UC Santa Barbara
UC Santa Barbara Electronic Theses and Dissertations

Title
Towards a Smart Drone Cinematographer for Filming Human Motion

Permalink
https://escholarship.org/uc/item/8tk2c096

Author
Huang, Chong

Publication Date
2020

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California
Santa Barbara

Towards a Smart Drone Cinematographer for Filming Human Motion

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Chong Huang

Committee in charge:

Professor Kwang-Ting (Tim) Cheng, Chair
Professor Matthew Turk
Professor B.S. Manjunath
Professor Xin Yang

March 2020

The Dissertation of Chong Huang is approved.

Professor Matthew Turk

Professor B.S. Manjunath

Professor Xin Yang

Professor Kwang-Ting (Tim) Cheng, Committee Chair

March 2020

Towards a Smart Drone Cinematographer for Filming Human Motion

Copyright © 2020

by

Chong Huang

I dedicate this dissertation to Avery Huang, whose sweetness encouraged and supported me.

Acknowledgements

I would like to express my sincere gratitude to Professor Tim Cheng for being an excellent advisor, mentor, and collaborator while I have been at UCSB. His guidance helped me to develop my critical thinking. His professional attitude, vision, flexibility and constant encouragement have deeply inspired me. It was a great privilege and honor to work and study under his guidance. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides, I would like to thank Professor Xin Yang for the continuous support of my Ph.D study and research, for her patience, motivation, enthusiasm and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.

I would also like to thank Professor Mathew Turk and B.S. Manjunath, for their interest in my research and sharing with me their expertise and visions in several different fields.

My sincere thanks also goes to my collaborator in Zhejiang University of Technology: Professor Peng Chen, for providing strong support of drone control and hardware technology. Also I thank my teammates: Yuanjie Dang, Chuan-En Lin, Zhenyu Yang, Weihao Qiu and Yan Kong, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last six years.

I love and admire my grandparents and want to thank them for always nurturing and encouraging me. I am deeply grateful to my parents, who are incredibly wise and generous, Mom and Dad, for continuous support me to pursue my Ph.D spiritually.

Finally, my heartfelt appreciation goes to my wife, Yun Ni, and our newborn baby, Avery Huang. Your company when the times got tough are much appreciated.

Curriculum Vitæ
Chong Huang

Education

2020	Ph.D. in Electrical and Computer Engineering, University of California, Santa Barbara
2014	M.S. in Information and Communication Engineering, Beijing University of Posts and Telecommunications
2011	B.S. in Information and Communication Engineering, Beijing University of Posts and Telecommunications

Experience

2019 Winter	Internship, Microsoft, Redmond, WA
2018 Summer	Visiting Student, Hong Kong University of Science and Technology, Hong Kong, China
2017 Summer	Visiting Student, Hong Kong University of Science and Technology, Hong Kong, China
2015 Summer	Internship, FX Palo Alto Laboratory, Inc. (FXPAL), Palo Alto, CA
2011-2014	Internship, France Telecom Orange Lab, Beijing, China

Research Interest

Human motion analysis, multi-feature fusion, motion planning, imitation learning

Publications

Chong Huang, Yuanjie Dang, Peng Chen, Xin Yang, and Kwang-Ting (Tim) Cheng, "One-Shot Imitation Filming of Human Motion Videos", axiv 2020

Yuanjie Dang, Chong Huang, Peng Chen, Ronghua Liang, Xin Yang, and Kwang-Ting (Tim) Cheng, "Imitation Learning Based Viewpoint Selection and Camera Control", IEEE/RSJ International Conference on Robotics and Automation (ICRA) 2019. (under review)

Chong Huang, Chuan-En Lin, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting (Tim) Cheng, "Learning to Film from Professional Human Motion Videos", IEEE Computer Vision and Pattern Recognition (CVPR) 2019.

Chong Huang, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting (Tim) Cheng, "Learning to Capture a Film-Look Video from a Camera Drone", IEEE/RSJ International Conference on Robotics and Automation (ICRA) 2019.

Chong Huang, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting (Tim) Cheng, "Through-the-Lens Drone Filming", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2018.

Chong Huang, Fei Gao, Jie Pan, Zhenyu Yang, Weihao Qiu, Peng Chen, Xin Yang, Shaojie Shen, and Kwang-Ting (Tim) Cheng, "ACT: An Autonomous Drone Cinematography System for Action Scenes", IEEE International Conference on Robotics and Automation (ICRA) 2018.

Chong Huang, Peng Chen, Xin Yang, and Kwang-Ting (Tim) Cheng, "REDBEE: A Visual-Inertial Drone System for Real-Time Moving Object Detection", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2017.

Chong Huang, Xin Yang, and Kwang-Ting (Tim) Cheng, "Accurate and Efficient Pulse Measurement from Facial Videos on Smartphones", IEEE Winter Conference on Application of Computer Vision (WACV), 2016.

Xin Yang, Chong Huang, and Kwang-Ting (Tim) Cheng, "libLDB: A Library for Extracting Ultrafast and Distinctive Binary Feature Description", ACM International Conference on Multimedia (MM), 2014.

Abstract

Towards a Smart Drone Cinematographer for Filming Human Motion

by

Chong Huang

Affordable consumer drones have made capturing aerial footage more convenient and accessible. However, shooting cinematic motion videos using a drone is challenging because it requires users to analyze dynamic scenarios while operating the controller.

In this thesis, our task is to develop an autonomous drone cinematography system to capture cinematic videos of human motion. We understand the system's filming performance to be influenced by three key components: 1) video quality metric, which measures the aesthetic quality – the angle, the distance, the image composition – of the captured video, 2) visual feature, which encapsulates the visual elements that influence the filming style, and 3) camera planning, which is a decision-making model that predicts the next best movement. By analyzing these three components, we designed two autonomous drone cinematography systems using both heuristic-based methods and learning-based methods.

For the first system, we designed an Autonomous CinemaTography system – "ACT" by proposing a viewpoint quality metric focusing on the visibility of the 3D human skeleton of the subject. We expanded the application of human motion analysis and simplified manual control by assisting viewpoint selection using a through-the-lens method. For the second system, we designed an imitation-based system that learns the artistic intention of the cameramen through watching professional aerial videos. We designed a camera planner that analyzes the video contents and previous camera motion to predict future camera motion. Furthermore, we propose a planning framework, which can imitate a

filming style by "seeing" only one single demonstration video of such style. We named it "one-shot imitation filming." To the best of our knowledge, this is the first work that extends imitation learning to autonomous filming. Experimental results in both simulation and field test exhibit significant improvements over existing techniques and our approach managed to help inexperienced pilots capture cinematic videos.

Contents

Curriculum Vitae **vi**

Abstract **viii**

1 Introduction **1**
1.1 Motivation 1
1.2 Thesis Goals and Contribution 3
1.3 Organization of Thesis 5

2 Background and Related Work **6**
2.1 History of Consumer Drones 6
2.2 Robotic Cinematography System 7
2.3 Virtual Cinematography System 9

3 Guiding the Camera based on Viewpoint Quality **11**
3.1 Related Work 11
3.2 ACT: An Autonomous Drone Cinematography System for Action Scenes 13
3.3 Through-the-Lens Drone Filming 30

4 Guiding the Camera based on Imitation Learning **50**
4.1 Related Work 52
4.2 Viewpoint-to-Viewpoint Camera Planning 53
4.3 Observation-to-Control Camera Planning 66
4.4 One-Shot Imitation Filming 73

5 Conclusion and Future Work **95**

A Appendix Title **98**

Bibliography **99**

Chapter 1

Introduction

1.1 Motivation

Storytelling through film is not just about recording the actions. It also requires one to select the proper viewpoint, plan the camera motion, and design the trajectory. The emergence of drones has raised the bar for cinematic quality and visual storytelling for cinematographers. Compared with conventional camera carriers (e.g., tripods, trucks and cranes), drones benefit from their high mobility to capture more cinematic shots with continuously varying viewpoints. However, filming human motions with a camera drone is a very challenging task, because it requires the cameraman to manipulate the remote controller and meet the desired filming style simultaneously. All of the above require the camera-person to have not only artistic insights but also the technical skills of camera control.

With the growing development of the onboard processors, drones become smarter and more autonomous. Some intelligent functions of the commercial products are developed to assist beginners to capture cinematic videos. For example, the ActiveTrack function in DJI Mavic can track the subject while capturing the video of the subject.

An alternative assistant tool (e.g. QuickShots) allows the drone to capture videos along predefined trajectories. Meanwhile, academic researchers focus on the problem of guiding camera based on specific configurations of image compositions. Joubert et al. [1] used the rule of thirds to guide filming static subjects. Nageli et al. [2] designed a model predictive control policy to track multiple moving subjects based on the customized on-screen positions. Although these techniques have achieved one-tap autonomous filming, the captured videos are fairly unexciting. There are three main reasons why this is the case: 1) the video quality metric is constrained by 2D image composition principles (e.g., rules of thirds). 2) The subject's on-screen appearance is oversimplified as a 2D point or bounding box. 3) The planning policy based on the single viewpoint metric limits the creativity of filming.

In summary, metric, feature and policy requirements of the state-of-the-art drone cinematography system have not yet reached a level that is satisfactory for autonomous filming. In this thesis, we target the autonomous drone cinematography system which can capture cinematic videos of human motion. We understand the systems filming performance to be influenced by three key components:

Video quality metric: Although the video aesthetic quality is subjective, we can still design some specific metrics to guide the camera to capture visually-pleasing video. One intuitive way to measure the video quality is based on the image aesthetic quality (e.g., camera angle, distance, image composition) over all the frames. An alternative solution is to consider the video as a sequential pattern of camera behaviors. In this context, the video quality is reflected by frame-to-frame coherency of viewpoint transition.

Visual feature: The design of visual feature determines the quantification performance of video quality. The desired visual feature should satisfy two requirements: 1) robustly describe the visual elements that influence filming style and 2) avoid significant delay on feature extraction.

Camera planning policy: The planning policy aims to generate a feasible trajectory to meet the desired effect. The desired planning policy should not only follow the video quality metric and but also allow the creativity in different scenarios.

In the following Section 1.2, we list the contributions of this thesis and Section 1.3 concludes by outlining the contents of the thesiss Chapters.

1.2 Thesis Goals and Contribution

This thesis concentrates on the autonomous drone cinematography system for capturing cinematic human motion videos. With this goal in mind, we first analyze the cause for unexciting shot in existing on-tap autonomous flight mode (e.g., DJI ActiveTrack), and we identify that the viewpoint metric based on 2D image compositions greatly limits the creativity of aerial filming because it only guides camera movements such as pan and follow. Based on the analysis, we proposed a quality metric "visibility of the subject," which enables the camera to adaptively select the camera angle around the subject based on the subject's 3D skeleton. To the end, we design a planning policy to dynamically drive the camera to the position that maximizes the viewpoint quality of the moving subject. Furthermore, we expand the application of the human motion analysis to simplify manual control. We discover that although moving control sticks can directly control a drones motion parameters (i.e. roll, yaw, pitch, and throttle), controlling these parameters do not offer a precise control of the movement of objects in the camera screen. To address this problem, we propose a "through-the-lens drone filming" mode, which assists the pilots using three key techniques: 1) 3D human localization that enables real-time subject tracking and 2) "through-the-lens" interface that allows the user to freely customize viewpoint in the subject-centered coordinates and 3) planning policy that transfers the camera configuration in virtual environment to the desired camera motion in real-world.

Furthermore, we expand the goal to imitate professional videos and capture more complicated cinematic shots. We focus on the filming style, which is the specific sequential patterns of camera behavior. We learn the artistic intention of the professional cameramen by minimizing the imitation error. Analyzing the impact of previous observation on the future camera motions, we design a camera planner which incorporates the video contents and previous camera motions to predict future camera motions to capture professional videos. Furthermore, we design a filming style feature based on the sequential pattern of video content. Using the filming style feature, we propose a one-shot imitation filming framework, which can imitate a filming style by "seeing" only a single demonstration video of the same style. This framework can eliminate the need for training videos for each style. To the best of our knowledge, this is the first work to extend imitation learning to autonomous filming.

To summarize, the contributions of this thesis include:

- An autonomous drone cinematography system "ACT" based on analysis of 3D human motion for addressing the problem of repetitive viewpoints in existing auto filming techniques.
- A semi-auto mode "through-the-lens drone filming" for simplifying manual operation of the drone to freely customize the viewpoint.
- A filming style feature for representing the temporal interaction between the subject and the background.
- An imitation filming framework which could imitate the filming style from only a single demo video, and generalize it in a broader set of situations, eliminating the need for training videos for each style.

1.3 Organization of Thesis

The thesis is organized as follows. The next Chapter provides the readers with a background, including the history of consumer drones, robotic cinematography system and virtual cinematographer. In Chapter 3, I describe the techniques of guiding the camera based on viewpoint quality. Chapter 4 presents the techniques of guiding the camera based on imitation learning. Chapter 5 concludes this work and highlights future research directions.

Chapter 2

Background and Related Work

2.1 History of Consumer Drones

During the mid-to-late 2000s, drones continued to grow in popularity among hobbyists. In 2010, the French company Parrot released their "AR.Drone," the first commercially successful ready-to-fly consumer drone, and the first able to be controlled solely by a Wi-Fi connection. In 2012, DJI released "Phantom 2" powered by the "Intelligent Li-Po battery", which enabled flight times of almost 20 minutes. This improvement attracts more and more cameramen to apply the consumer drone in filming industry. In 2014, 3D Robotics released the drone "IRIS+". With the GoPro camera fitted in a 3-axis gimbal, it can capture aerial videos with high mobility. In 2016, DJI came out with the "Phantom 4", boasting computer vision and machine learning to track an object on the ground without simply following a GPS track. In addition, the built-in "obstacle avoidance" function makes it safe to capture videos in clutter environments. Since 2016, the consumer industry focuses on designing more compact and foldable drones while keeping almost all of the technology (e.g. obstacle avoidance). The representative drone products include DJI Mavic Pro (2016), DJI Spark (2017), DJI Tello (2018). With the

development of the onboard hardware, the future consumer drone will be smarter, safer, cheaper, more autonomous.

2.2 Robotic Cinematography System

Drone Cinematography System Automatic drone cinematography attracts more and more attention from industry and academia. Some intelligent functions of the commercial products are developed to assist beginners to capture cinematic videos. For example, "ActiveTrack" mode is designed to fully autonomously follow a subject using a RGB camera. The 3DR Solo, DJI Phantom, Yuneec Typhoon, AirDog, and Ghost Drone all feature a "ActiveTrack" mode that tracks subjects. Another one-button autonomous flight function is to allow drone to capture the video along the predefined trajectory, e.g., DJI "QuickShot". DJI "QuickShot" provides four flight modes "Rocket, Dronie, Circle, Helix", all of which allow the user to get different types of shots throughout the time.

In academia, researchers combines the principles of image compositions and characters' pose information to capture more professional shots. Joubert et al. [1] utilize the visual composition principle to guide the camera control. Although the system has been successfully used to film a range of activities, such as taking a selfie, the subjects are stationary and the camera control does not respond to the limbs movements. Nageli et al. [2] [3] represent the body motion as a set of 3D markers, and allow users to specify the subject size, viewing angle and position on the screen to generate quadrotor motion plans automatically. These techniques simplify the camera control in specific scenarios and do not consider any visual aesthetic objectives to automate the camera drone control. As a result, the quality of the footage highly relies on the user's input.

Pan Control Cinematography System Automatic broadcasting makes small events, such as lectures and amateur sporting competitions, available to a much larger audience.

Several systems based on pan-tilt-zoom (PTZ) camera are designed to decide where the cameras should look. For example, Chen et al. [4] determined important subregions by considering user-defined attentional interests (such as including star players). Daigo et al. [5] developed a system which controlled a robotic camera by tracking audience face directions and a rough region of where players were located on a basketball court. Most relevant to this thesis is the work in Chen et al. [6]. They learned the relationship between player locations and corresponding camera configurations by crafting features which can be derived from noisy player tracking data, and employ a new calibration algorithm to estimate the pan-tilt-zoom configuration of a human operated broadcast camera at each video frame. Using this data, they trained a regressor to predict the appropriate pan angle for new noisy input tracking data. Although we draw inspiration from their work, these tools are not directly applicable to drone cameras. The drone camera has more degree-of-freedom (DOF) (DOF: 6 vs 1), making it difficult to estimate the camera pose configuration from each video frame. In addition, aerial video has more complex visual elements (e.g. background) than the basketball game. It is more challenging to design a regressor to predict the future camera pose.

Stationary Cinematography System Stationary cinematography system operate in an offline or online manner by cropping subregions from recorded video after tracking players and/or the ball. Kim et al [7] designed a photographer robot for automatically controlling composition and taking pictures. Hu et al. [8] learned an agent for tracking a salient object through 360 anoramic videos. In contrast, our system explicitly considers the dynamics of drones when planning shots.

2.3 Virtual Cinematography System

Camera planning in virtual environments is an active research topic. Proposed solutions are dedicated to address different classes of problems: real-time tracking of targets, automated shot and edit planning, and designing cinematic narrative experiences.

Real-time tracking of targets. This refers to the problem of finding a collision-free camera transition from a start position to an end position such that the target is visible as long as possible. Halper et al. [9] designed a camera engine for track the player in computer games. The proposed method is based on dynamic consideration of the visibility of objects which are deemed to be important in a given game context, which can avoid the camera jumping around too much. Oskam et al. [10] proposed a real-time camera control system that uses a global planning algorithm to compute large, occlusion free camera paths through complex environments. The algorithm incorporates the visibility of a focus point into the search strategy, so that a path is chosen along which the focus target will be in view.

Automated shot and edit planning. This topic focuses on the problem of placing cameras to produce nice-looking views of the action in an offline manner. Galvane et al. [11] proposed a continuity editing model for 3D animations that provides a general solution to the automated creation of cinematographic sequences. Ranon et al. [12] introduced novel ways to define visual properties, evaluate their satisfaction, and initialize the search for optimal viewpoints, and test them in several problems under various time budgets, quantifying also, for the first time in the domain, the importance of tuning the parameters that control the behavior of the solving process.

Designing cinematic narrative experiences. With the advent of multi-player games, there is a significant demand in generating relevant cinematic replays of gaming sessions. Galvane et al. [13] presented a system that generates cinematic replays for

dialogue-based 3D video games. The system exploits the narrative and geometric information present in these games and automatically computes camera framings and edits to build a coherent cinematic replay of the gaming session.

The above techniques essentially focus on the problem of placing a camera using different levels of specification. Different from these work, the camera planning in the drone system is limited to the physical constraints and perception range. More importantly, aerial filming involves more complex visual elements that influences the video quality and camera motion. Therefore, we cannot directly apply virtual cinematographer in the field test.

Chapter 3

Guiding the Camera based on Viewpoint Quality

This chapter introduces two techniques about using viewpoint quality metric to guide camera. First, we introduce an autonomous drone cinematography system "ACT", which can adaptively adjust the camera angle based on the handcrafted viewpoint quality "the visibility of the subject". Second, we expand the application of the human motion analysis to simplify manual control, and propose a novel semi-auto viewpoint control mode "through-the-lens drone filming", which allows the user to freely customize the viewpoint in the virtual environment and then converts the camera configuration to the desired camera motion in real world.

3.1 Related Work

3D skeleton detection: Existing 3D skeleton detection methods rely on infrared-based depth sensors. The Kinect sensor is an easy-to-operate device for depth detection. The Kinect can track multiple subjects without requiring users to wear extra sensors.

Benefited from its compact size, the Kinect sensor [14] can be mounted on the robot to perceive unknown environments. However, the Kinect sensor calculates depth with a infrared laser projector, so it cannot work in the outdoor environment. Vicon is another widely-used system in the field of motion analysis because of its high accuracy of pose tracking. However, it is also restricted in the indoor environment because of its optical properties. Meanwhile, its immobility only track the subjects within a limited space.

Subject localization: The GPS-based [1] and the infrared-based [2] [3] wearable sensors are widely used for subject localization. However, the GPS does not work in the indoor environments, and the infrared-based sensors are restricted to the indoor environment because of their optical properties. Furthermore, it is not convenient to require every subject to wear sensors for filming. The vision-based localization frees the subject from wearable sensors, but the related work [15] [16] [17] requires the user to provide the subject's height, based on which to compute the global translation under perspective projection. However, these methods become invalid for users with unknown heights. Besides, Huang et al. [18] utilizes a stereo camera mounted on the drone to localize the subject, but its field of view is subjected to the drone body and cannot efficiently track the subject when the drone or the subject is moving.

Trajectory planning in computer graphics: Camera planning for human action has been widely studied in computer graphics. Researchers focus on searching for a set of suitable camera configurations for capturing an expressive video clip, while obeying a set of cinematographic rules [9], as well as other constraints such as occlusion [19], objects visibility [20] [21], layout in the resulting image [22], frame coherency [9] and orientations [23]. There are several metrics to quantify the aesthetic quality of the video clips such as subject's visibility [24] [25], shape saliency [26] and motion area [27]. Using these attributes, the system measures the quality of each frame taken from different viewpoint and outputs the best view. Camera planning is typically formulated

as an offline optimization problem which seeks a camera path in space-time 4D space by balancing the viewpoint quality and smoothness of the camera path. In addition, virtual environments are not limited by real-world physics and robot constraints and hence can produce arbitrary camera trajectories, velocities and viewpoints.

Viewpoint control in the computer graphics: Through-the-lens camera control [28] has been widely used in virtual cinematography [29] [30] [31] [32] and action games [33] [34]. However, these techniques are not feasible in real-world scenarios because a subject's position cannot be directly obtained like in virtual environments. Some researchers [1] [2] [3] use wearable sensors to localize a subject and automate filming for some predefined shots. In addition to the constraints imposed by sensors, their systems do not provide an efficient interaction for users to design desirable viewpoints.

3.2 ACT: An Autonomous Drone Cinematography System for Action Scenes

3.2.1 Introduction

In this section, we focus on the problem of filming human action video using the hand-crafted viewpoint metric. Existing auto-filming systems suffer from unexciting shots due to oversimplified viewpoint metrics and the representation of the subject. For instance, Joubert et al. [1] used the rule of thirds to guide filming static subjects. Nageli et al. [2] [3] designed a system, which can track multiple moving person automatically. However, their viewpoint metric is limited to the 2D image composition principles. In addition, the subject's perception highly replies on the wearable sensors, some of which are limited to indoor environments. All of these limits the creativity of camera filming.

