera's position T^s in the subject-oriented coordinate system.

5) We normalize the shot feature to balance the scale between the screen position and the spatial position as follows:

$$\begin{aligned} \hat{x}^v &= x^v/(width/2) \\ \hat{y}^v &= y^v/(height/2) \\ \hat{x}^s &= x^s/max(x^s) \\ \hat{y}^s &= y^s/max(y^s) \\ \hat{z}^s &= z^s/max(z^s), \end{aligned} \tag{3}$$

where $max(x^s), max(y^s)$ and $max(z^s)$ are the maximum distances of the training data in the three respective axes. The $width$ and $height$ are the pixel-wise width and height of the input video, respectively. Each video is represented as a sequence of vectors $s = [\hat{x}^v, \hat{y}^v, \hat{x}^s, \hat{y}^s, \hat{z}^s]$.

C. Training Data Generation

In this subsection, we introduce how to construct the training tuples given a sequence of shot features. We utilize an N-length sliding window to scan the whole sequence. We select the feature of the first K frames (K will be discussed in Sec. 5. D) of the window to be $s_{\leq t}$. We set the shot feature of the $(K+1)_{th}$ frame and the N_{th} as the next shot s_e and s_d, respectively. To cover more cases, the length of the sliding window starts with 20 frames for each scan and increases until it is the length of the entire video clip. In addition, we flip each frame of the video horizontally to augment the training video.

D. Learning Network

In this subsection we describe how to model the conditional probability $p(s_e|s_{\leq t}, s_d)$. A natural choice is to use a neural machine translation (NMT) architecture [27]–[30], which consists of two components: (a) an encoder, which computes a hidden state for the source input words, and (b) a decoder, which generates one target output word given a target input word. The objective is formulated as follows:

$$J_\theta = \sum_{(s_e, s_{\leq t}, s_d) \in \mathbb{D}} -\log p(s_e|s_{\leq t}, s_d, \theta), \tag{4}$$

where θ are learned parameters of the encoder and the decoder, and $\mathbb{D}$ are our parallel training corpus.

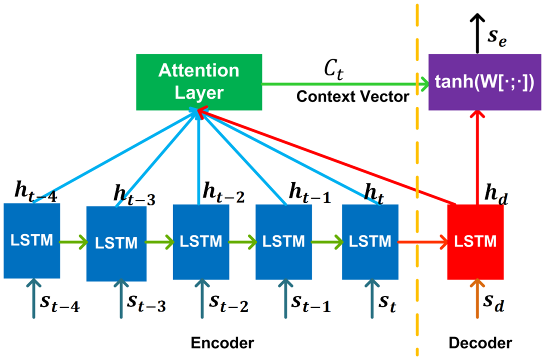

Fig. 3. The network architecture for imitation filming.

In our application, the encoder and decoder architecture are based on two long short-term memory (LSTM) networks [31] with 512 hidden units. This allows the network to learn when to forget previous hidden states and when to update hidden states given new information. In addition, we wrap the LSTM with an attention layer [29, 30] to handle possible long-length sequences.

The network architecture is illustrated in Fig. 3. The encoder receives a sequence of shot features $s_{\leq t}$ and produces a context vector C_t. The decoder is responsible for predicting the next shot s_e given the context vector C_t and s_d. The context vector C_t is the linear combination of the previous K hidden states from the source input and corresponding attention weights, as follows:

$$C_t = \sum_{k=0}^{K} a_{t-k} h_{t-k}, \tag{5}$$

The attention weight a_k is derived by comparing the current hidden state h_d from the decoder with each source hidden state of h_t from the encoder:

$$\begin{aligned} a_k &= align(h_k, h_d) \\ &= \frac{exp(score(h_k, h_d))}{\sum_{k \leq t} exp(score(h_k, h_d))}, \end{aligned} \tag{6}$$

where $score$ is referred to as a parameterized function to evaluate the similarity between h_d and h_k. Here we adopt Luong's multiplicative style $score(h_k, h_d) = h_d^T W h_k$ [29]. We analyze the attention weight a_k to understand where the network should focus its attention during decoding. Fig. 4 illustrates the distribution of the attention weights when the length of the input sequence is 5. The last hidden state obtains the highest attention (a_4) from the model and creates a context vector with more than fifty percent weights. Because the sum of the last four average attention weights ($a_{1\sim4}$) is more than 90%, it is sufficient to set the length of the input sequence to five ($K = 5$).

