Autonomous Cinematography using Unmanned Aerial Vehicles

Yanfu Zhang*[1], Wenshan Wang*[2], Rogerio Bonatti*[2], Daniel Maturana [2], and Sebastian Scherer[2]

I. Introduction

In this paper we present a system to enable autonomous cinematography using UAVs. Aerial cinematography is popular with both professional and amateur film makers due to its capability of composing viewpoints that are not feasible using traditional hand-held cameras. With recent advances in state estimation and control technology for multi-rotors, consumer-level drones have become easier to operate; even amateur pilots can easily use them for static landscape filming. However, aerial filming for moving actors in action scenes is still a difficult task for drone pilots and requires considerable expertise. There is increasing demand for an autonomous cinematographer; one that can track the actor and capture exciting moments without demanding attention and effort from a human operator.

Previous approaches in autonomous aerial filming only partially address the full problem for real-life scenarios. They use simplifications such as using high-precision RTK GNSS [1] or motion-capture systems [2], generating offline trajectories [3], or only dealing with limited obstacle representations [2].

In order to address the full filming problem in a generalizable manner, we borrow commonly used shot parameters from professional cinematography [4]. Based on the artistic principles, we define the problem as: at each time step, we control the UAV and the camera pose relative to the actor to satisfy the pre-defined shot parameters, while avoiding obstacles and occlusions. Therefore, the filming system's perception modules should be able to estimate the actor's pose, forecast the actor's future motion, and model local obstacles. Given the outputs from the perception module, the planning module plans motions that optimize for shot quality, obstacle avoidance, occlusion-free views, and smoothness, in real time. Figure 1 shows a schematic of the problem definition and example of a scene filmed with our system.

The major challenges in enabling such an autonomous aerial filming system then become: 1) Estimating pose of the actor under noisy sensor measurements, 2) Forecasting the actor's motion in the environment, and 3) Optimize in real time the poses from the UAV and camera with respect to a set of non-trivial artistic objectives. Section II briefly describes a system that addresses these challenges, and sections III, IV and V detail our approaches in each area.

*Equal contribution

[1]Yamaha Motor Co., Ltd., Fukuroi, Shizuoka 4370061, Japan zhangya@yamaha-motor.co.jp

[2]Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213, USA {wenshanw, rbonatti, dmaturan, basti}@andrew.cmu.edu

II. System overview

A. Hardware

Our base platform is a DJI M210 quadcopter, equipped with a NVIDIA TX2 computer and a Zenmuse X4S camera gimbal. The TX2 build-in GPU enables our real-time neural network inference for actor detection and heading estimation. We will also add a custom rotating VLP-16 LiDAR to the full system in the future for real-time mapping.

B. Software

Our software system is broadly divided into two subsystems: vision, which controls the orientation of the camera gimbal, and planning, which controls the UAV pose, shown in Figure 2.

A key aspect of our design is that we decouple the control of the UAV pose and the camera orientation. The camera orientation is controlled by the vision subsystem using image-space feedback, while the UAV pose is controlled by the planning subsystem using the estimated actor pose in the world frame. The main reason for this design is that the estimated actor pose in the world frame is too noisy to use as a signal for controlling the camera orientation, given that small position errors lead to large offsets in the image space, causing degradation of the video aesthetics.

In the vision subsystem, we first detect and track the actor using a monocular camera. We use the bounding box of the tracked actor as a image-space signal to control the camera gimbal, keeping it in the desired screen position. We further adopt a ray-casting approach to estimate the actor's current pose and a learning-based method to forecast the actor's future poses in the world frame.

In the planning subsystem, we first calculate the artistically optimal UAV poses analytically without considering smoothness and obstacles. Then we plan the UAV trajectory using a novel trajectory optimization algorithm that considers shot quality, obstacle avoidance, occlusions, and smoothness [5].

C. Simulator

Based on AirSim [6], we built a ROS wrapper to test our software in a photo-realistic simulator. We design the interface in a way such that the autonomy software can seamlessly be used with either the simulator or the real robot. The underlying photo-realistic rendering allows us to test the perception algorithms together with the motion planner in simulation.