To address the above challenges, our autonomous drone cinematography system in-

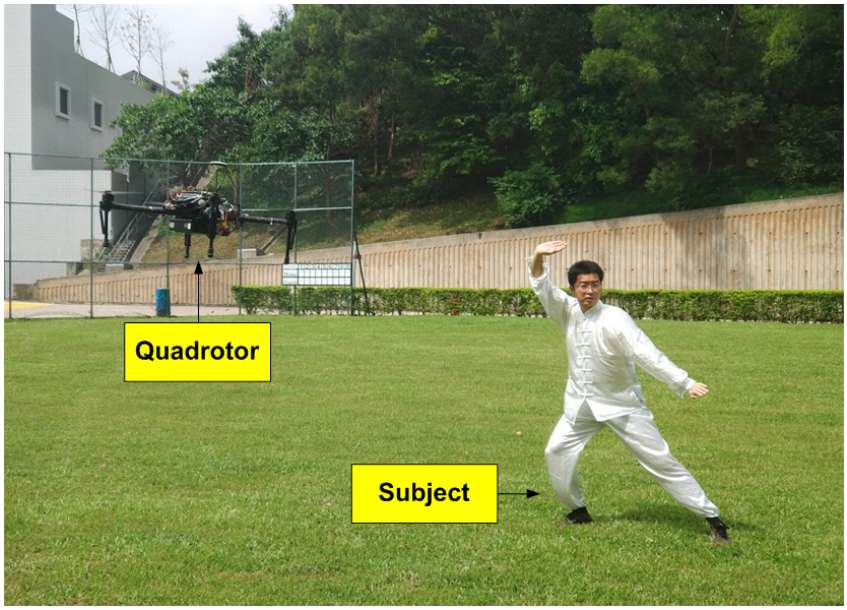


Figure 3.1: Autonoumous filming human action of the proposed ACT system.

cludes the following techniques:

1. For pose estimation, we combine stereo-based depth estimation and 2D body skeleton detection to estimate the 3D skeleton pose, and we refine the pose information based on the temporal properties of body movement.

2. For camera planning, we propose a real-time dynamically trajectory generator to guide the camera control for unknown body movements. The generated trajectory can balance aesthetic objectives and the physical limits of real robots.

To achieve real-time onboard pose estimation and camera planning, we mount a stereo camera and two GPUs on a DJI Matrix 100 drone. We use a gimbal RGB camera to capture the stabilized footage.

In summary, our contributions are two-fold. First, we propose an efficient 3D skeleton detection method based on a stereo camera and a real-time camera planning algorithm that can balance the aesthetic objective and physical limits. The system can be used in both indoor and outdoor environments. Second, we implement the entire system on the limited computation resource of the drone platform, including skeleton detection,

viewpoint estimation, trajectory planning and localization, and demonstrate its feasibility of running the system in real-time (see Fig. 3.1).

3.2.2 3D Skeleton Detection based on Stereo Camera

In this section, we introduce 3D skeleton detection. Our intuition is to recover the depth of the 2D skeleton position from the depth map. We do not consider the active depth sensor (e.g., Kinect) because it cannot work in outdoor environments. Instead, we use stereo cameras to calculate depth. However, the stereo-based depth estimation may generate inaccurate depth for the region with motion blurs. We add some constraints to refine the result. Details are introduced as follows:

Raw Depth Acquisition

We utilize a stereo camera to calculate depth based on semi-global block-matching (SGBM) [35] as Fig. 3.2 (b) shows. The rectified image stream from left camera feeds to a opensource library OpenPose [36] to detect 2D skeleton points. If the full body parts are visible, OpenPose can detect 13 keypoints, including the head, nose, hip, left and right shoulders, elbows, hands, knees and feet (see Fig. 3.2 (a)), which are used to represent the full human pose. Given a depth map based on left camera, we can convert each 2D body keypoint (x, y) on the image plane to a 3D body keypoint (X, Y, Z) as Eq. 3.1. The 3D body keypoints are connected as 3D skeleton (see Fig. 3.2(c)).

$$\begin{aligned} Z &= depth(x, y), \\ X &= (x - c_x) \cdot Z / f_x, \\ Y &= (y - c_y) \cdot Z / f_y, \end{aligned} \tag{3.1}$$

where c_x, c_y are the center of the image and f_x, f_y are the focal length of the camera on both axes.

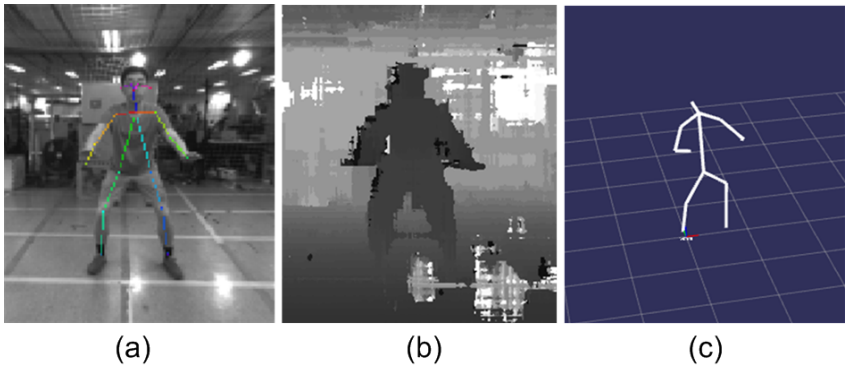


Figure 3.2: (a) 2D skeleton (b) depth map, and (c) 3D skeleton

Skeleton Refinement

As mentioned before, the noise of the depth map may affect the recovered 3D pose, especially moving subjects. We will refine the 3D pose based on the temporal consistency of human action.

Assuming that the movement of the body joints is smooth, we can use polynomial regression to parameterize the trajectory of each keypoint in terms of time. The predictor resulting from polynomial expansion can determine whether the current pose estimated from the depth map is trustworthy. Given a set of trustworthy poses $[s_0,\ s_1,...,s_N]$, we set the following optimization function to solve the polynomial coefficients:

$$\begin{aligned} &\min_{\{a_n\, a_{n-1} \cdots a_0\}} \sum_{i=0}^{N} (\bar{s}_i - s_i)^2 \\ &\text{subject to} \quad \bar{s}_i = a_n t_i^n + a_n t_i^n + \cdots + a_0, \end{aligned} \tag{3.2}$$

where $\bar{s}_i$ is the pose to be modeled at the ith frame and t_i is the timestamp of the ith frame. N is the number of training frames and n is the order of the polynomial function.

Considering that the acceleration of the body movement respects the physical limits of limb, we add a penalty term to control the acceleration along the trajectory as follows:

$$\min_{\{a_n\, a_{n-1} \cdots a_0\}} \sum_{i=0}^{N} (\bar{s}_i - s_i)^2 + w \int_0^T (\ddot{s})^2 dt, \tag{3.3}$$

where w is the penalty weight and is set as 200, and T is the time taken during N frames.

However, the actual limb movement is complex, and the accuracy of polynomial fitting is related to the pace and speed of human action. We discuss the selection of the parameter n and N in four scenarios from CMU Motion Capture Dataset: 1) TaiChi, 2) Walk, 3) Ballet dance, and 4) Run. Each clip of data takes 10 seconds, and the 3D skeleton of the entire sequence is known. We use the pose history in the past N frames to predict the current pose. We evaluate the performance by the average limb distance between the predicted and actual current pose.

Fig. 3.3 shows the distribution of the prediction error in different human actions. TaiChi and Walk have smooth limb motion and our polynomial function can achieve highly accurate prediction. In contrast, Run and Ballet Dance have fast limb motion with various paces. The limb movements are more difficult to predict, because they require a shorter time window to model the rapidly changing limb movement. The higher order polynomial function cannot improve the prediction accuracy. Therefore, we set the N and n as 15 and 5, respectively in our polynomial fitting.

In our system, we apply a simple voter to refine the 3D pose. If the distance between the pose estimated from the depth map and the modeled pose is larger than 0.5m, we set the modeled pose as refined pose. Otherwise, pose estimated from the depth map is set as refined pose. The refined pose is trustworthy and will be used to predict the future pose.

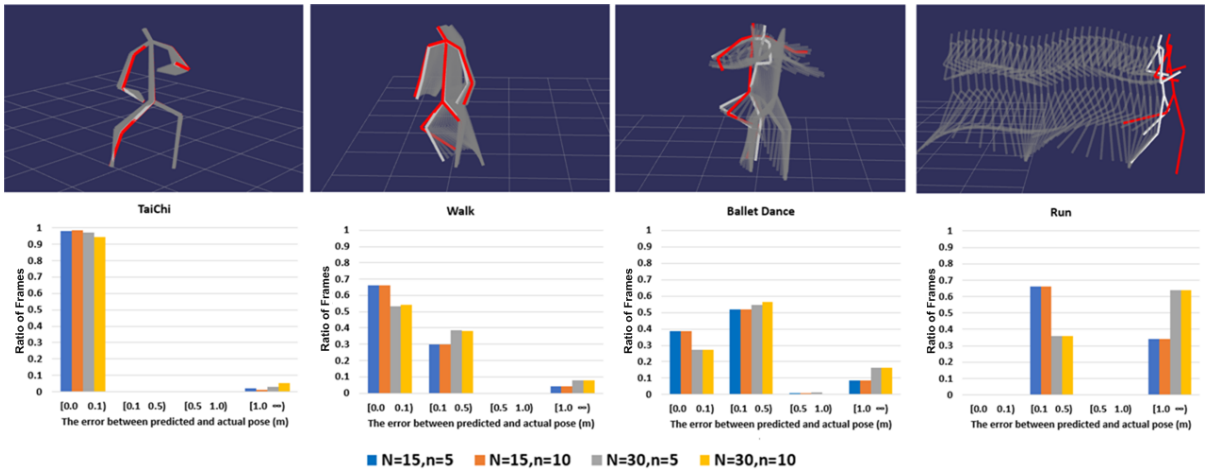


Figure 3.3: First row: actual current pose (white), previous pose (gray) and predicted current pose (red). The red pose is predicted from the previous 30 poses within one second. The gap between actual and predicted pose becomes larger as the faster movement pace. Second row: The distribution of prediction error in different human actions. The vertical axis refers to the ratio of frames with the specific prediction error in the whole video sequence, and the horizonal axis is the reconstruciton error between predicted and actual poses. N and n are the size of temporal window and the order of polynomial function, respectively.

3.2.3 Camera Planning based on Next-Best-View

In this section, we introduce camera planning. The goal is to design a trajectory that fulfills the aesthetic objectives and respects the physical limits of the real drone. First, we predict the human pose in the next frame by using predictor in Sec. III.A, and then calculate the best viewpoint of this pose, and then we generate the physically feasible trajectory that points to this viewpoint. Details on viewpoint selection and trajectory planning follow.

Viewpoint Selection

The viewing space is a subject-center sphere for each human pose. We estimate the best viewpoint in terms of the radius and orientation angle.

The radius, the camera-to-subject distance, determines the size of the subject on the

image. On the one hand, we need to ensure a smooth displacement of the subject on the image plane. Vzquez et al. [37] proposes to increase the camera-to-subject distance if the subject moves fast, and vice versa. On the other hand, we must keep a safe distance between the camera and the subject to avoid collision. The radius is estimated as follows:

$$r = r_0(1 + kv), \tag{3.4}$$

where r_0 is the minimum camera distance and k is a constant parameter to adjust the camera distance. v is the subjects current speed, which is represented by the average speed of the neck and hip keypoints. To gurantee a view of the subject's whole body, we set minimum distance r_0 as 3m. We set k as 0.4 to keep smooth and stable camera movement.

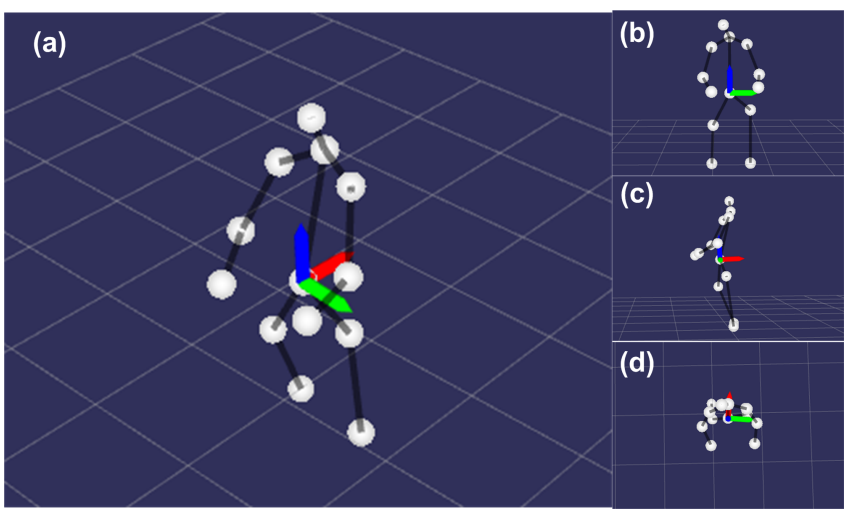


Figure 3.4: (a) The 3D skeleton points and its three eigenvectors corresponding to the eigenvalues in the descending order (BGR). The right images illustrate the camera view from (b) the third eigenvector, (c) the second eigenvector and (d) the first eigenvector. It is obvious that the camea view from the third eigenvector displays the maximum projection of point cloud.

The orientation angle defines the pitch and yaw of the camera relative to the subject, which determines the visible part of the subject on the image. There are several ways

[24] [25] [26] [27] to measure the quality of a view of a subject in computer graphics. The objective function that measures this is called a view descriptor, and the best view is that which maximizes this function. The view descriptor in Assa et al. [24] [25] measures the visiblity of the joint points of a character to quantify viewpoint quality. Because we represent human pose as 13 3D skeleton keypoints, we evaluate each frame with this metric. First, we calculate PCA for 3D skeleton keypoints to get three eigenvalues (λ_1, λ_2, λ_3) in descending order. The eigenvector corresponding to the minimum eigenvalue is set as the best view angle, because it is perpendicular to the plane with the largest projection of the point cloud as Fig. 3.4 shows.

$$\theta = \{\theta | P\theta = \lambda_3 \theta\}, \tag{3.5}$$

where P is a matrix consisted of 3D skeleton points and λ_3 is the minimum eigenvalue. The best viewing direction is along with the eigenvector θ pointing to the subject's center.

It is noted that not all the viewpoints with maximum projection are feasible. First, if the projections of the subject from different viewpoints are similar, subtle motion in the consecutive frames also cause "viewpoint jumping" (see Fig. 3.5). "Viewpoint jump" will cause a sudden change of acceleration during trajectory planning. This not only increases the instability of the flight control but also makes the footage unpleasing, so we do not move the camera in this case. Considering that eigenvalues is inversely proportional to the variation of the projection to which corresponding eigenvector is perpendicular, we can compare eigenvalues to evaluate the distinctiveness of different viewpoints. Therefore, we define Eq. 3.6 to estimate the probability of "viewpint jumping". If it is smaller than a threshold, it is likely to be "viewpoint jumping" and we skip this viewpoint. The threshold is set as 1.5.

$$\epsilon = \lambda_2 / \lambda_3. \tag{3.6}$$

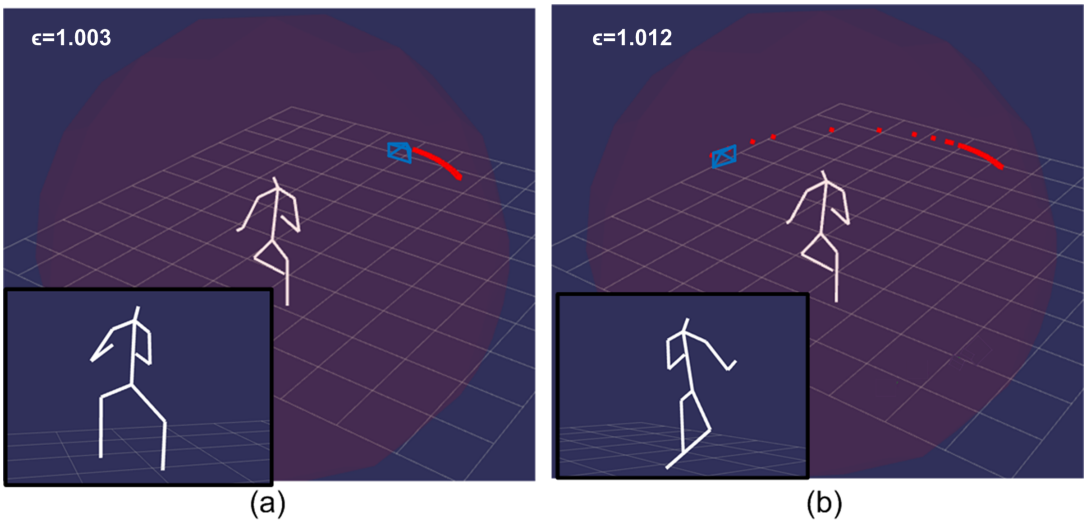


Figure 3.5: Viewpoint jumping. The world view and camera view (subfigure on the left-bottom) are shown at (a) T_0 and (b) $T_0 + 0.3s$. The red points are the best viewpoint for each frame and the blue camera is the current camera pose. Because the viewpoint quality of both camera views is very similar (ϵ is close to 1), subtle limb movement causes switch between second and third eigenvectors. The dramatic change of best viewpoint makes it infeasible to move the camera within short time in the real scenarios.

Second, because the stereo camera is fixed on the front of the drone, the observatoin range is determined by the drone's pose. If the viewing direction calculated by our algorithm can generate too high- or too low-angle shots, the stereo camera fails to track the subject within the field of view. To prevent this case, we set the maximum and minimum pitch angle of viewpoint as 15 and -45 degrees. In addition, we set the minimum flight height as 0.3 m to avoid colliding ground.

Trajectory Planning

Although our system shares the same view descriptor as [24] [25], there are several distinct differences in the constraints of the camera control. First, we cannot formulate the camera planning as an offline optimization for unknown human movement. Second, the flight control must respect its physical limists. Third, the drone must keep a safety distance with the subject. In this section, we present our optimization-based trajectory

generation method under the above constraints.

In our system, we re-plan a trajectory for each frame in real-time. Each trajectory is calculated based on current camera pose and next waypoint. We model the trajectory as one-piece polynomial, which is parameterized to the time variable t in each dimension x, y, z, yaw. The trajectory of one dimension can be written as follows:

$$f_\mu(t) = \sum_{j=0}^{n} p_j t^j \quad t \in [0, T], \tag{3.7}$$

where p_j is the jth order polynomial coefficient of the trajectory, and T is total time of the trajectory, which is calculated by the segement length, maximum velocity and acceleration based on trapezoidal acceleration profile [38]. The polynomial coefficients are computed by minimizing the integral of the square of the kth derivative along the trajectory. In this paper, we minimize the snap along the trajectory, so k is 4. Instead of formulating the cost function for each dimension as in [39], in this paper, the coefficients in all x, y, z, yaw dimensions are coupled into one single equation:

$$J = \sum_{\mu \in \{x,y,z,yaw\}} \int_0^T \left(\frac{d^k f_\mu(t)}{dt^k} \right)^2 dt. \tag{3.8}$$

The objective function can be written in a quadratic formulation $p^T Q p$, where p is a vector containing all polynomial coefficients in all four dimensions of x, y, z, yaw and Q is the Hessian matrix of the objective function.

We must define the following constraints to ensure the feasibility of the trajectory:

1) *Waypoint Constraints*: If there exists a waypoint at the time of T, we have

$$f_\mu(T) = d_T. \tag{3.9}$$

2) *Continuity Constraints*: The trajecotry must be continuous at all the kth derivatives at each waypoint between two polynomial segments:

$$\lim_{x \to T^-} f_\mu^{(k)}(T) = \lim_{x \to T^+} f_\mu^{(k)}(T). \tag{3.10}$$

The both constraints can be compiled into a set of linear equality constraints ($Ap = d$) in [40]. Thus, the trajectory generation problem can be reformulated as a quadratic programming problem:

$$\begin{aligned} &\min \quad p^T Q p \\ &\text{subject to} \quad Ap = d. \end{aligned} \tag{3.11}$$

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 T and recalculate Eq. 3.11 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 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.

In addition, although we have limited the minimum distance of each waypoint, the distance between subject and generated trajectory is likely to be less than safety distance. For safety, we define a sphere centered at the subject with radius r_s as a safety region. If the generated trajectory intersects with safety region, we skip this waypoint and do not move the camera. The radius r_s is set as 2m.

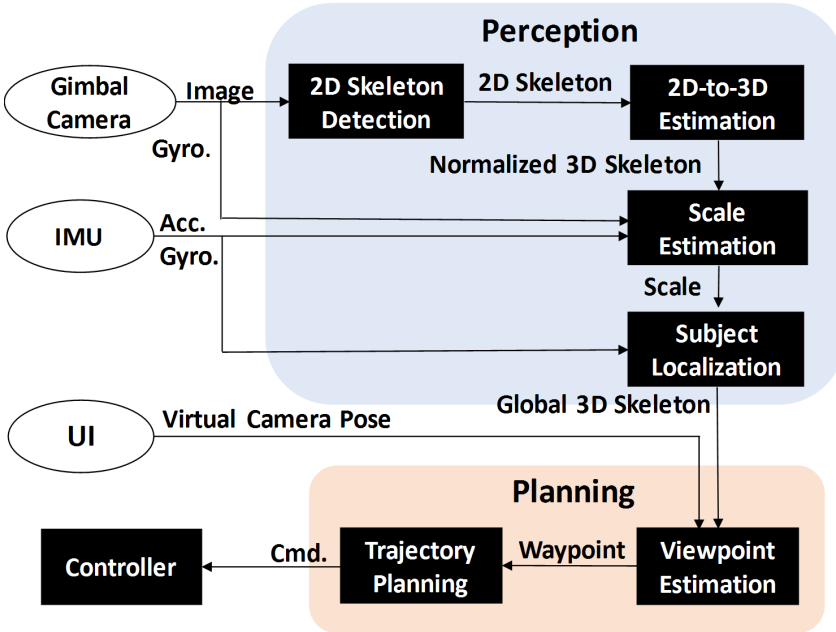


Figure 3.6: The Architecture of the System

3.2.4 System Architecture

The system architecture is shown in Fig. 3.19. In the perception module, we estimate 3D skeleton points by fusing stereo depth and monocular 2D Skeleton detection. Meanwhile, we adopt the state-of-the-art visual inertial system (VINS) [41] to get the 6 degrees of freedom (DoF) state estimation using image stream of left camera and IMU data stream. It is noted that we use image stream of left camera for 2D skeleton detection and state estimation. Given the pose of the camera, we can obtain the subject's pose in the world coordinates. In the planning module, we predict the next best viewpoint based on the subjects 3D movements and publish a sequence of the waypoints. Then the trajectory planning converts the waypoints to a feasible trajectory in real-time. The drone is commanded to fly through the trajectory and capture the footage with gimbal camera. Tab. 4.1 shows the runtime of different modules for each frame. Because the 2D skeleton detection and other modules are running parallelly, the runtime of a frame

is less than 200ms, which is sufficient in the aerial filming.