Given the target hidden state h_d and the source-side context vector C_t, we employ a simple concatenation layer to combine the information from both vectors to produce a transition viewpoint as follows:

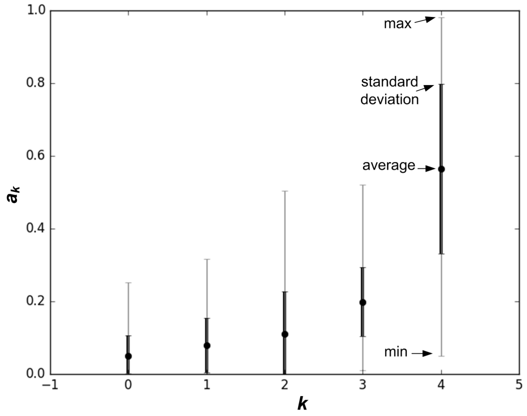

Fig. 4. The box plot of the 5-steps attention weights of 250 sequences. The attention weight a_k of the last step is the highest and plays the most important role to create a context vector.

$$s_e = \tanh(W[C_t; h_d]), \tag{7}$$

Across all the experiments, we use Adamax [32] to perform the optimization, with a learning rate of 0.0001.

VI. Testing

In this section, we introduce the implementation of the test phase (see Fig. 1(bottom)). First, the system extracts shot features of the input video stream in real-time and collect the shot features of the latest 5 frames as the buffer $s_{\leq t}$. Simultaneously, we follow [25] [26] to use the extracted skeleton and the prior knowledge of the subject's height to estimate the subject's relative position to the camera. Given the known drone's positioning information, we can obtain the position and orientation of the subject in the real world.

Second, we take the framing objective given by the user as s_d, and then feed $s_{\leq t}$ and s_d into the learned network (Sec. V. D) to predict the feature of the next shot s_e.

Finally, we apply the joint quadrotor and camera model used in [19] [?] to model the gimbal and drone body. The subject's screen position p^v and the relative camera position T^s in s_e are used to guide the gimbal and drone body movement independently. It is noted that the predicted p^v and T^s are required to recover their scale based on the inverse operation of Eq. 3 before further processing.

A. Gimbal Control

We apply the PD controller to adjust the gimbal camera to place the subject in the predicted screen position p^v.

B. Drone Body Control

We adopt min-snap (second derivative of acceleration) piecewise trajectory planning [33] to guide the drone. The generated trajectory consists of two-segments: $s_t \rightarrow s_e$ and $s_e \rightarrow s_d$, where each polynomial is parameterized to the time variable t in each dimension out of x, y, z and yaw. The 2-segment trajectory of one dimension can be written as:

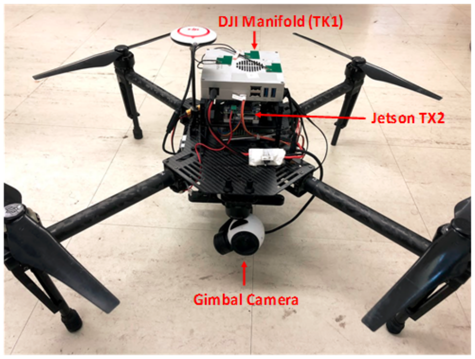

Fig. 5. The prototype of our drone platform

$$D_{\mu}(t) = \begin{cases} \sum_{j=0}^{n} p_{1j}(t-t_0)^j & t \in [t_0, t_1] \quad s_t \rightarrow s_e \\ \sum_{j=0}^{n} p_{2j}(t-t_1)^j & t \in [t_1, t_2] \quad s_e \rightarrow s_d, \end{cases} \tag{8}$$

where p_j is the jth order polynomial coefficient of the trajectory. t_1 and t_2 are the end time of each trajectory, with the total time of $\triangle t = t_2 - t_0$, which is calculated by the segment length, maximum velocity and acceleration based on the trapezoidal acceleration profile [34].