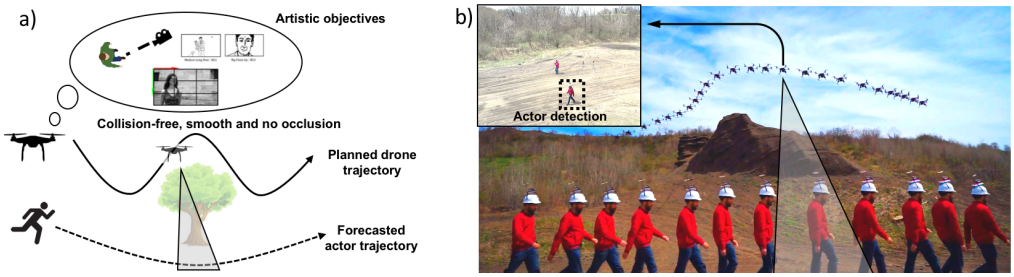

Fig. 1: We propose a system for autonomous aerial cinematography: a) The drone reasons about artistic guidelines provided by a user, forecasts the motion of the actor and calculates a smooth, collision-free trajectory, avoiding occlusion. b) Real-life results validate the system working with arbitrary obstacle shapes, producing visually appealing images. The camera detects and tracks the actor on the input images, keeping him in the desired screen position.

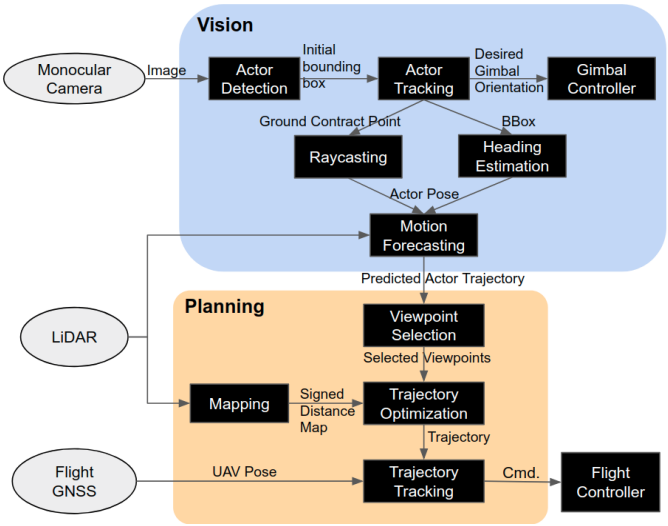

Fig. 2: Software system architecture. The vision subsystem controls the camera orientation using image-space feedback and forecasts the actor's future poses in the world frame. The planning subsystem uses the forecasted actor motion, current UAV pose, and truncated signed distance map to generate flight trajectories.

III. Vision-based actor detection, tracking and heading estimation

A. Detection and Tracking

In the vision subsystem, the detection module outputs a bounding box from a single image frame to initialize the tracking process, and re-initialize whenever it fails. We adopt this approach, as opposed to detecting the actor in every image frame (a form of tracking-by-detection), because the higher rate of tracking gives a smoother signal for gimbal control.

Object detection is among the most prevalent tasks in computer vision. We select algorithms from the state of the art approaches that strike a good speed-accuracy tradeoff for our task. We tested three algorithms: single shot detector (SSD) [7], Faster R-CNN [8], and YOLO2 [9] using our dataset, and found that Faster R-CNN performs the best in terms of precision-recall metrics while operating at a reasonable speed for our task. We use MobileNet [10] for feature extraction due to its low memory usage and fast inference speed. The per-frame inference is around 300 ms, running on the TX2. Due to the small size of our dataset, we train all models using a mixture between COCO [11] dataset and our dataset with a 1:10 ratio. We limited the detection categories only to *person*, *car*, *bicycle*, and *motorcycle*, which commonly appear as actors in aerial filming. Finally for actor tracking, we use KCF [12] due to its real-time performance.

B. Heading Estimation and Ray-casting

To estimate the actor pose in the world frame, we add the raycasting module and the heading estimation module. We estimate the pose of the actor as a vector $[x, y, \theta]$ on the ground plane. We infer the pose of the ground plane (assumed to be flat and horizontal) using the global state estimate from the flight controller (which uses GNSS/IMU, barometer, magnetometer and a multi-camera rig).