Table 3.1: Runtime of Different Modules

TX2	Module	Runtime(ms)
GPU1	Skeleton Detection	117.64
GPU2	Depth Estimation	35.21
	Skeleton Refinement	39.68
	Viewpoint Estimation	20.51
	Trajectory Planning	12.67
	State Estimation	52.22

We integrate processors, stereo cameras, and gimbal camera on DJI Matrix 100 as Fig. 3.7 shows. Because 2D skeleton detection and the stereo-based depth map take up most of the computation resources, they require GPU for real-time computation. Considering the power efficiency and limited load of the drone, we use two NVIDIA TX2 to run the whole system simultaneously. The TX2 is equipped with a quad-core ARM Cortex-A57 processor, a dual-core Denver2 processor and 8 GB memory, and consumes approximately 7.5 watts of power. The 256 GPU cores on the TX2 make it particularly suitable for parallel computing of depth images and body keypoints detection. The stereo camera module is constructed of two horizontal forward-looking MatrixVision mvBlueFOX-MLC200w5 global shutter cameras (740x480, 25 fps). We choose Zenmuse X3 Gimbal Camera for capturing stabilized footage and record the footage with resolution 1280x720.

We deploy different modules on two GPUs based on their computation complexity. More precisely, one GPU is only used for 2D skeleton detection, and the other GPU covers the rest of the computations. Both TX2 are powered by the battery of the DJI Matrix 100. The two TX2 are connected using an Ethernet cable. Communication between two computers is done by utilizing the ROS infrastructure.

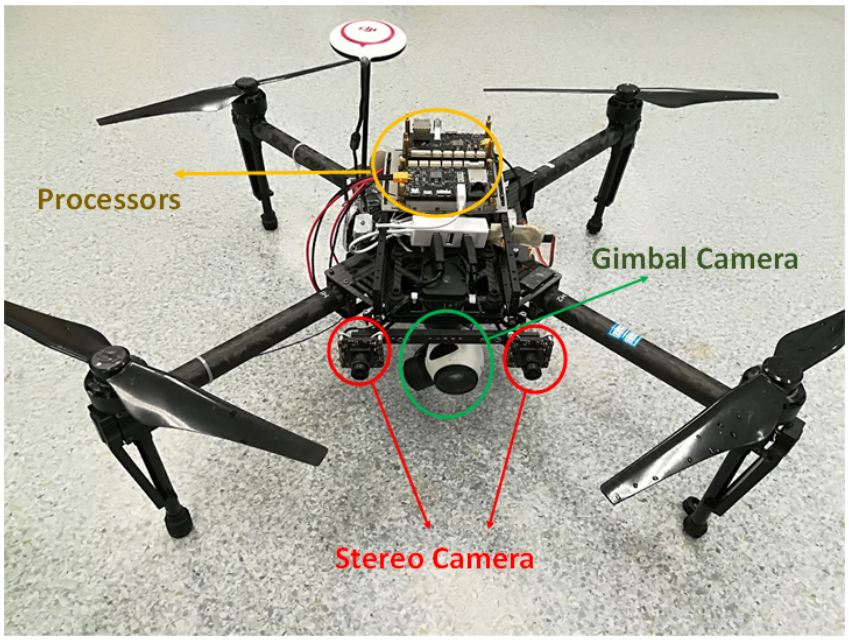


Figure 3.7: The prototype drone of the proposed ACT system

3.2.5 Experiments

In this section, we will evaluate our system on CMU Motion Capture Dataset and real-time action scenes. We compare our system with a state-of-the-art autonomous filming technique "Active Track". "Active Track" is an intelligent flight mode on the DJI Mavic Pro, in which the camera can autonomously keep the distance to follow the target and adjust the camera to place subject on the center of the camera screen. We develop the autonomous cinematography system based on DJI Matrix 100. In the following sections, we offer a detailed discussion on 3D skeleton detection (Sec. 3.1), camera planning (Sec. 3.2) and real-time aerial filming (Sec.3.3).

3D Skeleton Detection

This section we compare our skeleton detection algorithm with Kinect sensors in the indoor environment. The Kinect can achieve accurate skeleton detection with skeletal tracking SDK, so we set the skeleton keypoints from Kinect as the groundtruth. We

evaluate the performance of skeleton refinement in terms of two metrics: 1) Reconstruction Error: The average distance of the 3D point cloud. 2) Viewpoint Estimation Error: Angle difference of the best viewpoint of each frame. We use error distribution in the entire sequence to measure the performance of our system. The larger proportion in the low error case means better performance. We conduct this experiment for slow-paced TaiChi and fast-paced "Gangnam Style" dance .

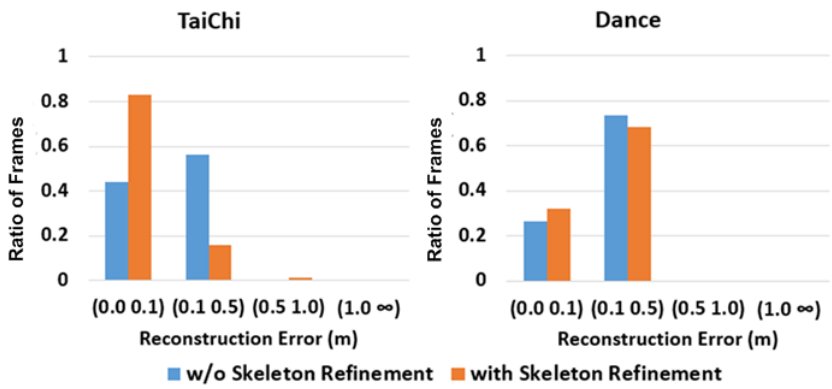


Figure 3.8: Comparison of reconstruction between our methods (w and w/o skeleton refinement).

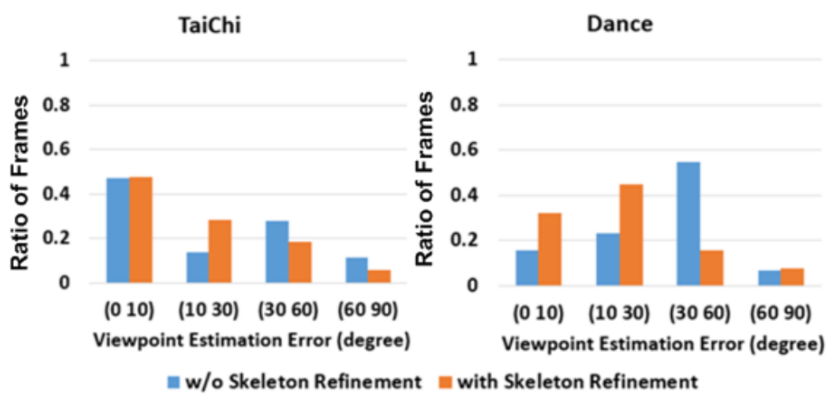


Figure 3.9: Comparison of viewpoint estimation between our methods (w and w/o skeleton refinement).

Fig. 3.8 and Fig. 3.9 show that skeleton refinement can improve the performance of reconstruction and viewpoint estimation. Meanwhile, the improvement of the reconstruc-

tion in fast motion is not obvious as in slow motion, because it is more difficult to achieve motion prediction in faster limbs movement. Even so, we can still achieve the accurate best viewpoint estimation after skeleton refinement. Fig. 3.9 shows that angle error of best viewpoint estimated from our method (with skeleotn refinement) is less than 30 degree in the most cases (more than 70%). In our application, the angle difference within 30 degrees is acceptable, which meets the requirement of view estimation. Moreover, our 3D skeleton detection can work in both indoor and outdoor environments. Fig. 3.10 demonstrate the skeleton detected from our method with skeleton refinement is similar to that from Kinect. In addition, the reconstruction accuracy decreases as the movement becomes fast.

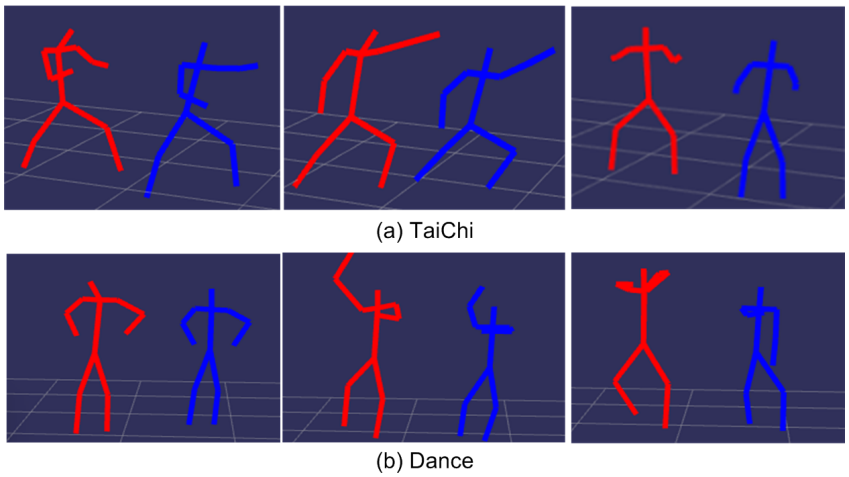


Figure 3.10: Comparison of 3D skeleton between Kinect (blue) and our method (red) with skeleton refinement

Camera Planning

We test our camera planning algorithms on the CMU Motion Capture Dataset (MOCAP), including a set of the motion data and reference video. There is only one subject in each clip of motion data. The 3D motion data is extracted from 41 Vicon markers taped on the subject's body. The motion data is recorded for 6-12 seconds in 120 Hz. In

our application, we only consider 13 markers to represent the human action. We select 10 clips of motion capture data from the MOCAP dataset and classify them as 5 clips of slow-paced motion (e.g. TaiChi, Walk) and 5 clips of fast-paced motion (e.g., Dance, Run).

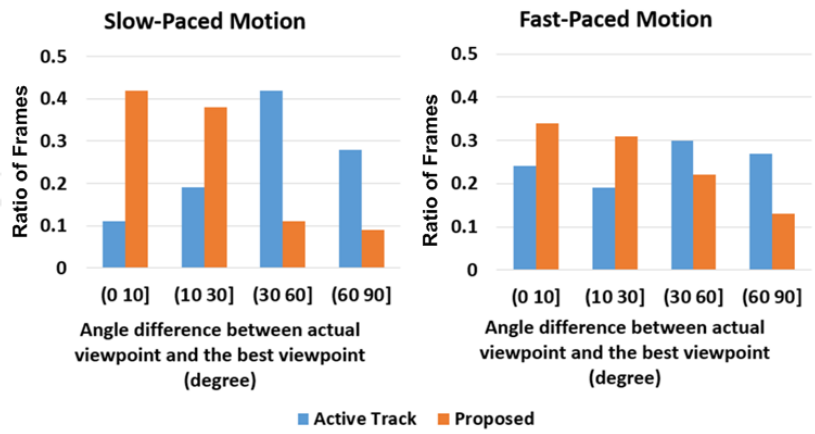


Figure 3.11: The distribution of viewpoint quality.

We define the viewpoint that corresponds to maximum projection of point cloud as best viewpoint. We evaluate the viewpoint quality of each frame by the angle difference between the actual viewpoint and the best viewpoint. The aesthetic effect of each piece of footage is evaluated by the distribution of the viewpoint quality. Fig. 3.11 shows that our drone system can capture the footage from a good viewpoint with higher frequency. This can be explained as the "Active Track" just follows the subject and ignores the pose. Meanwhile, we can see that fast-paced motion (Fig. 3.11(right)) makes it more difficult for the drone to capture the subject from the best viewpoint because of the physical limits of the drone.

Fig. 3.12 compares the camera trajectory from our systems and "Active Track" for Tai Chi. Compared with Active Track, the camera trajectory from our system covers more viewpoints and captures more creative footage.

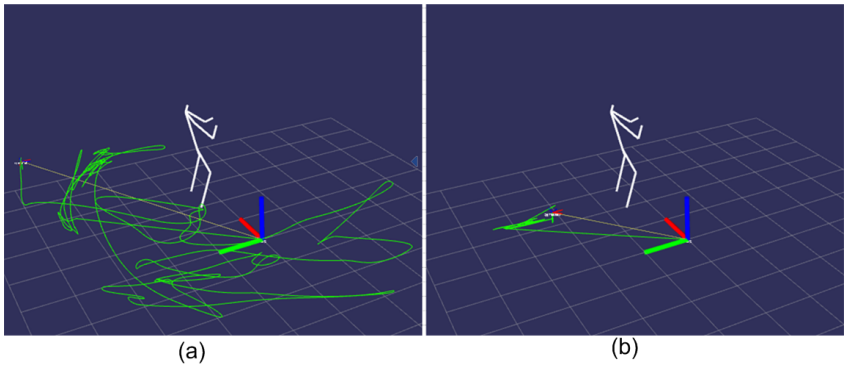


Figure 3.12: The camera trajectory of (a) the proposed system (b) Active Track

Real-Time Aerial Filming

We compare the proposed ACT system with "Active Track" mode of DJI Mavic in the outdoor environment. We initialize the same postion of camera and subject, and then subject performs TaiChi and dance in front of the drone camera. Fig. 14 shows several snapshots of footage captured from both systems. As the human motion goes on, the difference of the footages from both systems becomes more obvious. We can see that the subject in the footage from our system looks more pleasing because of more visible motion and fewer limbs occlusions. The attached videos will provide a more convincing comparison.

3.3 Through-the-Lens Drone Filming

3.3.1 Introduction

It is challenging for beginners to manipulate the remote controller to freely capture the desired viewpoint. While moving control sticks can directly control a drone's motion parameters (i.e. roll, yaw, pitch, and throttle), controlling these parameters do not offer a precise control of the movement of objects in the camera screen. Compared

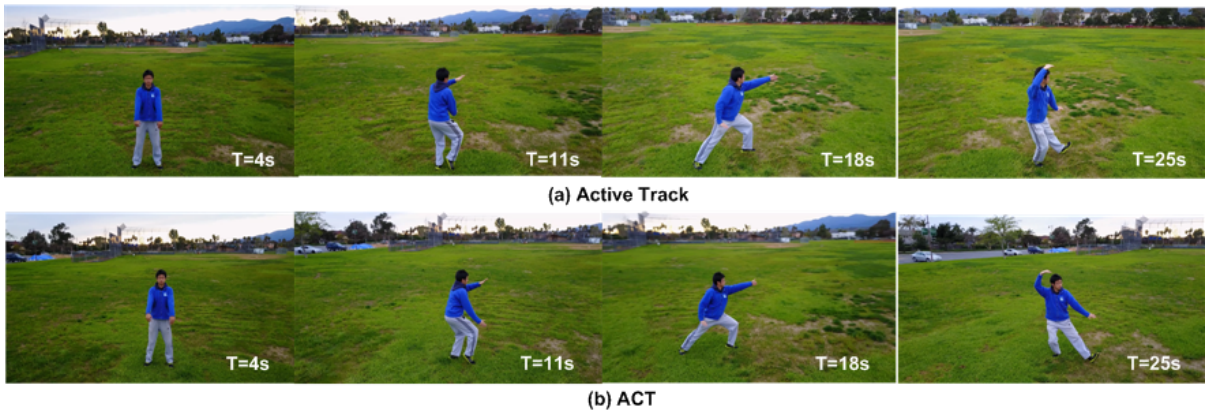


Figure 3.13: Comparison of the footage snapshots between (a) "Active Track" mode of DJI Mavic and (b) the proposed ACT system. We set the same initial relative position between subject and drone. The snapshots of both methods at T=4 s looks similar, but our system can capture more pleasing shots as the TaiChi performance goes on.

with the direct control of the drone's parameters, through-the-lens camera control [28] parameterizes the camera pose in terms of azimuth, elevation, and radius in a subject-centered spherical coordinate system rather than six degrees of freedom (DOF) in a reference-fixed Cartesian coordinate system. Through-the-lens camera control allows a user to drag or zoom in (out) the subject in the image space to adjust the image composition. This control mode greatly simplifies viewpoint control of a moving subject, so it is widely used in action games and 3D animation. Introducing through-the-lens control operations to drone filming can greatly reduce the difficulty of the manual control and allow a cameraman to focus more on the viewpoint selection.

However, it is difficult to apply this subject-centered control mode in real-world scenarios because the subject's position cannot be directly obtained like computer graphics. Some studies [1] [2] [3] localize the subjects by wearable sensors (e.g. GPS, Vicon) to assist in the drone filming, but these sensors are constrained to specific environments. For example, the GPS-based sensors work only in an outdoor environment. In addition, the wearable-sensors-based solutions are ineffective for unknown targets.

Figure 3.14: Overview of the through-the-lens drone filming. (1) The filming scene. (2) Camera View. (3) The preview of 3D human model estimated from camera view. (4) User manipulates the 3D model in the preview to design the viewpoint.

Some researchers [15] [16] [17] [42] use vision-based methods and the prior knowledge of a subject's height to localize the subject. These methods work in both indoor and outdoor environments. Lim et al. [17] localizes the subject based on the position and size of the bounding box estimated from person detection, but the size of the bounding box is sensitive to the person's pose (e.g. Bending over outputs the smaller bounding box than stretching), which affects the localization accuracy. More sophisticated methods [15] [16] [42] adopt 3D human pose estimation to improve subject localization. Because the output of 3D pose estimation loses the absolute scale and depth, the actual height of the subject is required to recover the scale and depth. However, it is not always feasible to request a user to input the height of each subject in unknown scenes.

To address the above challenges, we propose an efficient drone filming mode, called "Through-the-Lens drone filming" (see Fig. 4.10), which is enabled by the following

techniques:

1. An automatic subject localization method without the prior knowledge of a subject's height. We utilize the drone's motion information and the normalized 3D human pose to estimate the subject's height. With the estimated height, we can localize the subject accurately during filming.

2. An effective interaction that allows the user to control the drone by manipulating the virtual camera in the preview of a 3D model. In addition, our system can convert the desired viewpoints in the virtual environments to a physically-feasible trajectory in the true metric space.

To facilitate users' real-time operation, we mount two GPUs (NVIDIA Jetson TK1 and TX2) on a DJI Matrix 100 drone. In addition, we develop an Android app to provide through-the-lens drone control.

The contributions of this paper are three-fold. First, the localization does not require the prior knowledge of the subject's height, which broadens the application of the system to unknown scenes. Second, the proposed through-the-lens drone filming simplifies the manual control for capturing the subject-focused shot and enables the user to customize the viewpoint for moving subjects in real-time. Third, we optimize the implementation of the entire system based on the limited computation resource of a drone platform, including 2D skeleton detection, 3D pose estimation and localization, and camera trajectory planning, and demonstrate the feasibility of running the system in approximately real-time.

3.3.2 Subject Localization based on Visual-Inertial Fusion

In this section, we introduce subject localization based on visual-inertial fusion. Because skeleton-based localization [15] [16] is robust for the varying pose of the subject, and

a full 3D model can facilitate users to operate the camera, we adopt the monocular 3D human skeleton estimation (Sec.III.A) as baseline. As mentioned above, skeleton-based methods require a known height to recover the scale and depth of the normalized 3D pose. Fig. 3.15(a) shows that incorrect height inputs render biased localization. Under an undistorted perspective projection, the localization error is proportional to the error between the actual and the assumed height. Figs. 3.15(b)(c) show that a subject's positioning information from different camera viewpoints may differ when the assumption is inconsistent with the true height, so our intuition is to find an optimal height which can minimize the bias caused by the camera's movement. Compared with the conventional multi-view 3D reconstruction, the scale range of a human subject is limited (we set the height range for an adult between $1.4m$ and $2.2m$ in the proposed system to reduce the search space). In addition, the normalized 3D poses contain an inherent structure among 3D points cloud and thus, in turn, make it feasible to localize the subject from a moving camera, even when the subject is moving. The proposed subject localization includes three steps: 1) monocular camera 3D human pose estimation, 2) scale and depth initialization, and 3) global subject localization.

Monocular Camera 3D Human Pose Estimation

We extract 3D human pose from the images captured by a gimbal camera. This task consisted of two steps: First, we use OpenPose [36] to detect 14 2D joints, including the head, nose, left and right hip, shoulders, elbows, hands, knees, and feet. Second, we use a sequence-to-sequence network proposed in Hossain et al [42] to estimate 3D pose from a sequence of 2D joints. To address incomplete 2D joint estimation caused by occlusion, we use the value in the previous frame to compensate for the missing space of the current frame. Because the input of the network [42] has been normalized to zero mean and a standard deviation of 1, the estimated 3D pose loses the absolute scale and

depth information.

Scale and Depth Initialization

This subsection introduces how to recover the scale and depth of a normalized 3D pose by a moving camera. We start with notation definitions. 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. We assumed that the camera model is weak perspective projection and the subject's movement is smooth during initialization, we define the following optimization function to minimize two terms 1) F: the image projection error from 3D joint locations, and 2) G: the temporal smoothness of the subject's displacement.

$$\begin{aligned}
&\min_{\{\alpha, T_0^c, \cdots T_\tau^c\}} F(\alpha, T_0^c, \cdots T_\tau^c) + \lambda G(T_0^c, \cdots, T_\tau^c) \\
&F = \sum_{t=0}^{\tau} \sum_{n=0}^{N} \left\| p_{t,n}^v - K(\alpha \hat{P}_{t,n}^c + T_t^c) \right\|^2 \\
&G = \sum_{t=1}^{\tau} \left\| (R_t^w T_t^c + T_t^w) - (R_{t-1}^w T_{t-1}^c + T_{t-1}^w) \right\|^2
\end{aligned} \tag{3.12}$$

where τ and N are the size of the temporal window and the number of joints. Based on the assumption of smooth movements, the scale between the true height and the normalized 3D pose during the time interval $[0, \tau]$ shares the same α. Because 3D joints spread in the depth direction is negligible compared to its distance to the camera, we only use one T_t^c to represent the relative position between each joint and camera coordinates at time t. $p_{t,n}^v$ and $\hat{P}_{t,n}^c$ are the n-th 2D joint locations and the normalized 3D pose at time t respectively. K represents the camera projection matrix. λ is the parameter to balance the penalty between projection error and smoothness constraints.