Instead of formulating the objective function for each dimension as in [35], in this paper, the coefficients in all the x, y, z and yaw dimensions are coupled into one single equation:

$$J = \sum_{\mu \in \{x,y,z,yaw\}} \int_0^T \left(\frac{d^4 f_{\mu}(t)}{dt^4} \right)^2 dt. \tag{9}$$

Finally, we re-plan the trajectory by solving the objective function minimization Eq. 9 in real-time. The drone executes filming footage along the trajectory.

In practice, we need to check maximum velocity and acceleration of the trajectory to ensure dynamical feasibility. If the acceleration or velocity of trajectory exceeds the maximum value, we extend the flight time $\triangle t$ and recalculate Eq. 9 to get a new trajectory. Then check the feasibility of the trajectory until that it meets the requirement. For simplification, we only check the trajectory at most five iterations and extend the time $\triangle t$ by 1.2 times each iteration. The maximum acceleration and velocity is set as $2.5m/s^2$ and $1.5m/s$. If the trajectory is still infeasible after five iterations, we do not move the camera. In most cases, we can solve a feasible trajectory at most two iterations.

VII. System Architecture

We integrate two processors and a gimbal camera into a DJI Matrix 100, as Fig. 5 shows. We use DJI Guidance System to provide positioning information. We choose a powerful GPU Jetson TX2 to run shot feature extraction. Meanwhile, we use DJI's Mainifold (customized Jetson TK1) to decode the video of the onboard gimbal camera and to communicate with DJI Guidance System. As a result, we use a combination of one TX2 and one Manifold to run the entire system simultaneously. The TX2 is equipped with a quad-core ARM Cortex-A57 processor, a dual-core Denver2 processor and 8 GB memory. The 256 GPU cores on the TX2 make it particularly suitable for parallel computing of body keypoints detection. Compared with TX2, Manifold is less powerful and it is equipped with a quad-core ARM Cortex-A15 processor, 2 GB memory and 192 GPU cores. We use a Zenmuse X3 Gimbal Camera to capture stabilized footage. To achieve real-time performance, each frame is resized to 304x176 before further processing.

We deploy different modules to two processors based on their computation complexity. Table I shows the runtime of different modules for each frame. More precisely, the TX2 is dedicated for shot feature extraction, and the DJI Manifold covers the viewpoint prediction and camera planning. Both processors are powered by the battery of the DJI Matrix 100 and are connected using an Ethernet cable. Communication between two computers is done by utilizing the ROS infrastructure. Meanwhile, the user utilizes the user interface on the ground PC to design the shot and send it to the drone using Wi-Fi.

TABLE I
Runtime of Different Modules

GPU	Module	Runtime (ms)
TX2	Shot Feature Extraction	218.47
Manifold	Viewpoint Prediction	22.43
	Gimbal/Drone Body Control	17.36

VIII. Experiments

In this section, we conduct quantitative and qualitative experiments to evaluate our method. These experiments are designed to answer the following questions:

A. *Does the predictive model learn the filming skills from the professional videos?*

Experiment: We train two learning networks with the professional videos from gettyimages.com and the random single-subject videos from Youtube. We keep the same amount of training data (200 videos) and compare the prediction error of the test videos from gettyimages.com.

TABLE II
The prediction error of the screen position and relative camera position

Training Data	Screen Position	Relative Camera Position
	(pixel)	(m)
Professionals	**11**	**0.33**
Random	16	0.59

Result: Tab. II compares the prediction error of two models trained from different datasets. The screen positions and the relative camera positions predicted from the professional videos are more accurate than those from random videos. The