We propose a simple yet highly efficient method to initialize the scale. First, to

quantify the relation between scale α and the camera-subject relative position T^c in camera projection Eq. 3.13, we use the method in [15] to describe $T^c = (T^c_x, T^c_y, T^c_z)$ as expressed in Eq. 4.3.

$$\min_{T^c} \sum_{n=0}^{N} \left\| p^v_n - K(\alpha \hat{P}^c_n + T^c) \right\|^2 \tag{3.13}$$

$$\begin{aligned}
T^c_x &= \alpha(\frac{\gamma}{f_x}\bar{p}^v_x + \bar{\hat{P}}^c_x) \\
T^c_y &= \alpha(\frac{\gamma}{f_y}\bar{p}^v_y + \bar{\hat{P}}^c_y) \\
T^c_z &= \alpha\gamma \\
\gamma &= \frac{\sum_{n=0}^{N} \left\| H(\hat{P}^c_n - \bar{\hat{P}}^c_n) \right\|^2}{\sum_{n=0}^{N} \left\| (p^v_n - \bar{p}^v_n) H(\hat{P}^c_n - \bar{\hat{P}}^c_n) \right\|} \\
H &= \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \end{bmatrix}
\end{aligned} \tag{3.14}$$

where $\bar{p}_x$ and $\bar{p}_y$ are the average values of x and y of 2D joints, $\bar{\hat{P}}_x$ and $\bar{\hat{P}}_y$ are the average values of x and y of the normalized 3D joints. H is a matrix consisting of the focal length f_x and f_y.

Second, because T^c_t is proportional to α in Eq. 4.3, we rewrite T^c_t as $\alpha\hat{T}^c_t$, where $\hat{T}^c_t$ can be considered as the relative position of the normalized 3D pose in camera coordinates. We can solve α by substituting $\alpha\hat{T}^c_t$ into Eq. 3.15. The α in Eq. 3.16 is set as the estimated scale. Note that if the estimated height is beyond a reasonable range ($1.4m$-$2.2m$ in our experiments), or the variance of $\left\| \frac{\tilde{T}^w_{t-1} - \tilde{T}^w_t}{R^w_t \hat{T}^c_t - R^w_{t-1}\hat{T}^c_{t-1}} \right\|$ exceeds a threshold (0.4 in our experiments), we move the temporal sliding window to restart the initialization.

$$\min_{\alpha} \sum_{t=1}^{\tau} \left\| \alpha(R^w_t \hat{T}^c_t - R^w_{t-1}\hat{T}^c_{t-1}) + T^w_t - T^w_{t-1} \right\|^2 \tag{3.15}$$

$$\alpha = \frac{1}{\tau - 1} \sum_{t=1}^{\tau} \left\| \frac{\tilde{T}_{t-1}^{w} - \tilde{T}_{t}^{w}}{R_t^w \hat{T}_t^c - R_{t-1}^w \hat{T}_{t-1}^c} \right\| \tag{3.16}$$

Global Subject Localization

Once we finish scale initialization, we can estimate the subject's global position based on Eq. 3.17 for the following frames.

$$P_t^w = \alpha R_t^w \hat{T}_t^c + T_t^w \tag{3.17}$$

Discussion

In this subsection, we discuss how to move the camera to optimize the localization performance. Considering that the uncertainty of the depth is a function of the length of the baseline between different views, the drone automatically moves sideways to collect 15 images of a subject within a period of 2 seconds to estimate the height. Our strategy is partially motivated by DJI Spark's "ShallowFocus" mode in which a drone creates the effect of shallow depth of the field from 15 images captured during its automatic rising within 20cm. We do not adopt the strategy of elevating the drone to collect images because the change of elevation is likely to degrade the performance of pose estimation.

Scale estimation can possibly be affected by noisy measurements from the navigation system. We neglect the noise of the rotation because the camera gimbal stabilization system can achieve accurate and consistent rotation measurements. To evaluate the localization performance with respect to different initialization states, we design simulation experiments to evaluate the localization error with respect to different camera displacements, camera-subject distance and different levels of noise. For these experiments, we select 42 downsampled motion capture data (average 140 frames, 8 fps, including standing, jumping, sitting, climbing and walking) from Carnegie Mellon University Motion

Capture Dataset. We set the height of the 3D model as $1.8m$.

First, we tested the localization error when the subject has no displacement in the space during initialization, where the subject's center is fixed to the origin of the world coordinate system. Based on the physical property of the drone's navigation system, we evaluated the localization error when the camera's translational displacements are $0.4m$, $1.2m$ and $2.0m$ respectively and the standard deviation of the positioning noise is $0.00m$, $0.04m$, $0.08m$ and $0.12m$ respectively. Considering the subject's safety and the maximum distance of 3D pose estimation, we set the range of the camera-subject distance as [3-11]m. Fig. 3.16 shows that the larger displacement can improve the localization accuracy. In addition, it is harder to localize the subject if the subject is far away from the camera during initialization. This can be explained that the resolution of the limb decreases when the subject moves away from the camera, increasing the image projection error of the subject. In particular, when the displacement is $0.4m$, the increasing noise impacts the performance more obviously as the camera-subject distance increases.

Second, we tested the performance of localization when the subject is free to move within a region during initialization, where the radius of the moving region is set as [0.3, 0.5, 1.0, 2.0]m. We also set zero displacement (radius = $0m$) as a reference. This comparison focuses on the case when the camera's displacement is $1.2m$ and the standard deviation is $0.12m$. For cases of other displacement and standard deviation values, the trends are similar. Fig. 3.17 indicates that the localization becomes worse as the moving region is widened.

From simulation results, we can draw the conclusion that the localization accuracy is determined by a set of initialization states including the subject's movement, the camera-subject distance, the positioning noise and the length of camera's displacement. The localization error, greater than, say, $1.0m$, will affect the subject's safety and thus cannot be allowed. Therefore, to achieve accurate localization within an allowable range,

we better choose the moment when the subject is fairly static and set a closer viewpoint to launch initialization.

3.3.3 Through-the-lens Camera Planning

In this section, we introduce through-the-lens camera control and trajectory planning. First, we introduce how a user manipulates the 3D preview to design the viewpoint. Second, we describe a novel automatic filming mode to track the moving subject. Third, we present our trajectory planning strategy to handle these tasks.

Through-the-lens Viewpoint Control

This section starts with a short description of the User Interface (illustrated in Fig. 3.18(A)). The 3D model is rendered by OpenGL based on the normalized 3D pose. Our system allows the user to touch the screen to move the camera view while keeping the position of the 3D object fixed. A user can adjust the viewpoint by rotating and zooming the 3D model and command the drone to capture the desired viewpoint. Through-the-lens control includes:

Rotate: Swipe the screen to orbit the virtual camera in the horizontal and vertical direction (illustrated in Fig. 3.18(B)).

Zoom: Spread or pinch the screen to change the field of view of the virtual camera (illustrated in Fig. 3.18(C)).

We denote $(\cdot)_c^w$ and $(\cdot)_s^w$ as the position of the drone (camera) and the subject in the world coordinates respectively. In addition, we use $(\cdot)_c^o$ to describe the pose of the virtual camera in the virtual 3D environment. Once the user publishes the desired viewpoint to the drone, our system will map the state of the camera ($P(x, y, z)$, yaw) from the virtual

3D environment to the true world space as follows:

$$
\begin{aligned}
P_c^w &= R^w(\beta P_c^o + T^c) + T^w \\
yaw_c^w &= R^w yaw_c^o
\end{aligned}
\tag{3.18}
$$

where β is a constant to amplify camera-subject distance for safety.

Subject-Oriented Tracking

The above interface allows the user to manipulate 3D model to set the viewpoint. To facilitate users to track the moving person from the desired viewpoint, we further extend it with an automatic filming mode: subject-oriented tracking. In this mode, once the user sets the virtual viewpoint of the subject, the camera will track the subject from a fixed relative position between the subject and the camera. To this end, our system automatically analyzes the subject's skeleton to estimate its orientation. Considering that the direction of two shoulders are normally parallel to the ground, we denote $(\cdot)^s$ as the subject-oriented coordinates, where three axes can be defined as follows:

$$
\begin{aligned}
z^s &= z^c \\
x^s &= \frac{p_{rs}^c - p_{ls}^c}{norm(p_{rs}^c - p_{ls}^c)} \times z^s \\
y^s &= z^s \times x^s
\end{aligned}
\tag{3.19}
$$

where p_{ls}^c and p_{rs}^c denote the 3D positions of the left shoulder and the right shoulder in the camera coordinates. z^c denotes the z-axis of the camera coordinates. The rotation matrix R^{cs} from camera to subject coordinates is described as (x^s, y^s, z^s). Once the user sets the tracking viewpoint, our system records the pose of the virtual camera P_c^s and yaw_c^s in the subject coordinates. The corresponding tracking viewpoint in the world

space can be expressed as follows:

$$\begin{aligned} P_c^w &= R^w(\beta R^{cs} P_c^s + T^c) + T^w \\ yaw_c^w &= R^w R^{cs} yaw_c^s \end{aligned} \tag{3.20}$$

Trajectory Planning

This subsection discusses generation of a feasible path given the customized viewpoints. First, we require the camera to move along the spherical surface centered around the subject P_s^w to achieve visual-pleasing footage and avoid collision with the subject. Therefore, we adopt Spherical Linear Interpolation (Slerp) [43] to uniformly interpolate a set of intermediate waypoints between the current position P_{now}^w and the desired position P_{des}^w along an arc. The interpolated points are described as follows:

$$\begin{aligned} P_{c,i}^w = &\frac{\sin((1-\frac{i}{N})*\theta)}{sin\theta} * (P_{c,now}^w - P_s^w) \\ &+ \frac{\sin(\frac{i}{N}*\theta)}{sin\theta} * (P_{c,des}^w - P_s^w) + P_s^w \quad i = 1, ..., N \end{aligned} \tag{3.21}$$

where θ is the angle between $P_{c,now}^w - P_s^w$ and $P_{c,des}^w - P_s^w$, and N is the number of interpolated points. In particular, if P_s^w is in the middle of a line between $P_{c,now}^w$ and $P_{c,des}^w$, Eq. 3.21 will be reduced to a linear interpolation, rendering that the camera moves across the subject. In order to keep a safe camera-subject distance, we add the midpoint $P_{c,m}^w$ of the semicircular arc as the interpolated point. After that, we use the same algorithm as Sec. 3.1.4 to perform trajectory planning for each waypoint.

3.3.4 System Architecture

The system architecture is shown in Fig. 3.19. In the perception module, we extract the normalized 3D skeleton from the result of 2D skeleton detection. We estimate the scale by fusing the normalized skeleton and motion data from the drone's navigation

system. After the scale is estimated, we can localize the subject in the world space. In the planning module, the system receives the virtual camera pose from the mobile device and estimates the user's desired viewpoint. Then the trajectory planning converts the waypoints to a feasible trajectory. The drone is commanded to fly along the trajectory and capture the footage.

Table 3.2: Runtime of Different Modules

GPU	Module	Runtime(ms)
TX2	2D Skeleton Detection	218.47
Manifold	2D-to-3D Estimation	37.44
	Scale Estimation	28.16
	Subject Localization	9.40
	Viewpoint Estimation	12.09
	Trajectory Planning	33.87

The system hardware is similar to the Sec.3.1.5. The difference is that we remove the stereo camera.

Table 4.1 shows the runtime of different modules for each frame. We deploy different modules to the two GPUs based on their computation complexity. More precisely, one GPU is dedicated for 2D skeleton detection, and the other GPU covers the rest of the computations. Both GPUs 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. The system takes about 300ms to respond to the user's input, which is sufficiently fast for our filming application.

3.3.5 Experiments

Subject Localization

In this section, we test the localization accuracy of our system on 8 persons ($1.6m$-$1.9m$) in real filming. We set the camera-subject distance during initialization as $5m$ and

allow the subjects to move. Fig. 3.21 illustrates that our system can achieve sufficient location accuracy (error is less than $1.0m$) in real scenes within $7m$ camera-subject distance. The localization bias becomes more obvious when the camera-subject distance is farther than $7m$. This trend is quite intuitive as decreasing the subject size in the image increases the difficulty of 2D skeleton detection, where the incorrect 3D skeleton further degrades the localization accuracy.

Camera Planning

In this section, we evaluate the footage captured from two modes: through-the-lens viewpoint control and subject-oriented tracking. We start with subject-oriented tracking in the simulation. We use the distance between the current and the desired camera position to measure the tracking error. We tested 3 motion capture data (walking, dancing and Tai Chi, average 1300 frames, 30 fps) from the CMU Motion Capture Dataset. We set the height of the 3D model as $1.8m$ and the maximum speed and acceleration as $1.5m{\cdot}s^{-1}$ and $1.0m{\cdot}s^{-2}$, respectively. Meanwhile, we set the viewpoint to focus on the frontal direction of the subject by $3.5m$. We use the average speed of the desired viewpoint (ASDV) to describe the intensity of the human movements. Table 3.3 shows that our system can reach the desired viewpoint, with a tracking error less than $1.0m$, for different human movements.

Table 3.3: Subect-Oriented Tracking on Different Movements

Motion Description	ASDV (m/s)	Tracking Error (m)
Walk	0.11	0.12
Tai Chi	0.47	0.39
Dance	1.04	0.53

For the real-world scenes, we compare the actual and desired viewpoints in both modes. Fig. 3.22(a) shows that when the user customizes the viewpoint by zooming

in, rotating horizontally and vertically, the viewpoints of the captured footage match the desired viewpoints of 3D model in the through-the-lens viewpoint control mode. Fig. 3.22(b) shows that the proposed subject-oriented tracking enables the drone to capture the subject from a consistent viewpoint, even when the subject is moving and rotating. The attached demo video confirms high accuracy and impressive performance.

Discussion

The current system works well when the subject's limbs are clearly visible, but it becomes difficult for users to manipulate the drone when the human pose cannot be accurately recognized. Fig. 3.23(a) shows that the limb of the subject is vague due to a long camera-subject distance. In addition, the camera viewpoint also affects the 3D model visualization. Fig. 3.23(b) shows that a sharp angle decreases the body's visibility and makes it difficult to recognize the limbs. These problems are partially due to the fact that our system processes resized images (304x176) to reduce the computation delay. We plan to compress the current 2D skeleton network to process a larger-size image and to perceive a greater range.

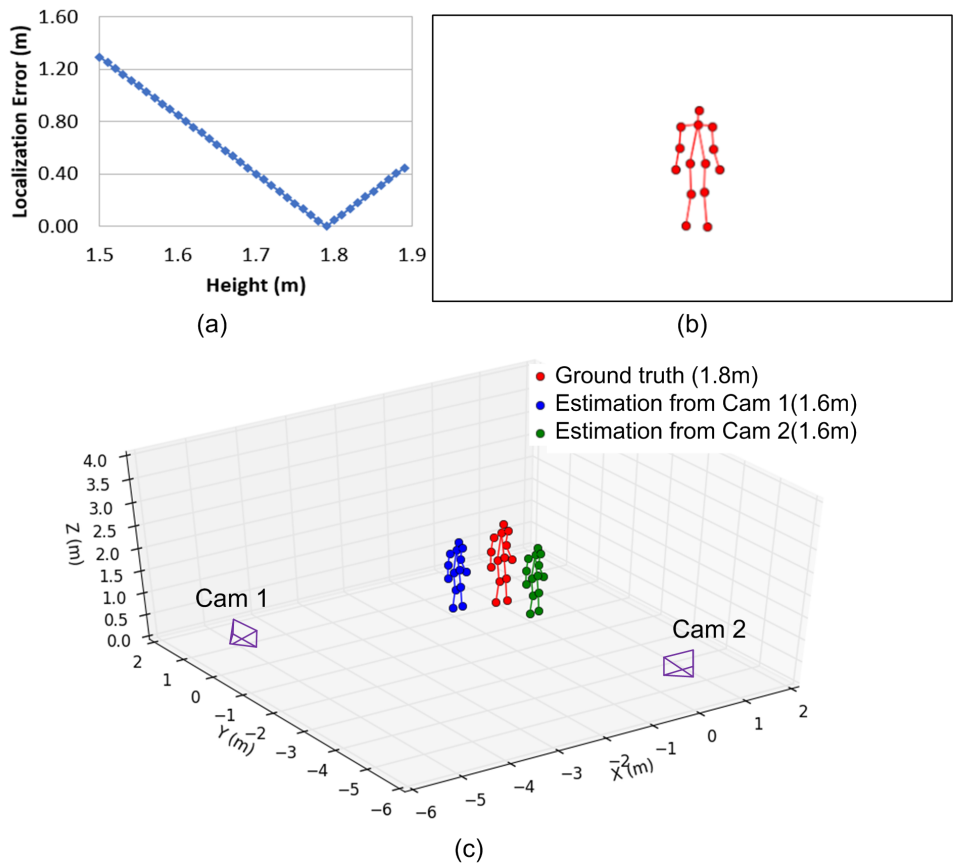


Figure 3.15: (a) The localization error for a person with $1.8m$ height standing in front of a camera (x- and y-focal length is 380 pixel without distortion) by 5m. The x-axis represents the height guess and y-axis represents the error of localization in depth. (b) The camera view captured from Cam 2. (c) A static subject with $1.8m$ height stands in the groundtruth position (red skeleton). The blue and green skeletons are estimated based on $1.6m$ assumption from Cam 1 and Cam 2 placed 5 meters away from the subject in different directions. The estimated position from different viewpoints differs from each other.

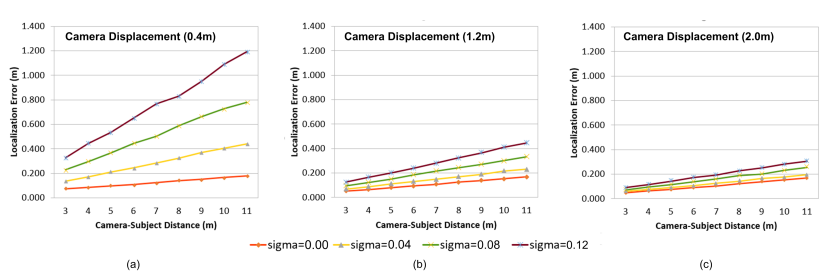


Figure 3.16: Localization error in terms of different initialization states (camera-subject distance, noise levels and camera displacement)

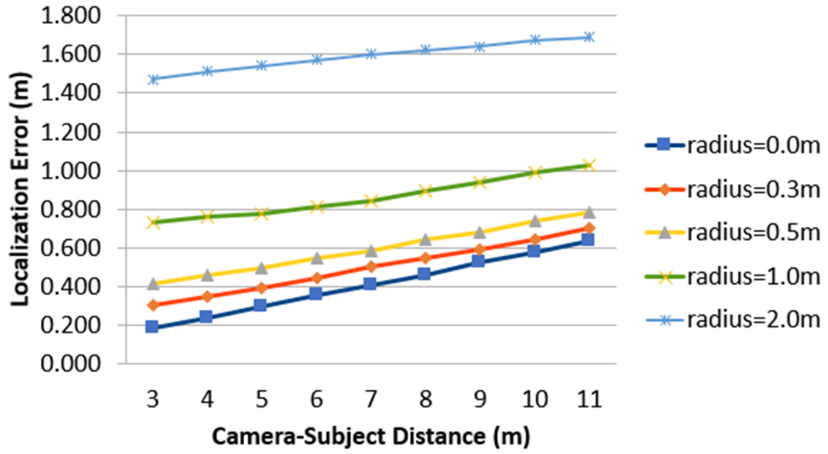


Figure 3.17: Localization error with respect to different initialization states (subject's moving regions and camera-subject distance).

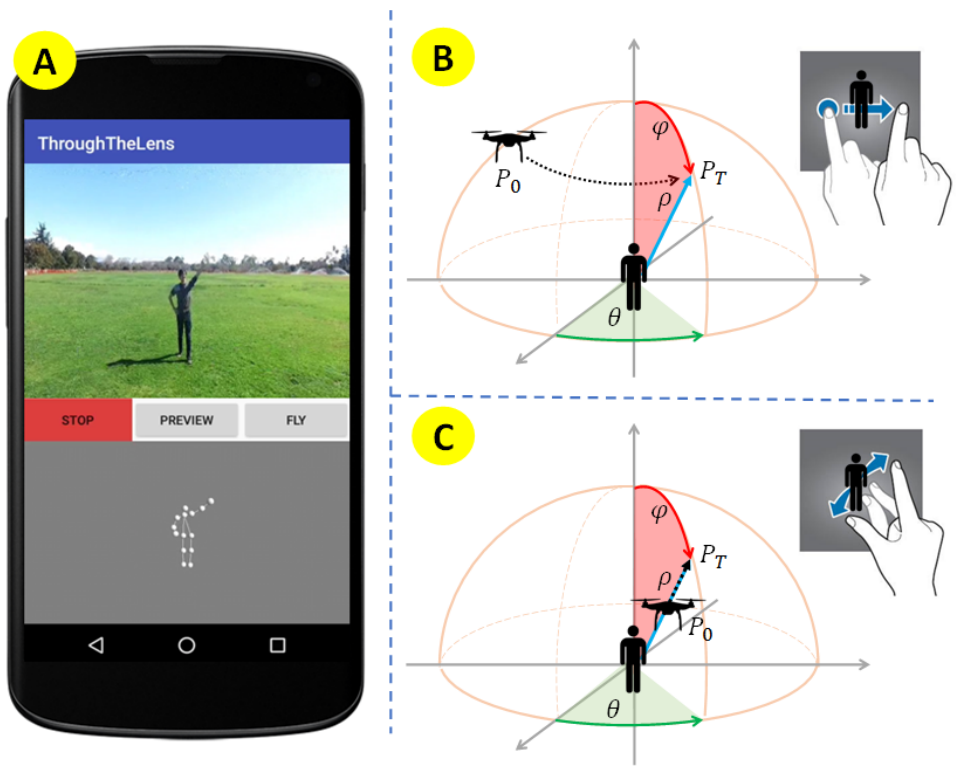


Figure 3.18: (A) User Interface of the through-the-lens viewpoint control is consisted of the camera, including a camera view (the upper window) and a 3D model preview (the lower window). The user moves the virtual camera by (B) rotating the 3D model or (C) zooming the 3D model in the 3D model preview window.

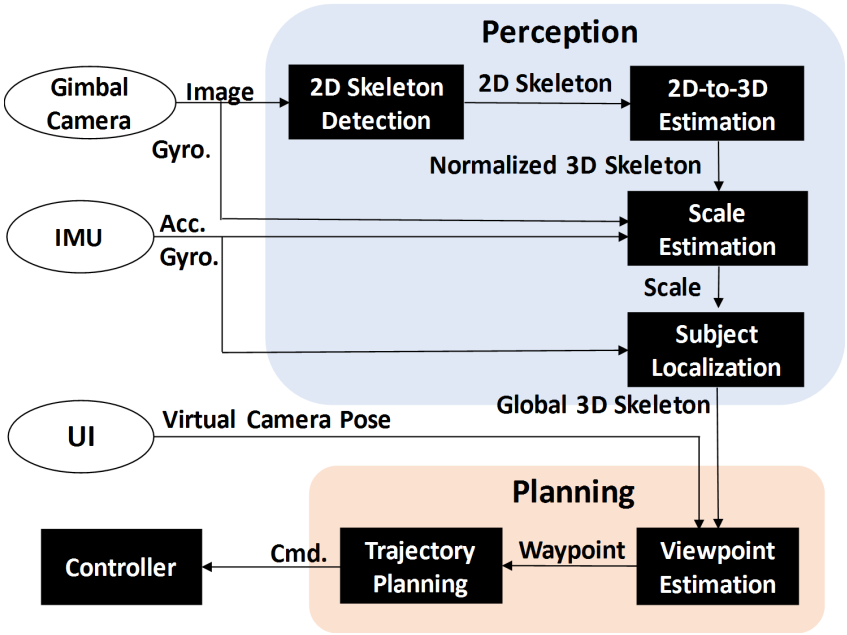


Figure 3.19: The architecture of the system

Figure 3.20: The prototype drone based on through-the-lens control

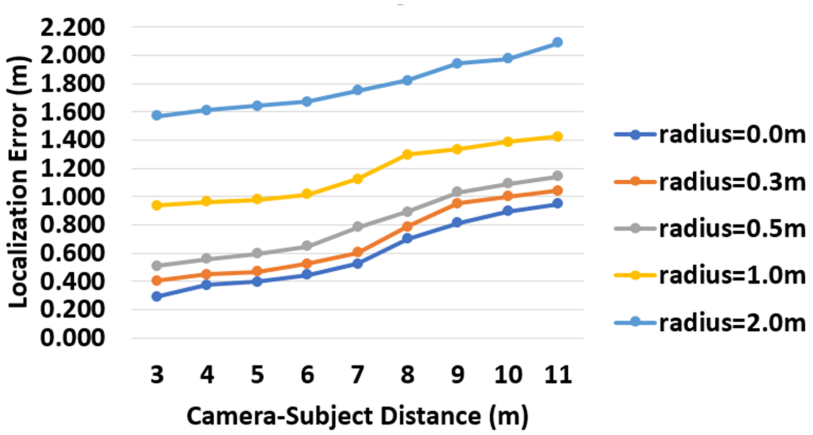


Figure 3.21: Localization error in terms of different moving regions (during initialization) and camera-subject distance (after initialization).

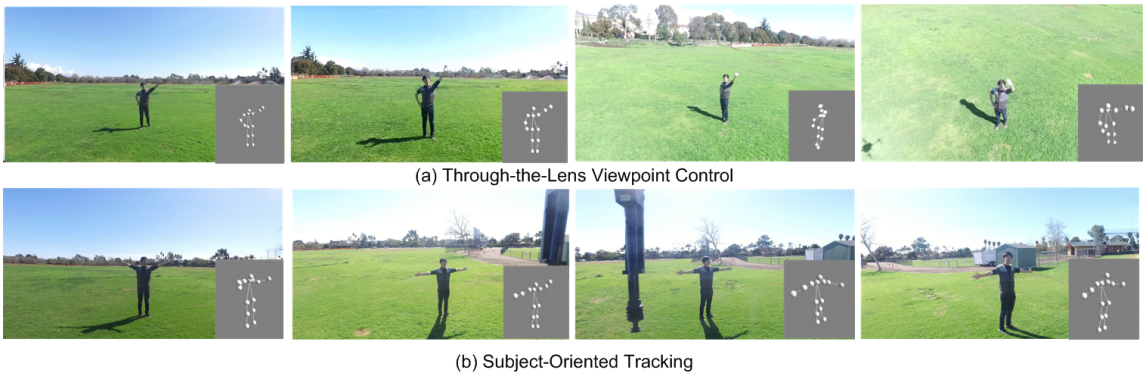


Figure 3.22: The actual viewpoint in real-world filming and the desired viewpoint of the 3D model (the subfigure on the right-bottom). (Top) The snapshot of through-the-lens viewpoint control. The user controls the drone by manipulating the 3D model. (Bottom) The snapshot of the subject-oriented tracking. The user sets the desired viewpoint as the front-right direction in the 3D preview window (first from the left), and then the drone camera keeps tracking the subject from the customized viewpoint.

Figure 3.23: The viewpoint and distance affects the 2D skeleton detection, making it difficult for user to visualize the 3D model.

Chapter 4

Guiding the Camera based on Imitation Learning

In this section, we focus on the problem of capturing the professional videos, which create a film look via specific viewpoint transition. It is difficult to handcraft a metric to reproduce the similar effect. 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". [44] and [6] learned a recurrent decision tree to automate basketball game broadcasting with a pan-tilt-zoom (PTZ) camera. [8] 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 [45, 46, 47] 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 [48, 49]) 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 continue to focus on filming videos containing one subject. We present our learning framework by three levels: First, considering that the cinematic videos create visual-pleasing effect by a specific pattern of viewpoint transition, the network receives the previous viewpoints and learns to predict the next camera viewpoint, i.e., viewpoint-to-viewpoint camera planning. Second, we further analyze the impact of foreground and background on the future camera motions, and design a camera planner which incorporates the video contents and previous camera motions to predict the future camera motions that enable the capture of professional videos, which we call observation-to-control camera planning. Finally, we propose a more efficient framework "one-shot imitation filming", which can imitate a filming style by seeing only a single demonstration video of the same style. Compared with the previous framework, "one-shot imitation filming" does not need to train multiple style-specific models to imitate different filming styles. This advantage can enable the camera agent to quickly learn to generalize the unknown style of a given video to the new situation.

4.1 Related Work

Imitation Filming: Imitation filming is essentially a data-driven autonomous camera planning. [44] and [6] directly use the video clips of basketball games to imitate professional filming for team sports. [8] 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.

Filming Style Characterization Filming styles characterization have been well studied in multimedia community. Rath et al. [50] proposed a four-parameter linear global motion model to describe the camera motion, i.e. pan and zoom. Bhattacharya et al. [51] presented a discriminative representation of video shot which can effectively distinguish among eight cinematographic shot classes: aerial, bird-eye, crane, dolly, establishing, pan, tilt and zoom. Li et al. [52] constructed a videography dictionary to represent the foreground and background motion of each video clip. However, their video representation based on bag-of-visual-words does not consider the sequential patterns of video content, so it cannot distinguish the long sequence (concatenation) of multiple different foreground/background motions.

Video Aesthetic Quality Assessment: Many studies have been conducted to imitate human's way for evaluating the aesthetic quality of videos. Conventional methods mainly employ handcrafted features including color distribution [53, 54, 55], the rule of thirds [56], simplicity [45, 57], composition [58] and motion [59] for describing the aesthetic quality of a video. Recent works focus on learning deep convolutional neural networks (CNNs) features [60, 61] for the aesthetic quality assessment task [62, 63, 64, 65, 66].

One-Shot Imitation Learning: Several studies have been conducted in the literature for imitation learning from very few demonstration. Duan et al [67] proposed a "one-shot imitation learning" framework to enable the robotic to stack blocks as the desired height of the block towers given a single demonstration. Finn et al [68] proposed a model-agnostic meta-learning (MAML) for better generalization performance on the same task. However, few of them are directly applicable to imitate the filming style in terms of the camera motion. On the one hand, existing one/few-shot imitation learning methods aim to learn a policy to reach the final state. Therefore, their models can learn the intention of the video from the snapshot of the final state. In our filming task, we focus on learning the filming style, which refers to a sequential pattern of the camera behavior (e.g., camera angles, distance to characters and on-screen layout). This requires a representative feature which can communicate the filming style from a video to the agent and retarget the cinematic characteristics to a new scenario. On the other hand, the conventional imitation learning methods requires the video and synchronized action as training data. However, our training data is collected from the website without the associated action variable. Although existing structure from motion techniques can estimate the camera trajectory, the ambiguous scale makes it infeasible to use the sequence of the camera pose to supervise learning the model and drive the camera motion.

4.2 Viewpoint-to-Viewpoint Camera Planning

4.2.1 Preliminary

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:

$$\begin{aligned} z^s &= z^w \\ x^s &= norm(p^w_{ls} - p^w_{rs}) \times z^s \\ y^s &= z^s \times x^s, \end{aligned} \tag{4.1}$$

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.

Shot Definition

We define the subject's appearance on the screen as "shot" [3], 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:

$$s = \{p^v, T^s\} \in \mathbb{R}^5, \tag{4.2}$$

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.

4.2.2 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

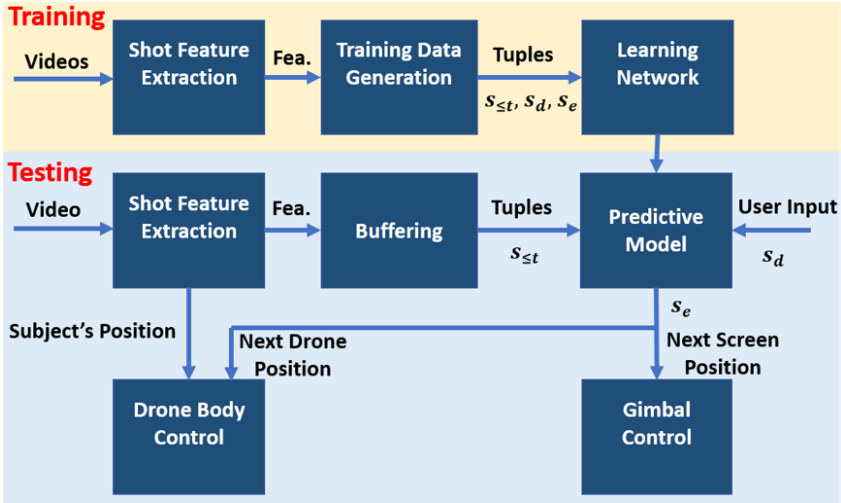


Figure 4.1: The framework of imitation filming.

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 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.

4.2.3 Training

We illustrate the training part of our method in Fig. 4.1(top). First, we extract the shot features from the collected professional videos. Second, we use the sliding windows

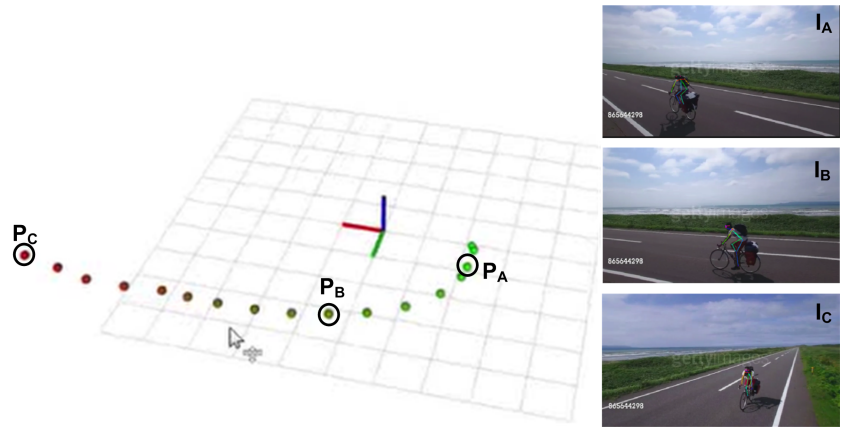


Figure 4.2: The camera poses (P_A, P_B and P_C) are estimated from a sequence of frames (I_A, I_B and I_C). The estimated camera positions are represented in the subject-oriented coordinate system.

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 $p(s_e|s_{\leq t}, s_d)$.

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.

Shot Feature Extraction

In this subsection, we present how to extract the feature Eq. 4.2 from the aerial video. We divide feature extraction into five main steps:

1) We use Openpose [36] 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 [42] to estimate the 3D skeleton from the 2D skeleton. The estimated 3D skeleton is without global position information.

3) We follow [16] [15] 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:

$$\begin{aligned}
\hat{x}^v &= x^v/(width/2) \\
\hat{y}^v &= y^v/(height/2) \\
\hat{x}^s &= x^s/max(x^s) \\
\hat{y}^s &= y^s/max(y^s) \\
\hat{z}^s &= z^s/max(z^s),
\end{aligned} \tag{4.3}$$

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 $s = [\hat{x}^v, \hat{y}^v, \hat{x}^s, \hat{y}^s, \hat{z}^s]$.

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 later) 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.

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 [69, 70, 71, 72], 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:

$$J_\theta = \sum_{(s_e, s_{\leq t}, s_d) \in \mathbb{D}} -\log p(s_e|s_{\leq t}, s_d, \theta), \tag{4.4}$$

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

In our application, the encoder and decoder architecture are based on two long short-

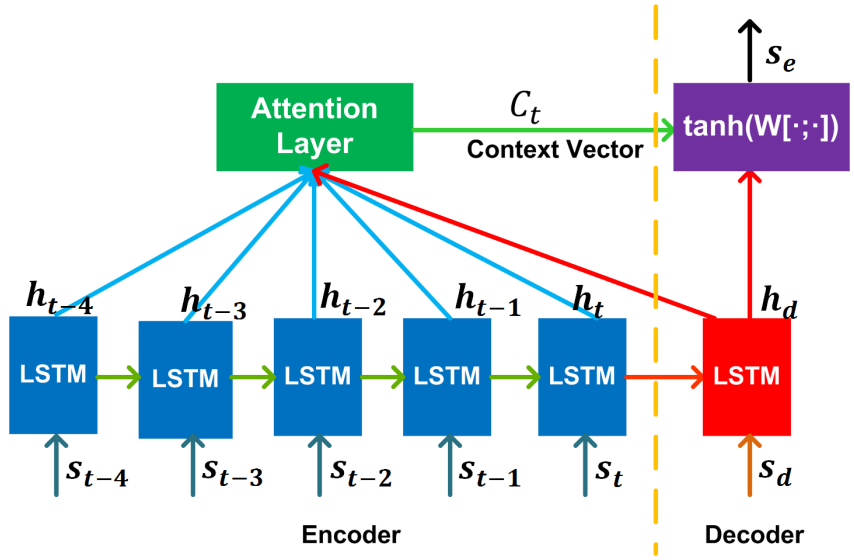


Figure 4.3: The network architecture for imitation filming.

term memory (LSTM) networks [73] 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 [71, 72] to handle possible long-length sequences.

The network architecture is illustrated in Fig. 4.3. 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:

$$C_t = \sum_{k=0}^{K} a_{t-k} h_{t-k}, \tag{4.5}$$

The attention weight a_k is derived by comparing the current hidden state h_d from the

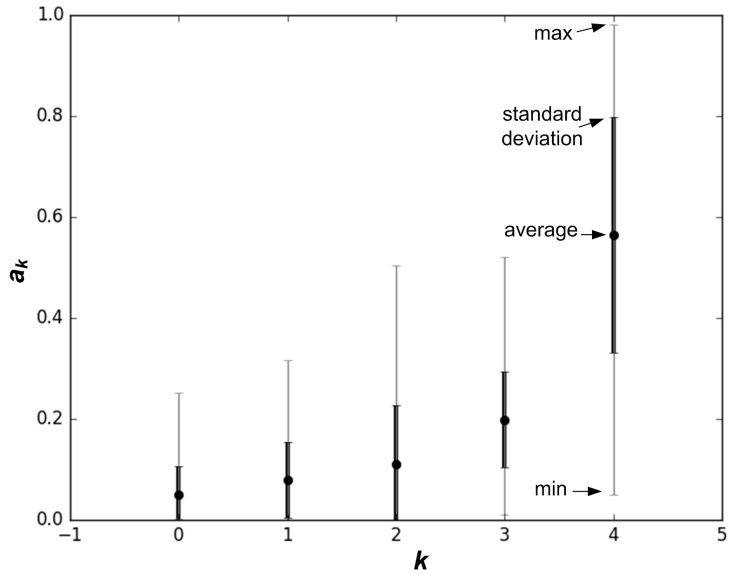


Figure 4.4: The box plot of the 5-steps attention weights of 250 sequences. The attention weight a_k of the last step is the highest and plays the most important role to create a context vector.

decoder with each source hidden state of h_t from the encoder:

$$\begin{aligned} a_k &= align(h_k, h_d) \\ &= \frac{exp(score(h_k, h_d))}{\sum_{k \leq t} exp(score(h_k, h_d))}, \end{aligned} \tag{4.6}$$

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 $score(h_k, h_d) = h_d^T W h_k$ [71]. We analyze the attention weight a_k to understand where the network should focus its attention during decoding. Fig. 4.4 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:

$$s_e = \tanh(W[C_t; h_d]), \tag{4.7}$$

Across all the experiments, we use Adamax [74] to perform the optimization, with a learning rate of 0.0001.

4.2.4 Testing

In this section, we introduce the implementation of the test phase (see Fig. **??**(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 [16] [15] 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 to predict the feature of the next shot s_e.

Finally, we apply the joint quadrotor and camera model used in [1] [**?**] 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. 4.3 before further processing.

Gimbal Control

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

Drone Body Control

We adopt min-snap (second derivative of acceleration) piecewise trajectory planning [40] to guide the drone.

4.2.5 System Architecture

The system hardware is same as the Sec.3.1.5. We deploy different modules to two processors (i.e., TX2 and Manifold) based on their computation complexity. Table 4.1 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 4.1: 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

Figure 4.5: (A, C) The snapshots of two footages captured by our drone system. (B, D) The framing objectives given by users in the User Interface.

4.2.6 Experiments

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

4.2.7 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 4.2: The prediction error of the screen position and relative camera position

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

Result: Tab. 4.2 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 predicted

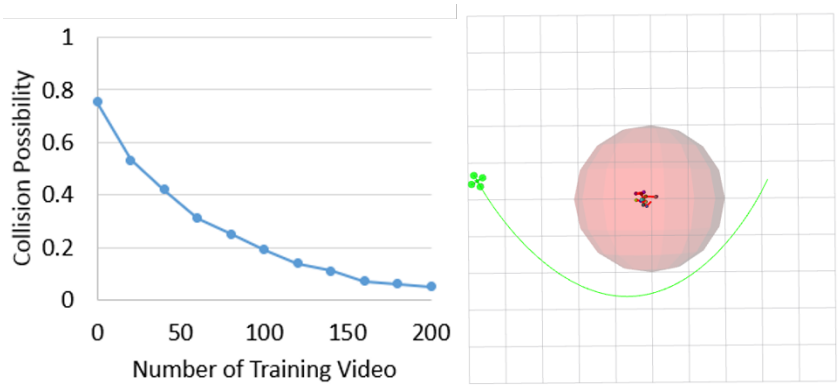


Figure 4.6: Left: The possibility of collision decreases with the increasing training data. Right: The drone can pass by the subject in the test phase.

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.

4.2.8 Is the drone system capable of avoiding collision with the subject?

Experiment: We export one human model from the CMU motion capture dataset [75] 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. 4.6 (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. 4.6 (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 [76], we believe that the drone system can be more robust in avoiding collisions if we feed in more training video captured in different conditions.

4.2.9 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 4.3: 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. 4.5 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.

4.3 Observation-to-Control Camera Planning

4.3.1 Problem Formulation

We aim at a learner that can imitate human experts' policy for camera planning by "watching" a large collection of professional videos from experts. Let d_o denote a distribution of input observation sequence $\dot{o} = (o_{t-M+1}, o_{t-1}, ..., o_t) \sim d_o$, where t denotes a temporal point and o_t is feature vector representing imitation-related information captured from the input video frames $(I_{t-M+1}, ..., I_t)$.

Let π denote a class of policies that our learner is considering. Given a sequence of observation o, each $\pi \in \Pi$ generates a stream of outputs $\dot{c} = (c_{t+1}, c_{t+2}, ..., c_{t+N})$ denoting imitation-related labels during the future period $[t+1, t+N]$. Operationally, c could be 6DOF camera motions, control commands or any other aesthetic quality assessment metrics.

The goal of the learner is to find a policy $\hat{\pi}$ that best imitates the human experts π^*. Thus the learning process is to minimize the imitation loss as:

$$\hat{\pi} = \arg\min_{\pi} \mathbb{E}_{\dot{o} \sim d_o} L(\dot{o}, \pi, \pi^*) \tag{4.8}$$

Three key factors in the formulation that affect the final performance are: 1) the imitation-related information encoded in the input feature representation, 2) the accuracy of the ground truth outputs (i.e. $\pi^*(o)$ for training), and 3) the learning ability of the model for regressing the correlations between the inputs and outputs.

Input feature design: Three components are highly related to imitation-oriented camera planning: 1) The motions (e.g. velocity, position and pose) of the subject, which would determine the camera's moving velocity, trajectory and viewpoint to provide the best view of the subject; 2) The background scene which would affect the composition

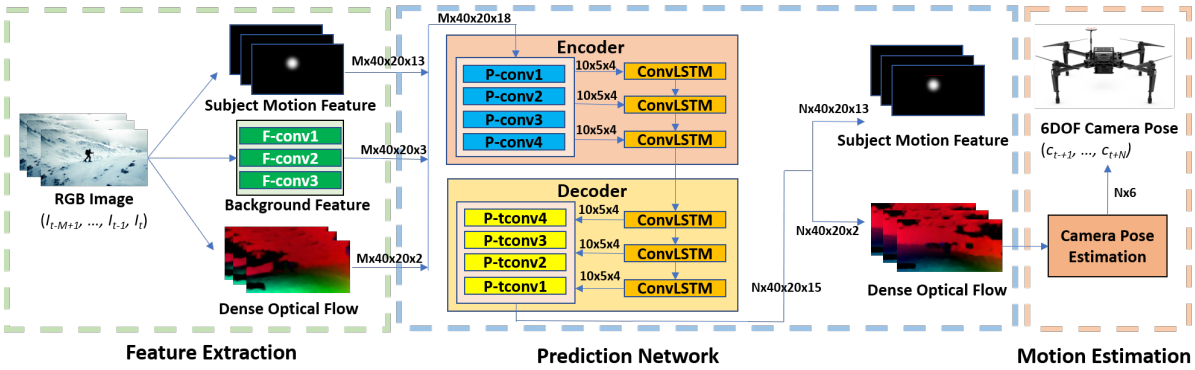


Figure 4.7: Overview of our imitation learning framework. The framework is consisted of three modules: 1) feature extraction, 2) prediction network and 3) camera motion estimation. We illustrate the dimension of the data flow as (time-step × width × height × depth).

of a frame, e.g. it is better to include both the subject and flowering shrubs in a frame, and exclude disordered clutters from the frame; 3) The previous camera motions which could ensure a smooth footage. Imitating an expert to film is a highly complex task and we think it is necessary to jointly consider all the three components and their impacts on the final captured footage. Therefore, in this work we design novel features to effectively represent the three components. Details will be presented in Sec. 4.3.