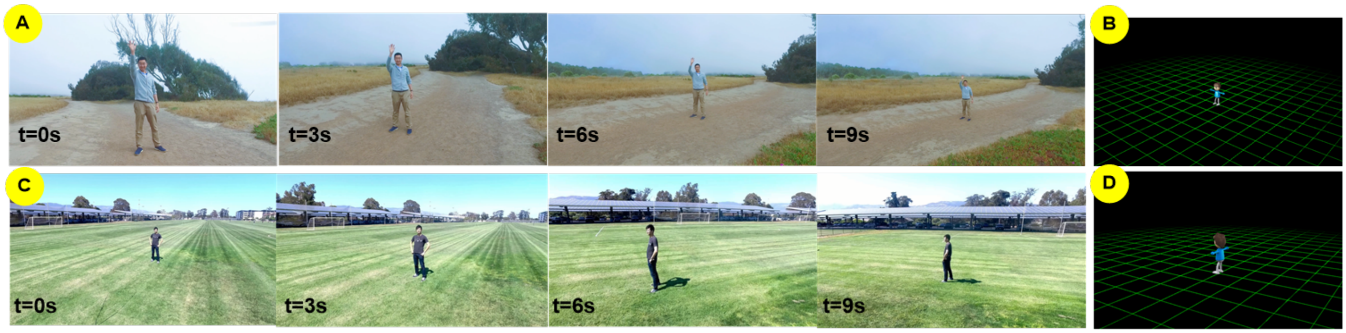

Fig. 6. (A, C) The snapshots of two footages captured by our drone system. (B, D) The framing objectives given by users in the User Interface.

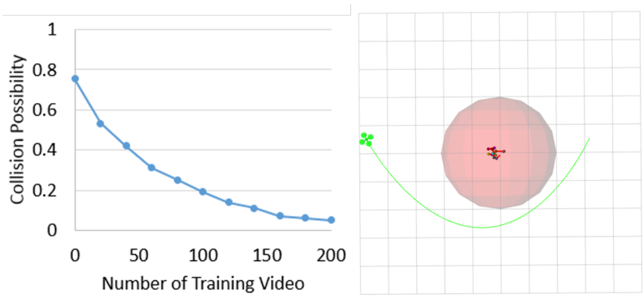

Fig. 7. Left: The possibility of collision decreases with the increasing training data. Right: The drone can pass by the subject in the test phase.

predicted results are related to not only the previous inputs but also the memory of the training data in the network. We can draw the conclusion that the predictive model does learn the filming skills from the professional videos.

B. Is the drone system capable of avoiding collision with the subject?

Experiment: We export one human model from the CMU motion capture dataset [36] to the simulation. Given the random user's input and the initial position of the drone, we count the collision times along with the increasing training data. We model the avoided region (see the red sphere in Fig. 7 (right)) using a sphere around the subject (2 m). If the drone intrudes into the avoided region, we consider it to be a collision. We test 100 times for each set of training data.

Result: Fig. 7 (left) shows that the possibility of collision decreases with the increasing training data. This can be explained by the fact that the training video implicitly includes the information of keeping the safety distance. Inspired by [37], we believe that the drone system can be more robust in avoiding collisions if we feed in more training video captured in different conditions.

C. What are the benefits of learning?

Experiment: We utilize the user study to analyze the benefits of learning via two experiments: 1) comparing the quality of the footage captured by beginners with versus without the assistance of our system, and 2) comparing the footages captured manually by experts and by beginners with the assistance of our system. We recruited 5 novice volunteers with no prior knowledge of cinematography nor drone piloting experience, and 5 volunteers with aerial filming experience. Each participant is required to capture 2 pieces of video clips with and without using our system, and then each one is assigned a questionnaire to score (from 1 (worst) to 5 (best)) the quality of all the video clips.

TABLE III
THE USER STUDY FOR BENEFITS OF LEARNING

Manual Filming by Beginners	Automatic Filming by Beginners	Manual Filming by Experts
1.71 ± 1.29	4.25 ± 0.53	4.11 ± 0.82

Result: The experimental result among beginners shows that the scores (4.25 ± 0.53) for the footage captured with our system are higher than for the footage captured manually (1.71 ± 1.29). In the second experiment, the footage captured by the beginners with the assistance of our system is close to that filmed by the experts, which demonstrates that our system does successfully mimic a professional cameraman.