Output label design: Just like beginners always learn to paint by first copying every stroke of masters' work, we think that training the learner to duplicate the flying trajectories and camera poses provided by the experts should be an effective scheme for imitating filming. Therefore, we define $\dot{c} = (c_{t+1}, c_{t+2}, ..., c_{t+N})$ as 6DOF camera motions. It is feasible to obtain camera motions via visual-inertial navigation on a real camera drone platform equipped with inertial sensors [77]; however, it is impossible to derive absolute camera motions directly from training videos which do not contain inertial sensor data collected from the internet. As a result, directly utilizing camera motions as the output prohibits the usage of enormous professional videos publicly available online for training, and in turn impose great difficulties in collecting sufficient training data.

To alleviate the problem of collecting training data, we utilize dense optical flow of the static regions in a video as the output label of the learner to reflect the camera motion. The camera motions in the testing phase are recovered via a VIN system. Details of converting optical flows to camera motions are presented in Sec. 4.3.

Learning method: A desired learning model should effectively fuse information from multiple inputs and meanwhile spatial and temporal information from the inputs and their correlations with output predictions. To this end, we design on learning model based on the convolution long short-term memory (ConvLSTM) [78] to mine spatial and temporal information of each type of input and to fuse multiple input types in the network.

In addition, the length of the input and output sequences may be not equal. Therefore, we apply sequence-to-sequence architecture (Seq2Seq) [69] to map the input observation to the future camera motion.

4.3.2 Imitation Learning Framework

This section describes the framework of our imitation learning algorithm (see Fig. 4.7), including feature extraction, prediction network and camera motion estimation.

Feature extraction

Subject motion feature: As discussed in the previous section, we desire to encode the subject's motion in the feature representation. To this end, we design the subject motion feature in which the position and pose of the subject are represented using 13 keypoints of a skeleton extracted from OpenPose [36] (see Fig. 4.8 3rd column). The velocity of the subject could be partially reflected by consecutive subject motion maps. However, OpenPose detects only a single pixel for each keypoint, which could be very

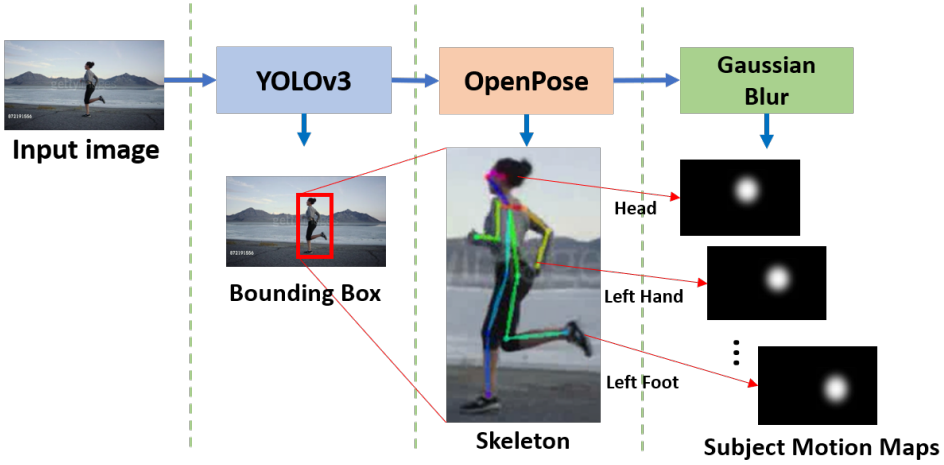


Figure 4.8: The extraction process of the subject motion feature.

sensitive to small geometric changes. To address this problem, we convolve each of 13 keypoints using a Gaussian kennel independently to blur and dilate the keypoint, yielding 13 subject motion maps for a video frame (see Fig. 4.8 last column).

It is common that OpenPose could fail when the size of the subject is too small, yielding incorrect subject motion maps and noisy inputs to the learner. To alleviate this problem, rather than directly applying OpenPose to the entire image, we use subregions of the image containing human detected by the YOLOv3 [79] (see Fig. 4.8 2nd column). Such preprocess step could greatly exclude background clutters and remove distractors for OpenPose. If the YOLOv3 detects the human but the OpenPose fails to detect the keypoints, we will copy the keypoints in the previous frame to this frame. We noticed that this scheme performs well on our benchmark videos despite the size of the subject is small.

The subject motion maps at temporal point t could effectively represent the pose of the moving subject. Concatenating subject motion maps of successive temporal points could reflect the relative motion between the subject and camera. In our experiment, we resize each feature map into 40×20 pixels.

Background Feature: To represent background scenes, we extract CNN features of the original RGB image using a 3-layer convolutional encoder as describe in Tab. 4.4. The final output feature maps are converted to three maps with size of 40×20 pixels.

Table 4.4: Layer parameters of background feature extraction network. The output dimension is given by (width × height × depth). PS: patch size for convolutional and transposed convolutional layers; S: stride. Layer types: C: convolutional

Name	Type	Output Dim	PS	S
F-conv1	C	160x80x32	3x3	2
F-conv2	C	80x40x64	3x3	2
F-conv3	C	40x20x3	3x3	2

Camera motion: As discussed in Sec.4.3.1, we use optical flow to represent the camera motion. In particular, we adopt the dense optical-flow method because it outputs the fixed amount of motion vectors, each of which corresponds to a pixel with the same spatial position in RGB image. This design facilitates learning the spatiotemporal relationship between the subject and the background. Many advanced dense optical flow extraction methods could be used, e.g. FlowNet1.0 [80] and FlowNet2.0 [81]. In this work, we utilize the method proposed by Liu et al. [82] for its high efficiency on a drone platform and sufficient robustness in our task. For each frame, two dense optical flow maps are outputted by [82], representing the horizontal and vertical components of the optical flow for every pixel respectively. The dense optical flow maps are also resized to images of 40×20 pixels for the subsequent processing.

For each frame, we stack the 13-channel subject motion maps, the 3-channel scene maps and the 2-channel optical flow maps to form an 18-channel representation. We concatenate feature representation of M consecutive frames as the input of the prediction network.

Prediction Network

The prediction network is based on a Seq2Seq ConvLSTM model (see Fig. 4.7), including an encoder and a decoder. All the ConvLSTM [78] cells in the encoder and decoder share the same weights.

The encoder first processes each input of M feature maps ($40 \times 20 \times 18$) using 4 convolutional layers, and then feeds the output of the last convolutional layer to the ConvLSTM recurrently. The decoder receives the state vector of encoder conditioned on M inputs and produces predictions for the following N steps. The outputs of the ConvLSTM are further processed using 4 transposed convolutional layers [83] to predict the subject's motion and camera motion. Each subject's motion is represented using 13 subject motion maps and the corresponding camera motion is described using 2 dense optical maps. Thus we split the output of the last transposed convolutional layer to two groups, the first group consists of $N \times 40 \times 20 \times 13$ maps representing the subject's motions of N temporal points and the other group consists of $N \times 40 \times 20 \times 2$ maps representing the camera motions of N temporal points. Details of the prediction network are shown in Tab. 4.5. The selection of M and N is experimentally evaluated in Sec.4.4.

Table 4.5: Layer parameters of prediction network. The output dimension is given by (width $\times$ height $\times$ depth). PS: patch size for convolutional and transposed convolutional layers; S: stride. Layer types: C: convolutional, TC: transposed convolutional, CL: convolutional LSTM cell.

Name	Type	Output Dim	PS	S
P-conv1	C	40x20x8	3x3	2
P-conv2	C	20x10x8	3x3	1
P-conv3	C	20x10x8	3x3	2
P-conv4	C	10x5x4	1x1	1
convLSTM	CL	10x5x4	3x3	1
P-tconv4	TC	10x5x4	1x1	1
P-tconv3	TC	20x10x8	3x3	2
P-tconv2	TC	20x10x8	3x3	1
P-tconv1	TC	40x20x15	1x1	2

We train our prediction network using a combination of two losses: 1) the pixel-wise mean square errors (MSE) between the predicted optical flow and the corresponding ground truth, i.e. $L2(\dot{f}, \dot{f}^*)$, and 2) the pixel-wise MSE between the predicted subject motion feature and the corresponding ground truth, i.e. $L2(\dot{p}, \dot{p}^*)$, as shown in Eq. 4.9.

$$min \quad \alpha * L2(\dot{f}, \dot{f}^*) + \beta * L2(\dot{p}, \dot{p}^*) \tag{4.9}$$

where $\dot{f}$ and $\dot{p}$ refer to the future dense optical flows $(f_{t+1}, ..., f_{t+N})$ and subject motions $(p_{t+1}, ..., p_{t+N})$, respectively. $()^*$ is used to distinguish the ground-truth from the prediction. We use α and β to balance the weight of the prediction of the optical flow and human pose. In our experiments, α and β are set as 1 and 0.3.

The first loss $L2(\dot{f}, \dot{f}^*)$ ensures a high-fidelity imitation of the planned camera's trajectories and poses. The second loss $L2(\dot{p}, \dot{p}^*)$ ensures the proper composition of the picture and the view of the moving subject.

Camera Motion Estimation

This section describes how to estimate the camera motion from optical flow during online filming. We apply a two-stage strategy to generate a camera trajectory: First, we use the learned model to predict the future optical flow $\{f_{t+1}, .., f_{t+N}\}$ for the new input images $\{I_t, .., I_{t-M+1}\}$. Second, according to the optical flow maps at time $[t + 1t + N]$, we identify 800 matching points, based on which we can derive an essential matrix E [84]. We decompose E as Eq. 4.10 to obtain the 3DOF rotation and 3DOF translation of the camera at time $t + i$:

$$\begin{gathered} (p')^T K^{-T} E K^{-1}(p) = 0 \\ E = R[t]_\times \end{gathered} \tag{4.10}$$

where p and p' are homogeneous image coordinates of the start point and end point of the optical flow vectors. K, E, R and $[t]_\times$ refer to the camera intrinsic matrix, the essential matrix, the rotation matrix and the matrix representation of the cross product with the translation vector, respectively.

Because the essential matrix is up to scale (i.e. the scale of the translation is ambiguous), we apply a simple and efficient method to get the scale parameter before autonomous filming: the drone automatically moves backward to collect 10 images of a subject within 2 meters. We estimate the translation (up to scale) from the collected images based on the decomposition of the essential matrix. We calculate the scale parameter by dividing the camera trajectory from the drone's navigation system and the estimated translation. After initialization, the scale, as a constant factor, is multiplied to the camera translation estimated from optical flow as the final translation.

Once N steps of camera motions are obtained, we generate a smooth and feasible trajectory with a min-snap polynomial trajectory planning algorithm [40]. This trajectory will be sent to the flight control module to guide the drone to move.

4.4 One-Shot Imitation Filming

4.4.1 Introduction

In the previous framework "viewpoint-to-viewpoint camera planning" and "observation-to-control camera planning", we utilize imitation learning to train a prediction network from drone video clips, where the prediction network can predict the next location of the drone, given the past locations and video content. However, both techniques suffer from one common drawback: the learned model is style-specific. For example, a policy might have been trained through an imitation learning algorithm to orbit around the subject,

and then another policy would be trained to chase the subject, etc. Each model has only a single-style control policy. In addition, these methods require significant amount of samples and training time for each style. If there are few video samples (even single sample) for one style, the learning model in the auto filming system will suffer from overfitting. This is far from what we desire. Ideally, a sophisticated learning algorithm should enable the camera agent to perform like a skillful cameraman so that the camera agent can reproduce the same filming style from a given demo video without long time to practice, which we call one-shot imitation filming.

In this work, we aim to design a one-shot imitation filming framework, including the following techniques to address the above problems:

First, the filming style feature is required to represent the sequential pattern of the video content. We design a filming style feature extractor which takes the subject's on-screen position, size and orientation and the background's motion field as input and outputs a low-dimensional feature. We apply a cascaded long short term memory networks with attention layers to fuse the temporal information of different inputs. Considering that most filming styles are a combination of finite basic styles, we learn the feature by minimization of the classification error of a classifier, which aims to categorize 5 predefined basic styles (i.e., fly-by, orbiting, fly-through, follow, and super-dolly).

Second, the filming style is required to transferred from a demo video to a video being recorded. We design an improved learning strategy based on meta-learning. Inspired by the fact that the prior experience of imitating basic filming style enables the agent to imitate the complex filming style, we focus on learning the transferring ability of 5 basic styles under the different circumstances. To enable the agent to read the sequential pattern of camera behavior from the style feature, we design a multi-task loss function to train the model by minimizing the camera motion prediction error of the demo video and content video.

Third, although there is no available camera motion ground-truth from video, we apply a simple and efficient method to represent the necessary control variable for our applications. Because we focus on imitating the relative motion between the subject and the camera, we leverage the angular speed and linear velocity direction of the camera and the subject's on-screen size to guide the camera movement. These control variable can be obtained from the video by existing structure from motion and human detection techniques. We can further convert these variables into the actual motion by incorporating with odometry information during the online filming.

We deploy our model to a real drone platform and the real demo shows that our drone cinematography system can successfully achieve one-shot imitation filming. We analyze the impact of different inputs and compare the proposed method with several baselines. Experimental results demonstrate the superiority of our method to conventional baselines. To the best of our knowledge, this is the first work to analyze and design features for representing filming styles and re-apply the cinematic characteristics to a new scenario through an improved one-shot imitation learning framework.

We detail one-shot imitation filming in Sec. 4.4.2. We present the experimental results to evaluate our system in Sec. 4.4.3. Finally, we give the conclusion in Sec. 4.4.4.

4.4.2 One-Shot Imitation Filming

Problem Formulation

In this work, we focus on imitating the human motion video with one single person in the scene.The filming style refers to the sequential pattern of cinematic characteristics of a video (camera angles, distance to the subject, on-screen layout). We can represent the filming style as a sequence of camera poses, which are parameterized to the time variable t in each dimension of 6 DOF as follows:

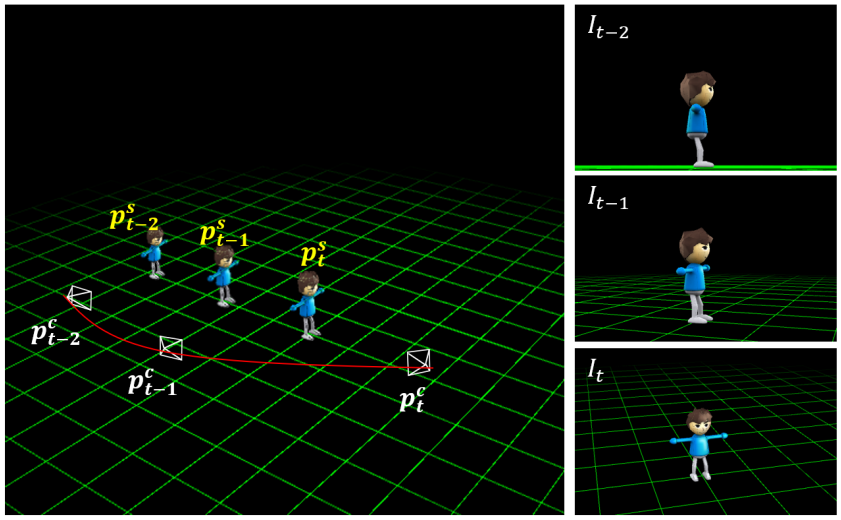


Figure 4.9: The relative motion between the camera and the subject influences the cinematic characteristics of the video. p^c_t and p^s_t represents the pose of the camera and the subject in the world coordinates at time t, respectively.

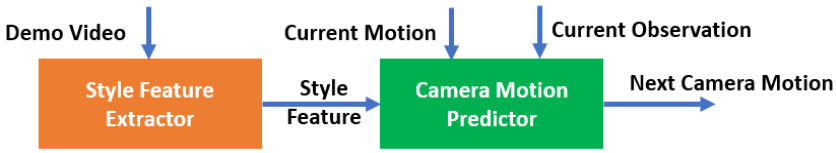


Figure 4.10: Our proposed framework for one-shot imitation filming.

$$
\begin{aligned}
style &\sim \{a_t\}_{0:T} \\
a_t &= \{x^c_t - x^s_t, y^c_t - y^s_t, z^c_t - z^s_t, roll^c_t, yaw^c_t, pitch^c_t\}
\end{aligned}
\tag{4.11}
$$

where $(.)^c$ and $(.)^s$ are the pose of the camera and subject in the world coordinates respectively, and T is the duration of a video clip.

We aim to learn a camera agent, which can adaptively predict the next camera motion (i.e. a_{t+1}) based on the current observation o_t and the camera motion a_t so that the capture video conforms to the filming style of the demo video $g_\phi(d)$.

$$
a_t = f_\theta(a_{t+1}|o_t, a_t, g_\phi(d)) \tag{4.12}
$$

As Fig. 4.10 shows, our objective is to learn two models:

1) The style feature extractor $g_\phi()$, which can encode cinematic characteristics of a variable-length video clip to a low-dimensional fixed-length feature vector.

2) The camera motion predictor $f_\theta()$, which can predict the next camera poses given the style feature of the demo video and the current observation and camera poses.

This section is organized as follows: we start by presenting the filming style feature and detail the design of our cinematic feature extractor. We then describe the camera motion predictor, followed by the camera pose estimation in the field test. Finally, we introduce an segmentation-based method to handle the input video with the mixed style.

Filming Style Feature

The filming style is closely related to the relative motion between subject and camera, but we cannot obtain the actual trajectories of the camera and subject from the video. Existing solutions rely on the strong priors on the content of the scene. Because the relative motion between the camera and the subject influences the cinematic characteristics of the video, we utilize the appearance change of the foreground and background to represent the filming style feature instead. For the foreground, we utilize the subject's on-screen position, size and orientation to represent the camera angels and distance to the subject. The static background is an important reference to identify the camera movement, especially when the subject keeps relatively still to the camera. For example, when a camera is orbiting around a spinning dancer, the subject's on-screen appearance may look relatively static to the camera but we can identify the orbiting pattern of the camera from the background.

The design of the filming style features is guided by

1) the efficiency of the feature extraction that allows the time-critical applications

2) the feasibility of encoding the temporal correlations between the camera motions

and the scene contents into fixed-length vector.

3) the distinguishability of style feature in terms of different filming styles.

Feature Design **Efficiency**: We downsample the input video as 4fps to reduce the duplicate information. We extract the foreground/background feature as follows:

Background: The motion pattern on the static background can reflect the camera movement, so we utilize the sparse optical flow method to represent the background.

Foreground: The subject's appearance (i.e., the on-screen position, size and upper-body orientation) includes the information of the camera angles, distance to characters and on-screen layout. We utilize the person detection and human upper-body orientation jointly to extract the foreground feature. Although the skeleton positions on the 2D image implicitly embed the above information, the 2D skeleton detection is much more time-consuming. More importantly, the increased dimensionality will cause "curse of dimensionality", which makes it more difficult to learn a model from limited dataset.

Feasibility Since our neural network needs to handle demo videos with variable lengths, long short term memory (LSTM) network [73] is a natural operation due to its ability to map the variable-length temporal signal to fixed-dimensional vector. In addition, inspired by that the snippet embedding in [85] can enhance the temporal representation, we apply a two-state cascade temporal encoding strategy: 1) we adopt a overlapping sliding window to segment the video as multiple snippets and utilize two LSTM networks to encode the foreground and background feature within each snippet as a snippet embedding. 2) We utilize another set of LSTM networks to encode all the snippet embedding of the entire video to represent the video feature.

Distinguishability Because the most filming styles are a combination of several basic styles, we learn a feature extractor by adding a multi-class classifier and minimizing the misclassification error of several predefined basic filming styles. According to [86],

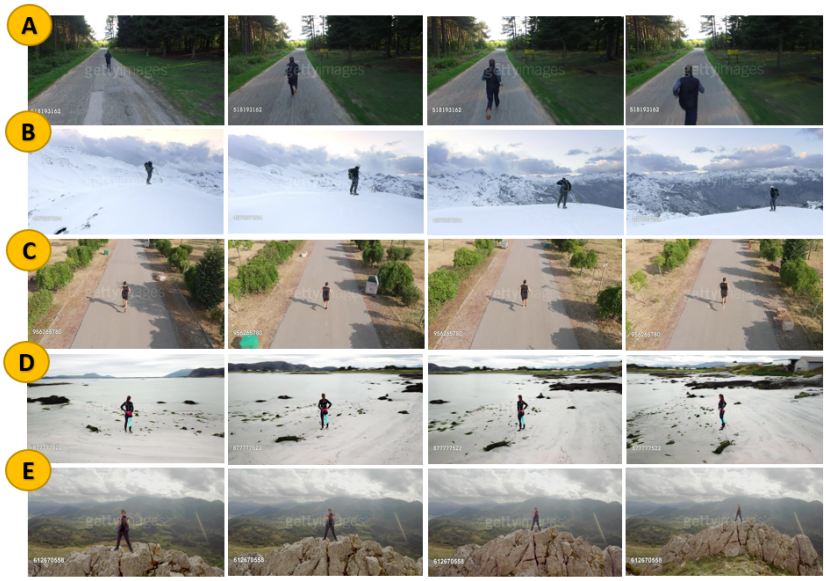


Figure 4.11: Exemplar videos with the style (A) fly-through (B) fly-by (C) follow (D) orbiting and (E) super-dolly.

five widely-used basic filming styles in single-subject drone cinematography include (see Fig. 4.6):

- Fly-through: pass through a subject without rotation.
- Fly-by: fly past the target in a straight line while rotating the camera to keep the subject framed in the shot.
- Follow: follow the subject with the fixed distance.
- Orbiting: rotate around the subject of interest while pointing to the subject.
- Super-dolly: fly backwards, leading the subject.

Implementation In this section , we detail the implementation of style feature extractor based on deep neural network. As the Fig. 4.12 shows, the style feature extractor takes the entire video as input, and produces a fixed-length vector.

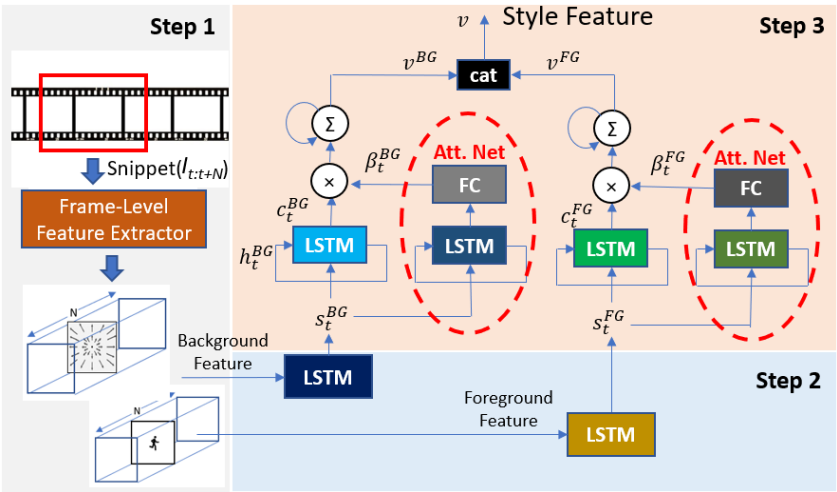


Figure 4.12: The style feature extractor. Step 1: frame-level feature extraction. Step 2: snippet-level embedding. Step 3: video-level embedding

Step 1: Frame-level feature extraction We extract the foreground/background feature for each frame as follows:

Background. We adopt the grid-based motion field [52] to describe the motion between adjacent frames. In details, we densely computed Kanade-Lucas-Tomasi (KLT) tracks [87] over the entire image. The image plane is splitted as a K × L regular grids. We calculate the per-grid velocity $V_{k\times l}$ as the average from multiple KLT tracks intersecting that grid. As suggested in [52], the block size is set to 8×8.

Foreground. We utilize YOLO network [79] and Deep Orientation netwwork [88] to detect the bounding box and upper-body of the subject, respectively. The position and size can be described as a 4-dimensional bounding box, and the upper-body orientation is represented as a 1-dimensional Euler angles.

Step 2: Snippet-level embedding We utilize a sliding window with length N frames (i.e. N=8) to group the features as a snippet. Given a snippet of the foreground/backgroujnd feature, we utilize an auto-encoder [89] based on LSTM networks [78] to learn snippet embedding. We train two encoders for the snippet embedding of background and foreground, respectively. As the sliding window scans the entire video,

a video can be represented as a sequence of the background embedding and a sequence of foreground embedding. The foreground encoder is a LSTM network with 64 hidden units and the background encoder is one with 128 hidden units.

Step 3: Video-level embedding We apply two different feature embedding networks to encode the snippet embeddings of foreground and background, respectively. It is noted that the amount of valuable information provided by different frames is in general not equal. Only some of the frames (key frames) contain the most distinctive information about the style. For example, for the style "fly-through", the snippet captured when the camera is passing by the subject should have higher importance than the snippet when the camera is moving closer to the subject. Based on such insight, we design a temporal attention network to automatically pay different levels of attention to different snippets. Similarly, we apply two parallel temporal attention networks to process the foreground and background independently. Finally, we output the style feature by fusing the temporal attention and the feature embedding. Let $c_t^{FG/BG}$ and $\beta_t^{FG/BG}$ denote the outputs of the foreground/background main networks and the temporal attention value of the foreground/background attention network at each time step t, the style feature v is the concatenation of the weighted sum of the outputs at all time steps T:

$$v = [\sum_{t=0}^{T} \beta_t^{FG} \cdot c_t^{FG} : \sum_{t=0}^{T} \beta_t^{BG} \cdot c_t^{BG}]. \tag{4.13}$$

We implement the feature embedding network and temporal attention network of foreground and background with 4 different LSTM network with 80 hidden units. The final style feature is a 160-dimensional feature vector.

We train the network by feeding the style feature into a 5-dim fully-connected layer p followed with a softmax layer to convert to probability of five basic styles. The loss function is written as Eq. 4.14, including 1) the cross-entropy loss of style mis-classification

2) the L2-norm loss of the foreground temporal attention and 3) the L2-norm loss of the background temporal attention.

$$\begin{aligned} \min_{\phi} \quad & \sum_{c=1}^{C} y_c log p_c(g_\phi(s_{1:T}^{FG}, s_{1:T}^{BG})) \\ & + \frac{\lambda_1}{T}\sum_{t=1}^{T}\|\beta_t^{FG}\|_2 + \frac{\lambda_2}{T}\sum_{t=1}^{T}\|\beta_t^{BG}\|_2 \end{aligned} \tag{4.14}$$

where C is the number of the basic styles (i.e. C=5) and T is the number of the snippets of each video. In this work, we set both of λ_1 and λ_2 as 0.01.

Camera Motion Prediction

In this subsection, we introduce the camera motion predictor, which can re-apply the filming style of the demo video to the new scenario by adaptively predicting the next camera motion based on the current observation. Inspired by the fact that prior experience of imitating basic filming style enables the agent to imitate the complex filming style, we can train the predictor to transfer the basic filming style under different circumstances so that the predictor can quickly learn to generalize the unknown style of a given video to the new situation. The design of the camera motion predictor should answer the following questions:

1) How can we represent the camera motion when building the ground-truth? It is impossible to derive 6 DOF camera motions directly from training videos which do not contain inertial sensor data collected from the internet.

2) How can we transfer the filming style (i.e., the sequential pattern of camera behavior) of the demo video to a video being recorded?

Camera Motion Representation We apply a simple and efficient method to represent the necessary control variable for our applications. First, we utilize the structure

from motion techniques to estimate the camera trajectory (without the scale) from the video. Because each camera pose in the estimated trajectory has one timestamp, we can obtain the angular speed and linear velocity direction in the world coordinates. Second, because the style is essentially determined by the relative position between the subject and the camera, we can use the subject's height to represent the scale of the relative camera motion (if we assume that subject's height would not be changed frequently). Therefore, we use the 3-dimensional angular speed, 3-dimensional linear velocity direction and 1-dimensional the subject's on-screen height to approximate the camera motion relative to the subject as the output label of the network. It is noted that the subject's on-screen height is normalized between 0 and 1 by dividing the pixel-wise the subject's height by the height of the image screen.

Filming Style Transferring Ideally, a context embedding should include the style-related information from the observation and ignore the trivial factors (e.g. background layout). Although different videos with the same style are different in terms of the camera speed, background layout and the subject's motion, they share the same sequential pattern of the camera behavior. Therefore, there exists the temporal matches (the same development progress) between two videos with the same style. Based on this insight, we propose a novel meta-learning strategy to train the predictor (see Fig. 4.13). First we sample two videos (style video and content video) with the same style. We sample a snippet (content snippet) from the content video and find the matching snippet (style snippet) from the style video. The context embedding, which is calculated from the style feature of style video and the content snippet, should predict the next action for the content snippet and style snippet respectively (conditioned on the current action in the content and style snippets). We perform the same training procedure for 5 basic styles,

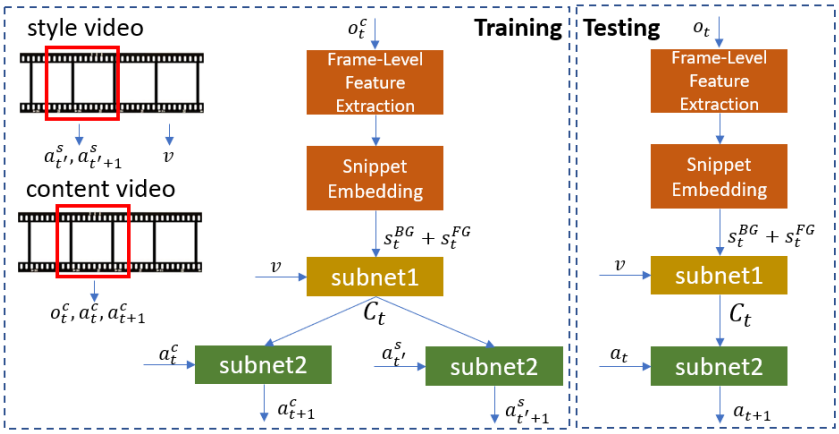


Figure 4.13: Left: Training the camera motion predictor. The snippets cropped by two red bounding boxes are the matching snippets. C_t refers to the context embedding. Right: Testing the camera motion predictor.

and the loss function can be written as follow:

$$\min_{\theta} \quad \|f_\theta(a_t^c, v, o_t^c) - a_{t+1}^{c*}\| + \lambda * \|f_\theta(a_{t'}^s, v, o_t^c) - a_{t'+1}^{s*}\| \tag{4.15}$$

where the $(.)^s$ and $(.)^c$ refer to the variable of style snippet and content snippet. $(.)^*$ indicates the ground-truth camera motion. t and t' are two matching timesteps from two videos. The hyper-parameter λ is set as 0.7.

How to find the matching snippets? There are multiple ways to find the optimal matching patterns between two variable-length sequences. In this work, we utilize the dynamic time warping (DTW) [90] to detect the matching snippet of two videos, while the video is represented as a sequence of concatenation of background and foreground embeddings. Fig. 4.14 shows that the warping path in which the two sequences with style "fly-by" are aligned in time. We can see that each red point on the warping path corresponds to two matching snippets a_i and b_i, which share the similar relative subject's on-screen position.

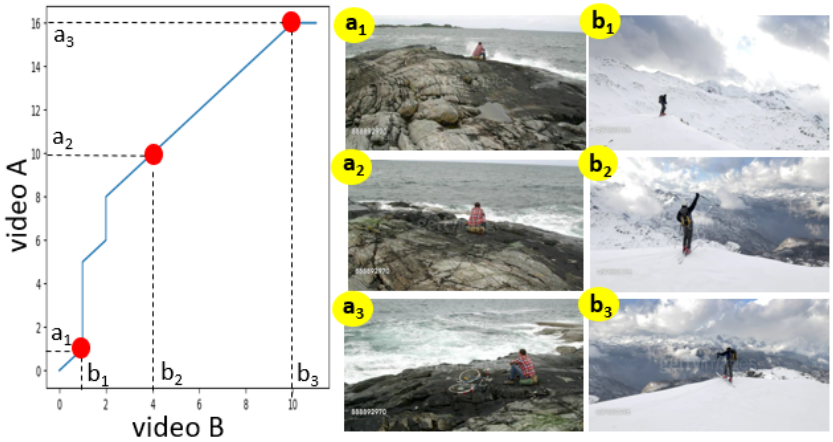


Figure 4.14: Left: the warping path generated from two videos with the same style. Right: the matching snippets in each row share the similar relative subject's position.

Implementation We utilize the deep neural network to implement the camera motion predictor. The camera motion predictor takes the style feature from the demo video, the current observation and camera motion as inputs and predicts the next camera motion. We embed the latest N frames from the camera as the foreground and background embeddings.

The network is constructed by two subnetworks. The first subnetwork, which consisted of two fully-connected layers (128 and 64 hidden units), produces a context embedding given the concatenation of style feature, the foreground and background embeddings, and the second subnetwork, which consisted of two fully-connected layers (32 and 7 hidden units), predicts the next camera motion from the context embedding and the current camera motion.

Although our imitation policy is similar to the conventional one-shot imitation learning [67, 68], we do not follow their end-to-end training strategy to learn the entire model by minimizing the action prediction error. As different filming styles may output similar camera motion (but different timesteps), it may cause the vanishing gradient in the style feature extractor, preventing us to train the network end-to-end. Another feasible

solution is to finetune the entire network end-to-end after separate training, but there is no obvious improvement.

The above training process is based on the assumption that the input demo video has only a single basic style. In the test phase, the input demo video could be a long sequence (concatenation) of multiple basic styles, we need to analyze the motion coherency of the video and decompose the video into a sequence of single-style segments. Then we perform imitation filming for each segment orderly.

Camera Motion Estimation

This section describes how to produce the camera motion from the outputs (angular speed ω, linear velocity direction v and subject's scale s) of the prediction network during online filming. In details, we have the drone's position p_t^d and orientation φ_t^d at the timestep t. If the subject's height is known, we can utilize Lim et al's method [17] to localize the subject's position p_t^s based on its bounding box s_t^s. We assume that the subject's movement is smooth and we can use Kalman Filter to predict the subject's location $p_{t+\Delta t}^s$ in the next timestep. As Fig. 4.15 shows, the drone's orientation $\varphi_{t+\Delta t}^d$ in the next timestep can be obtained by adding φ_t^d with $\omega\Delta t$. The drone's position $p_{t+\Delta t}^d$ in the next step is calculated by searching a position on the ray with the direction v to minimize the error between the observed bounding box $s_{t+\Delta t}^s$ on this position and the predicted subject's scale s. This estimated waypoint ($\varphi_{t+\Delta t}^d$ and $p_{t+\Delta t}^d$) will be sent to the actuator to control the drone.

4.4.3 Experiments

In this section, we first introduce the dataset collection, and then describe the experimental setup and the measurement metrics, followed by experimental results.

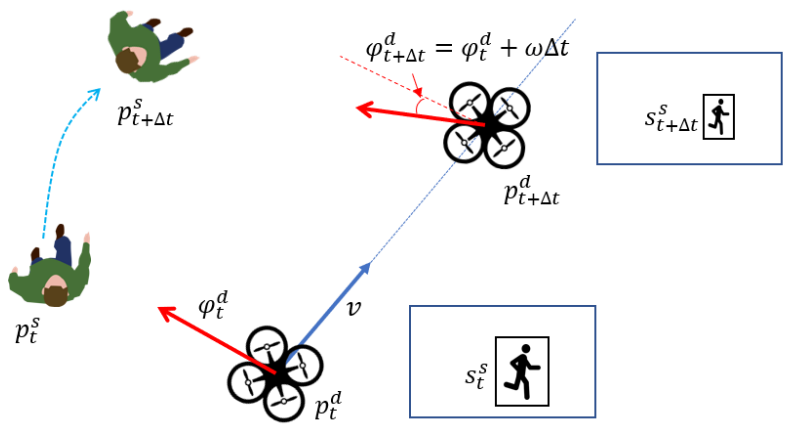


Figure 4.15: The camera dynamically estimates the next camera pose based on its actual observation and the network prediction.

Dataset

We collect the video clips from the website *www.gettyimages.com*, which offers professional photography and videography. Specifically, we used three keywords "*aerial view, one person only, outdoor*" to initialize our search. We excluded the searched video results which contain extremely poor lighting conditions, subjects taking up too small regions and/or being occluded for too long time during the video.

To select the video with the predefined basic styles , we recruit 3 human annotators and asked them to manually label the videos based on the definition of each style. Each video was labeled by 1 annotator and verified and corrected by the other 2 annotators. We will drop the video if it does not belong to any of the basic styles. Eventually, we obtain 146 videos, each of which is around 5-50 seconds long, yielding videos of totally 3218 seconds. Tab. 4.6 shows the statistics of the style annotations in our data.

We resized each video frame to 640x480 and down-sampled the video to frame rate of 4fps to adapt to the actual computation speed. In addition, we provide the ground-truth of camera trajectory (rotation and translation) and subject on-screen information (position, size and orientation). More specifically, we apply the state-of-the-art structure

from motion tool OpenSFM [91] to extract the ground-truth of camera trajectory. The ground-truth of subject on-screen bounding box is detected based on YOLOv3 [79] and the orientation is estimated based on the result of [88], while we manually correct the misidentified skeleton joints to replace the original result.

Table 4.6: Statistics of the style annotations in our data

Style	fly-by	fly-through	follow	orbiting	super-dolly
Videos	21	42	30	28	25
Duration (Second)	452	976	670	587	533

Experimental Setup

We split our dataset into 97 training videos and 49 test videos. The number of videos from the five styles (i.e. fly-by, fly-through, follow, orbiting, super-dolly) are 14, 28, 20, 18, 17 for the training set and 7,14,10,10,8 for the testing set. For each training and testing video, we applied an overlapping sliding window with a length of 8 to generate a set of snippets for background/foreground embedding. The stride of the overlapping sliding window is 4. Accordingly, we generate a total of 1960 training clips and 1002 testing clips. We further augmented the training data by flipping each video clip along the horizontal axis, yielding 3920 training clips. We train our network on the Nvidia Tesla K50c and utilize Adamax [74] to perform the optimization, with a learning rate of 0.001.

We evaluate the performance of our method as follows:

1) We utilize the confusion matrix of the style classification to examine the representation ability of the style feature.

2) We measure the accuracy of the camera motion predictor by calculating the mean square error (MSE) between the predicted camera motion and the ground-truth in terms

of angular speed (rad/s), linear velocity direction and (rad) the subject's normalized scale.

3) We evaluate the imitation performance of the capture video by comparing it with the demo video in terms of four metrics: 1) the on-screen position error, 2 the upper-body orientation error, and 3) motion field average end-point error.

4.4.4 Filming Style Feature

In this subsection, we design the experiments based on our dataset to carefully analyze two factors on the style classification results: 1) input feature selection and 2) attention mechanism. To the end, we design the following baselines: FG-only, BG-only, FG+BG, FG+BG+Att. Details of the four baselines are listed in Tab. 4.7.

Table 4.7: Design of the four network baselines

	FG-only	BG-only	FG+BG	FG+BG+Att
foreground	✓		✓	✓
background		✓	✓	✓
attention				✓

Because FG-only and BG-only baselines only use one branch network, we double the number of hidden neurons such that they have the same number of parameters as other two baselines. Fig. 4.16 shows that the foreground and background have complementary relationship for filming styles classification. For example, the orbiting style depends more on the background than the foreground, while it is reverse for the follow style. In addition, the baseline FG+BG performs better than the baselines with a single input (FG-only and BG-only), indicating that the combination of both feature can further improve the classification performance. The baseline FG+BG+Att achieves the best performance among all the baselines, which proves that the attention layer is beneficial to process long sequence.

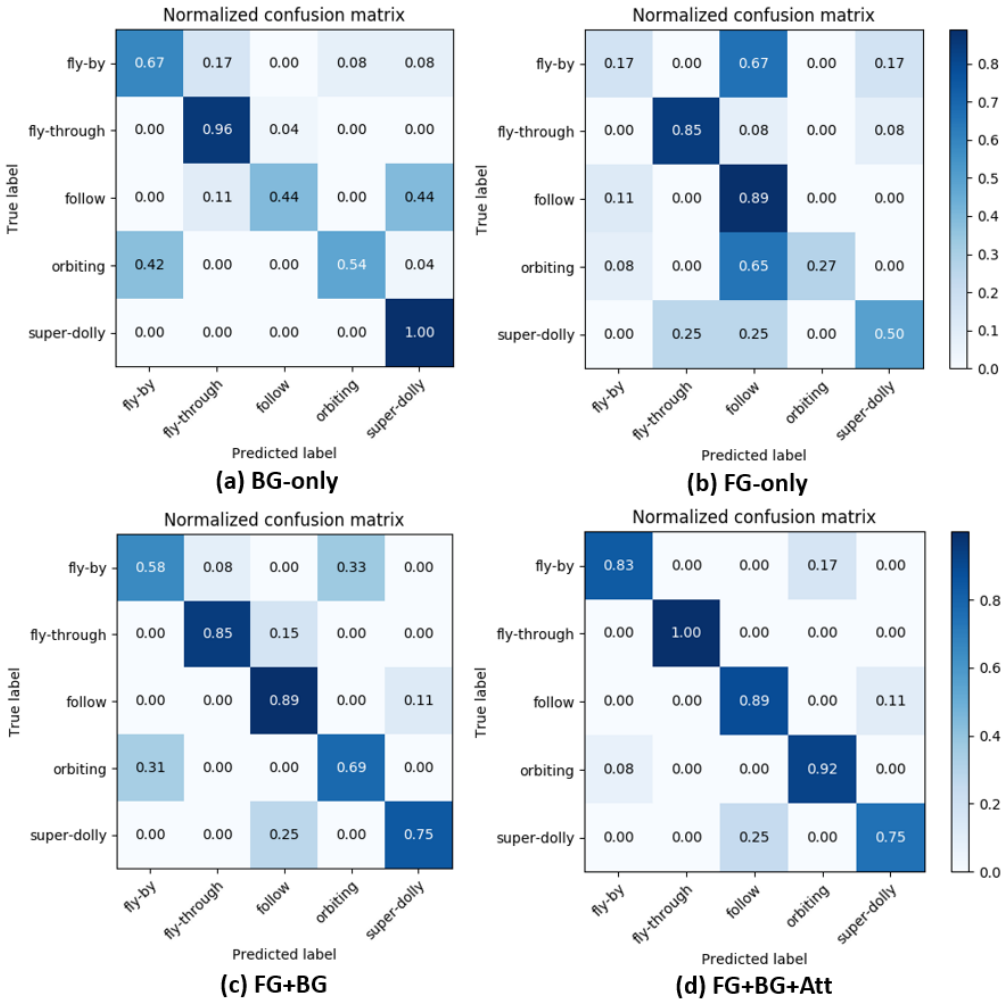
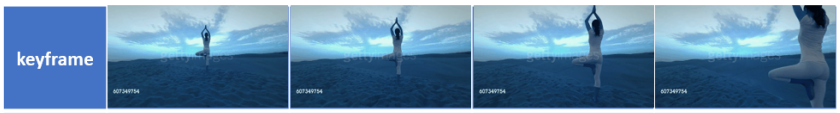


Figure 4.16: The confusion matrix of four baselines.

To further investigate where the attention layer focus on, we visualize the attention weights of the background and foreground. Fig. 4.17 shows the distribution of the attention weights of a video with the style "fly-through". We can see that when the subject's on-screen size becomes large, the network pays more attention on the foreground. Specifically, the network assigns the highest weight to the clip during time interval 0:14-0:16 because this 2-second clip is much more distinctive than other clips in terms of style recognition.



keyframe				
Time	0:00-0:02	0:04-0:08	0:10-0:12	0:14-0:16
BG att.	**0.08**	0.06	0.01	0.00
FG att.	0.01	0.01	0.02	**0.2**

Figure 4.17: The foreground/background attention weights (FG/BG att.) of a video with the style "fly-through".

Camera Motion Prediction

In this subsection, we compare the imitation performance of the models trained by the proposed loss function (Eq. 4.15) and the loss in [67]. Tab. 4.8 shows the prediction accuracy of the angular speed (ω), linear direction vector (v) and the scale (s) in terms of different filming styles. We can see that the our proposed method can keep consistent improvement over the model trained from the method [67] in different styles.