Fig. 6 illustrates two sequences of snapshots of the video (A and C) and two framing objectives (B and D) given by users in the UI. The end frame (t=9 s) is consistent with the user's inputs. The attached videos demonstrate that our system achieves film-look footage and successfully mimics a professional cameraman. More comparison results are demonstrated in our demo video.

IX. CONCLUSIONS

In this paper, we proposed a novel autonomous drone filming based on imitation learning. Our system allows the user to customize the camera viewpoint, and later create a cinematic footage that connects the desired viewpoint. We divided this task into two steps: 1) we used supervised learning techniques to train a network to predict the shot framing in the next timestep, and 2) our system generates control commands to move the drone and gimbal to achieve the predicted shot framing. Our experimental results demonstrate the effectiveness of our system for assisting pilot beginners to capture film-look footage.

References

[1] J. Chen, H. M. Le, P. Carr, Y. Yue, and J. J. Little, "Learning online smooth predictors for realtime camera planning using recurrent decision trees," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4688–4696.

[2] J. Chen and P. Carr, "Mimicking human camera operators," in *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE, 2015, pp. 215–222.

[3] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun, "Deep 360 pilot: Learning a deep agent for piloting through 360 sports video," in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[4] Y. Luo and X. Tang, "Photo and video quality evaluation: Focusing on the subject," in *European Conference on Computer Vision*. Springer, 2008, pp. 386–399.

[5] S. Chung, J. Sammartino, J. Bai, and B. A. Barsky, "Can motion features inform video aesthetic preferences," *University of California at Berkeley Technical Report No. UCB/EECS-2012-172June*, vol. 29, 2012.

[6] Y. Wang, Q. Dai, R. Feng, and Y.-G. Jiang, "Beauty is here: evaluating aesthetics in videos using multimodal features and free training data," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 369–372.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[8] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] D. Arijon, *Grammar of the film language*. Focal Press London, 1976.

[10] dji, https://www.dji.com/manifold/, 2017.

[11] nvidia, https://developer.nvidia.com/embedded/buy/jetson-tx2/, 2017.

[12] dji, https://www.dji.com/matrice100/, 2015.

[13] ——, https://store.dji.com/guides/film-like-a-pro-with-activetrack/, 2015.

[14] ——, https://www.drone-world.com/dji-mavic-air-quickshot-modes/, 2018.

[15] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," vol. 2, no. 3, pp. 1696–1703, 2017.

[16] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 132, 2017.

[17] Q. Galvane, J. Fleureau, F.-L. Tariolle, and P. Guillotel, "Automated cinematography with unmanned aerial vehicles," *In Proceedings of the Eurographics Workshop on Intelligent Cinematography and Editing*, 2016.

[18] Q. Galvane, C. Lino, M. Christie, J. Fleureau, F. Servant, F. Tariolle, P. Guillotel, *et al.*, "Directing cinematographic drones," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 3, p. 34, 2018.

[19] N. Joubert, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, P. Hanrahan, *et al.*, "Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles," *arXiv preprint arXiv:1610.01691*, 2016.

[20] H. Kang, H. Li, J. Zhang, X. Lu, and B. Benes, "Flycam: Multitouch gesture controlled drone gimbal photography," *IEEE Robotics and Automation Letters*, 2018.

[21] C. Huang, Z. Yang, Y. Kong, P. Chen, X. Yang, and K.-T. T. Cheng, "Through-the-lens drone filming," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4692–4699.

[22] C. Huang, F. Gao, J. Pan, Z. Yang, W. Qiu, P. Chen, X. Yang, S. Shen, and K.-T. T. Cheng, "Act: An autonomous drone cinematography system for action scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7039–7046.

[23] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multiperson 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1611.08050*, 2016.

[24] M. R. I. Hossain and J. J. Little, "Exploiting temporal information for 3d pose estimation," *arXiv preprint arXiv:1711.08585*, 2017.

[25] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 44, 2017.

[26] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *3D Vision (3DV), 2017 Fifth International Conference on*, 2017.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[28] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[29] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[34] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, 2010.

[35] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2520–2525.

[36] cmu, http://mocap.cs.cmu.edu/, 2009.

[37] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.