Table 4.8: Comparison of action prediction with different loss

	proposed method			Duan et al. [67]		
style	ω	v	s	ω	v	s
fly-by	0.043±0.012	0.435±0.097	0.046±0.010	0.079±0.013	0.863±0.118	0.048±0.014
fly-through	0.012±0.002	0.010±0.001	0.028±0.004	0.013±0.002	0.009±0.003	0.030±0.005
orbiting	0.049±0.005	0.031±0.002	0.028±0.003	0.062±0.007	0.044±0.002	0.026±0.008
follow	0.011±0.001	0.003±0.000	0.025±0.003	0.011±0.001	0.010±0.001	0.024±0.006
super-dolly	0.019±0.003	0.027±0.002	0.023±0.003	0.020±0.005	0.029±0.004	0.034±0.002

Application to Drone Cinematography System

In this subsection, we deploy our one-shot imitation filming method to a real drone platform for the autonomous cinematography task. Specifically, we build our drone cinematography system on the DJI Matrix 100 with two onboard embedded systems (Nvidia Jetson TX2 and DJI Manifold). We feed a demo video and captured a new video within the same duration.

First we evaluate imitation filming of the demo video with a single basic styles. We

invited 5 pilot beginners and 2 pilot professionals and asked each of them to capture 2 video clips for each style (each style only has one demo video). Meanwhile, we utilize our drone system to capture 2 video clips for each style. As Fig. 4.18 shows, our drone system can imitate the demo video as the professional cameramen. More specifically, Fly-by and orbiting are more difficult to imitate because they require users to manipulate the rotation and translation simultaneously. Follow is more difficult than fly-through and super-dolly to imitate because it require users to keep pace with the subject.

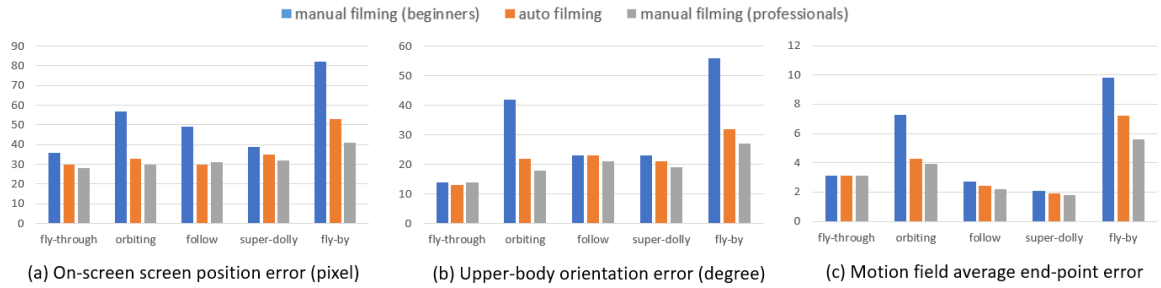


Figure 4.18: The difference between the captured video and the demo video in terms of different metrics

Second, we evaluate the performance on the demo video with the mixed style. We select 4 demo videos including orbit/super-dolly, super-dolly/orbit, follow/super-dolly, and follow/orbiting. Similarly, we invited 5 pilot beginners and 2 pilot professionals and asked each of them to capture 2 video clips for each video. Fig. 4.19 shows the difference between the captured video and the demo video. We can see that our proposed system can consistently imitate the demo video as the professional cameraman. In addition, we also find that follow/super-dolly is relatively easy to imitate because it is composed of two linear motions. Follow/orbiting is relatively difficult to imitate because it requires the camera to orbit around the subject while keeping tracking the subjects motion. Fig. 4.20 shows the snapshots of the demo video and captured videos with the same filming style "follow/orbiting". As the demo video (first row) of Fig. 4.20 shows, the camera starts by following a running person and then moves to the right side along a smooth curve. The

video (second row) of Fig. 4.20 shows that the beginner cannot skillfully manipulate the camera to adjust the image composition, especially when the camera is being transferred to the right side. The on-screen subject is placed to the edge of the image on the 250th frame. In the third row of Fig. 4.20, the professional cameraman can imitate the demo video with the consistent image composition and camera angle. Similar to the professional cameraman, our proposed system also can capture the similar video while there exists subtle different image composition. To dive deep into the imitation performance, we quantifies the temporal change of visual appearance of Fig. 4.20 in terms of the subject's on-screen position error and sparse optical flow error. When the back view begins to transfer to the side view at the 150th frame, the image composition error of the video manually captured by the beginner keeps increasing until the 210th frame (see Fig. 4.21(top)). The non-smooth optical flow change in Fig. 4.21(bottom) explains that the stiff operation of the beginner causes the sudden change of the image composition of the subject. We can see that our proposed system can generate consistent image composition and optical flow as the professional cameraman. Even the system suffers from the occasional biased tracking due to misprediction at the 100th frame, the system can adjust back to the correct camera pose via feedback control instantly. The attached videos will provide a more convincing comparison.

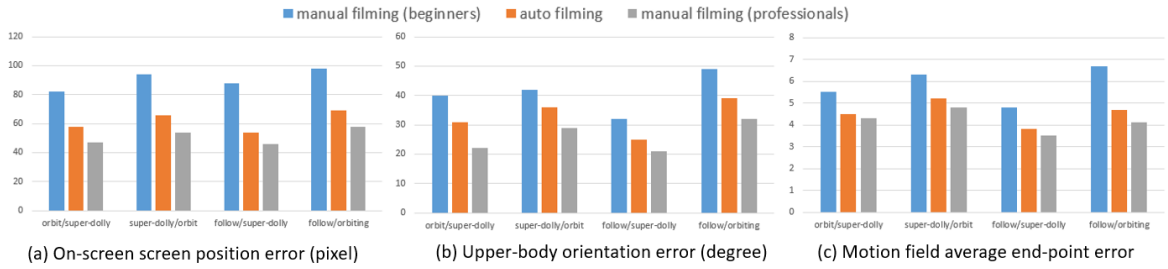


Figure 4.19: The difference between the captured video and the demo video in terms of different metrics

Figure 4.20: The snapshots of four videos: (first row) demo video, (second row) the video manually captured by a beginner, (third row) the video manully captured by a professional, and (fourth row) the video autonomously captured by our system.

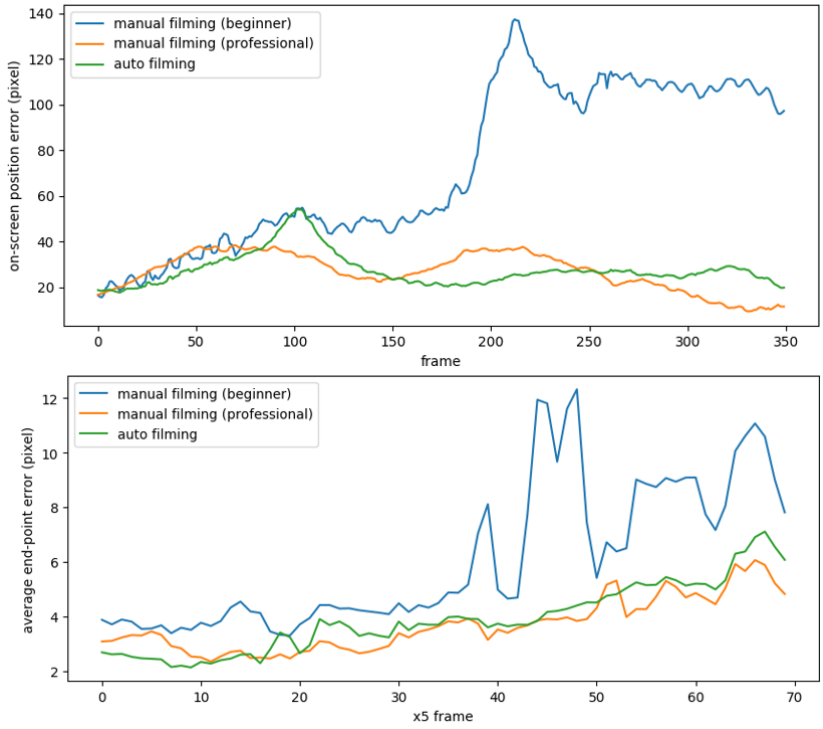


Figure 4.21: The difference between the capture videos and the demo video: (top) subject's on-screen position error, and (bottom) the average end-point error of the optical flow on the background.

Chapter 5

Conclusion and Future Work

In this thesis, we present two approaches of capturing a cinematic video of human motion using a drone: 1) guiding the camera based on the viewpoint quality and 2) guiding the camera based on imitation learning.

First, we aim to maximize the video quality by maximizing the viewpoint quality over all the frames. We found that the existing drone cinematography systems fail to capture the cinematic video due to the oversimplified viewpoint quality metric. Specifically, these methods apply the predefined configuration of image composition to guide only guides the camera movement such as pan and follow, which greatly limits the creativity of aerial filming. Meanwhile, these drone cinematography systems depend on the external motion capture systems to perceive the human action, which is limited to the indoor environment. Based on the analysis, we design a quality metric "visibility of the subject" as the guidance of camera planning. To the end, we propose an Autonomous CinemaTography system "ACT" on the drone platform which can adjust camera angle by analyzing human motion from the RGB camera. Experimental results in both simulation and real-world scenarios demonstrate that our cinematography system "ACT" can capture more expressive video footage of human action than existing drone systems. Meanwhile, we extend

the application of human motion analysis to manual operation of viewpoint control. We found that although moving control sticks can directly control a drones motion parameters (i.e. roll, yaw, pitch, and throttle), controlling these parameters do not offer a precise control of the movement of objects in the camera screen. To address this problem, we introduce the concept of "through-the-lens" from graphics community to simplify the manual operation of the camera. This is done by three key techniques: 1) real-time subject tracking based on 3D human localization and 2) "through-the-lens" interface that can allow the user to freely customize best viewpoint and 3) planning policy transfer the camera configuration in virtual environment to the desired camera motion in real-world.

Furthermore, we extend the goal to autonomously imitating the professional video, which creates a film look via the specific pattern of viewpoint transition. Because it is difficult to handcraft the viewpoint metric to reproduce the similar visual effect over different dynamic scenarios, we propose a data-driven learning-based approach, which can imitate a professional cameramans intention for capturing a film-look aerial footage of a single subject in real-time. Our journey to imitation filming includes three stages: First, we describe the viewpoint as the combination of the image composition and camera position, and present a "viewpoint-to-viewpoint" camera planning framework, which model the decision-making process of the cameraman with two steps: 1) we train a network to predict the future viewpoint, and 2) our system then generates control commands to achieve the desired shot framing. Second, analyzing the impact of different factors on the future camera motion, we design a camera planner which incorporates the foreground and background feature in video contents and previous camera motions to predict the future camera motions that enable the capture of professional videos. Finally, we propose a framework, which can imitate a filming style by seeing only a single demonstration video of the same style, i.e., one-shot imitation filming. This is done by two key enabling techniques: 1) feature extraction of the filming style from the demo video, and 2) filming style

transfer from the demo video to the new situation. Compared with the previous framework, "one-shot imitation filming does not need to train multiple style-specific models to imitate different filming styles. This advantage can enable the camera agent to quickly learn to generalize the unknown style of a given video to the new situation.

Although in this dissertation we only focus on filming the video that contains only one person, the proposed techniques can be transferred to multi-person activities. For example, we can utilize the same viewpoint quality metric to automate filming a video of boxing game. Of course, in more complex situations (e.g. soccer), the viewpoint quality is not just determined by the visibility of the players but also the attention of different players. The key player would be placed in the more visible on-screen position. Similarity, the imitation-based approach also works in the multi-person activities if we change the inputs of the network. For example, we can utilize the spatial and temporal distribution of the players on the ground as input.

In addition, with the development of the technologies of multi-drone collaboration, the multi-drone collaborative filming will be a new active topic in the future. The problem will be not just the combination of a set of independent viewpoint selection sub-problems from multiple drones. The related factors involve the the coverage of field view, energy consumption, communication quality, obstacle avoidance and so. In summary, a key question is how to integrate sensing, networking, and coordination on the resource-constrained drone platforms.

Last but not least, we can extend the input of auto filming to the audio. Camera movement appears to be a technique to make a music footage more interesting and keep the audience engaged. When the movement is done well, it can increase the audience's opinion of your production value. Therefore, we can apply imitation learning techniques to learn the connection between the pace of a given song and the change of video content.

Appendix A

Appendix Title

Bibliography

[1] 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).

[2] 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*, .

[3] 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)* **36** (2017), no. 4 132.

[4] F. Chen, D. Delannay, and C. De Vleeschouwer, *An autonomous framework to produce and distribute personalized team-sport video summaries: A basketball case study*, *IEEE Transactions on multimedia* **13** (2011), no. 6 1381–1394.

[5] S. Daigo and S. Ozawa, *Automatic pan control system for broadcasting ball games based on audience's face direction*, in *Proceedings of the 12th annual ACM international conference on Multimedia*, pp. 444–447, 2004.

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

[7] M.-J. Kim, T.-H. Song, S.-H. Jin, S.-M. Jung, G.-H. Go, K.-H. Kwon, and J.-W. Jeon, *Automatically available photographer robot for controlling composition and taking pictures*, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6010–6015, IEEE, 2010.

[8] 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, p. 3, 2017.

[9] N. Halper, R. Helbing, and T. Strothotte, *A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence*, in *Computer Graphics Forum*, vol. 20, pp. 174–183, Wiley Online Library, 2001.

[10] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross, *Visibility transition planning for dynamic camera control*, in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 55–65, 2009.

[11] Q. Galvane, R. Ronfard, C. Lino, and M. Christie, *Continuity editing for 3d animation*, in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[12] R. Ranon and T. Urli, *Improving the efficiency of viewpoint composition*, *IEEE Transactions on Visualization and Computer Graphics* **20** (2014), no. 5 795–807.

[13] Q. Galvane, R. Ronfard, M. Christie, and N. Szilas, *Narrative-driven camera control for cinematic replay of computer games*, in *Proceedings of the Seventh International Conference on motion in games*, pp. 109–117, 2014.

[14] A. Sanna, F. Lamberti, G. Paravati, and F. Manuri, *A kinect-based natural interface for quadrotor control*, *Entertainment Computing* **4** (2013), no. 3 179–186.

[15] 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.

[16] 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)* **36** (2017), no. 4 44.

[17] H. Lim and S. N. Sinha, *Monocular localization of a moving person onboard a quadrotor mav*, in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 2182–2189, IEEE, 2015.

[18] 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)*, pp. 7039–7046, IEEE, 2018.

[19] W. H. Bares, S. Thainimit, S. McDermott, and C. Boudreaux, *A model for constraint-based camera planning*, in *Proceedings of AAAI spring symposium on smart graphics*, pp. 84–91, 2000.

[20] B. Tomlinson, B. Blumberg, and D. Nain, *Expressive autonomous cinematography for interactive virtual environments*, in *Proceedings of the fourth international conference on Autonomous agents*, pp. 317–324, ACM, 2000.

[21] E. Dichter, *What's in an image*, *Journal of consumer marketing* **2** (1985), no. 1 75–81.

[22] M. Gleicher and A. Witkin, *Through-the-lens camera control*, in *ACM SIGGRAPH Computer Graphics*, vol. 26, pp. 331–340, ACM, 1992.

[23] M. Christie, P. Olivier, and J.-M. Normand, *Camera control in computer graphics*, in *Computer Graphics Forum*, vol. 27, pp. 2197–2218, Wiley Online Library, 2008.

[24] J. Assa, D. Cohen-Or, I.-C. Yeh, T.-Y. Lee, *et. al.*, *Motion overview of human actions*, in *ACM Transactions on Graphics (TOG)*, vol. 27, p. 115, ACM, 2008.

[25] I. Yeh, C.-H. Lin, H.-J. Chien, T.-Y. Lee, *et. al.*, *Efficient camera path planning algorithm for human motion overview*, *Computer Animation and Virtual Worlds* **22** (2011), no. 2-3 239–250.

[26] D. Rudoy and L. Zelnik-Manor, *Viewpoint selection for human actions*, *International journal of computer vision* **97** (2012), no. 3 243–254.

[27] J.-Y. Kwon and I.-K. Lee, *Determination of camera parameters for character motions using motion area*, *The Visual Computer* **24** (2008), no. 7 475–483.

[28] M. Gleicher and A. P. Witkin, *Through-the-lens camera control*, in *SIGGRAPH*, 1992.

[29] L.-w. He, M. F. Cohen, and D. H. Salesin, *The virtual cinematographer: a paradigm for automatic real-time camera control and directing*, in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 217–224, ACM, 1996.

[30] J. Assa, L. Wolf, and D. Cohen-Or, *The virtual director: A correlation-based online viewing of human motion*, in *Computer Graphics Forum*, vol. 29, pp. 595–604, Wiley Online Library, 2010.

[31] A. Litteneker and D. Terzopoulos, *Virtual cinematography using optimization and temporal smoothing*, in *Proceedings of the Tenth International Conference on Motion in Games*, p. 17, ACM, 2017.

[32] C. Lino and M. Christie, *Intuitive and efficient camera control with the toric space*, *ACM Transactions on Graphics (TOG)* **34** (2015), no. 4 82.

[33] T. Wischgoll, *Display systems for visualization and simulation in virtual environments*, *Electronic Imaging* **2017** (2017), no. 1 78–88.

[34] T. Kawagoe, Y. Yamada, H. Umemiya, and M. Ogawa, *Video game apparatus and method with enhanced virtual camera control*, 2003. US Patent 6,612,930.

[35] H. Hirschmuller, *Accurate and efficient stereo processing by semi-global matching and mutual information*, in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 807–814, IEEE, 2005.

[36] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, *Realtime multi-person 2d pose estimation using part affinity fields*, *arXiv preprint arXiv:1611.08050* (2016).

[37] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich, *Viewpoint selection using viewpoint entropy.*, in *VMV*, vol. 1, pp. 273–280, 2001.

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

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

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

[41] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, *Autonomous aerial navigation using monocular visual-inertial fusion*, *Journal of Field Robotics* (2017).

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

[43] K. Shoemake, *Animating rotation with quaternion curves*, in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, (New York, NY, USA), ACM, 1985.

[44] 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*, pp. 4688–4696, 2016.

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

[46] 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* **29** (2012).

[47] 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*, pp. 369–372, ACM, 2013.

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

[49] 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* **33** (2017), no. 5 1255–1262.

[50] G. B. Rath and A. Makur, *Iterative least squares and compression based estimations for a four-parameter linear global motion model and global motion compensation*, *IEEE Transactions on Circuits and Systems for Video Technology* **9** (1999), no. 7 1075–1099.

[51] S. Bhattacharya, R. Mehran, R. Sukthankar, and M. Shah, *Classification of cinematographic shots using lie algebra and its application to complex event recognition*, *IEEE Transactions on Multimedia* **16** (2014), no. 3 686–696.

[52] K. Li, S. Li, S. Oh, and Y. Fu, *Videography-based unconstrained video analysis*, *IEEE Transactions on Image Processing* **26** (2017), no. 5 2261–2273.

[53] K. Huang, Q. Wang, and Z. Wu, *Color image enhancement and evaluation algorithm based on human visual system*, in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 3, pp. iii–721, IEEE, 2004.

[54] K.-q. Huang, Z.-y. Wu, G. S. Fung, and F. H. Chan, *Color image denoising with wavelet thresholding based on human visual system model*, *Signal Processing: Image Communication* **20** (2005), no. 2 115–127.

[55] K.-Q. Huang, Q. Wang, and Z.-Y. Wu, *Natural color image enhancement and evaluation algorithm based on human visual system*, *Computer Vision and Image Understanding* **103** (2006), no. 1 52–63.

[56] R. Datta, D. Joshi, J. Li, and J. Z. Wang, *Studying aesthetics in photographic images using a computational approach*, in *European Conference on Computer Vision*, pp. 288–301, Springer, 2006.

[57] W. Luo, X. Wang, and X. Tang, *Content-based photo quality assessment*, in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2206–2213, IEEE, 2011.

[58] S. Dhar, V. Ordonez, and T. L. Berg, *High level describable attributes for predicting aesthetics and interestingness*, in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1657–1664, IEEE, 2011.

[59] A. K. Moorthy, P. Obrador, and N. Oliver, *Towards computational models of the visual aesthetic appeal of consumer videos*, in *European Conference on Computer Vision*, pp. 1–14, Springer, 2010.

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

[61] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[62] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, *Rapid: Rating pictorial aesthetics using deep learning*, in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 457–466, ACM, 2014.

[63] Y. Kao, C. Wang, and K. Huang, *Visual aesthetic quality assessment with a regression model*, in *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 1583–1587, IEEE, 2015.

[64] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Z. Wang, *Deep multi-patch aggregation network for image style, aesthetics, and quality estimation*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 990–998, 2015.

[65] L. Mai, H. Jin, and F. Liu, *Composition-preserving deep photo aesthetics assessment*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 497–506, 2016.

[66] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, and T.-S. Chua, *Adversarial training towards robust multimedia recommender system*, *IEEE Transactions on Knowledge and Data Engineering* (2019).

[67] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, *One-shot imitation learning*, in *Advances in neural information processing systems*, pp. 1087–1098, 2017.

[68] C. Finn, P. Abbeel, and S. Levine, *Model-agnostic meta-learning for fast adaptation of deep networks*, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135, JMLR. org, 2017.

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

[70] 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).

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

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

[73] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997), no. 8 1735–1780.

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

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

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

[77] T. Qin, P. Li, and S. Shen, *Vins-mono: A robust and versatile monocular visual-inertial state estimator*, *arXiv preprint arXiv:1708.03852* (2017).

[78] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, *Convolutional lstm network: A machine learning approach for precipitation nowcasting*, in *Advances in neural information processing systems*, pp. 802–810, 2015.

[79] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, *arXiv preprint arXiv:1804.02767* (2018).

[80] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, *Flownet: Learning optical flow with convolutional networks*, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2758–2766, 2015.

[81] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, *Flownet 2.0: Evolution of optical flow estimation with deep networks*, in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 2, p. 6, 2017.

[82] C. Liu *et. al.*, *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[83] V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, *arXiv preprint arXiv:1603.07285* (2016).

[84] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[85] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, *Temporal segment networks: Towards good practices for deep action recognition*, in *European conference on computer vision*, pp. 20–36, Springer, 2016.

[86] C. Smith, *The photographer's guide to drones*. Rocky Nook, Inc., 2016.

[87] C. Tomasi and T. K. Detection, *Tracking of point features*, tech. rep., Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, 1991.

[88] B. Lewandowski, D. Seichter, T. Wengefeld, L. Pfennig, H. Drumm, and H.-M. Gross, *Deep orientation: Fast and robust upper body orientation estimation for mobile robotic applications*, .

[89] N. Srivastava, E. Mansimov, and R. Salakhudinov, *Unsupervised learning of video representations using lstms*, in *International conference on machine learning*, pp. 843–852, 2015.

[90] D. J. Berndt and J. Clifford, *Using dynamic time warping to find patterns in time series.*, in *KDD workshop*, vol. 10, pp. 359–370, Seattle, WA, 1994.

[91] OpenSFM. `https://www.opensfm.org`, 2017.