To estimate the $[x, y]$ component of the pose, the raycasting module projects the bottom center of the detected bounding box onto the ground plane using the known extrinsic and intrinsic camera parameters. We experimentally verify our pose estimation method (Figure 4), and Figure 5 summarizes the results. We observe low errors in the actor's $[x, y]$ estimation for both a linear and circular motion patterns. Errors in sensor synchronization and in the bounding box detection cause most of the offset observed in the test.

To estimate the θ component, we first estimate the actor's heading in image space. This is achieved by using a light-weight custom convolutional neural network (CNN) to

Fig. 3: Photo-realistic simulator used to test the system. Third and first-person renderings shown on the left, and occupancy map with drone trajectory shown on the right.

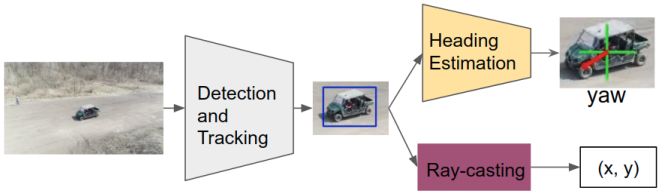

Fig. 4: A block diagram of the vision pipeline.

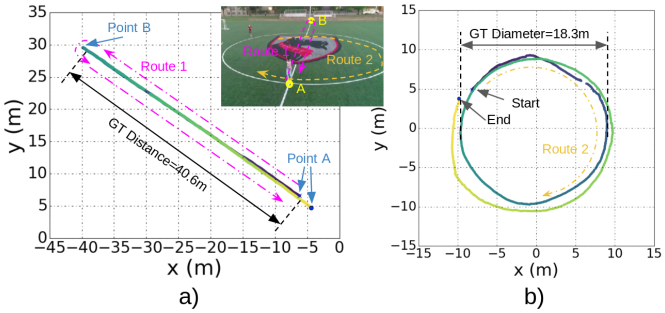

Fig. 5: Estimated actor trajectory by ray-casting module: a) The actor walks on a straight line from point A to B, and back to A again. While ground-truth trajectory length is 40.6m in total, the estimated motion length is of 42.3m; b) The actor walks twice on a circle. While the ground-truth diameter is 18.3m, the estimated diameter from ray casting is 18.7m.

regress the heading to an image-space angle, which is then projected to the world frame using the camera parameters, as before.

IV. Actor Motion Forecasting

The capability to forecast the filming actor's future motion is critical for the filming task, and more generally, for autonomous robots operating in dynamic environments. With accurate motion forecasting, our UAV can plan for more intelligent behaviors to achieve specified objectives, instead of acting in a purely reactive way. However, motion forecasting is difficult, because the interaction between the actor and the local environment is complex and hard to model in a generalizable way.

Inverse reinforcement learning (IRL) is a method that recovers the underlying reward function that motivates an agent to perform certain tasks [13]. IRL is suitable for our task, because it can forecast the actor's future motion without heavily relying on hand-designed cost functions and heuristics. We propose a novel two-stage network architecture to reason about environmental and kinematic context when computing rewards [14], as explained in Figure 6. In the first stage, we adopt a four-layer fully convolutional network (FCN) [15] which takes environmental context features as input (a local top-down view of LiDAR shape statistics plus color information from cameras), and outputs 25 learned feature maps. As inputs to the second stage, we concatenate the feature maps from the first stage with two feature maps encoding positional information and three feature maps representing kinematic information from the past trajectory.

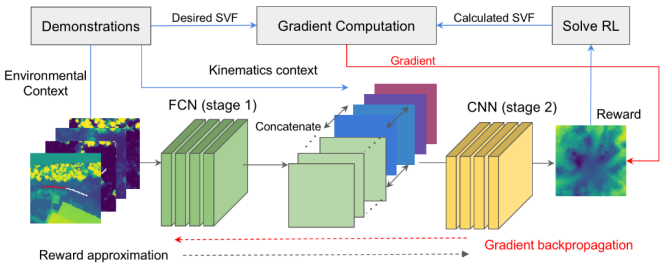

Fig. 6: Proposed two-stage network deep IRL framework and training procedure. Environment context represented by 2.5D LiDAR statics is the input to the first-stage network, and the resulting feature maps are concatenated with the kinematics context. The reward approximation is the output of the second-stage network, and the state visitation frequency difference between learned policy and demonstration is used to compute gradients for backpropagation.

The output of the second stage is a reward map which can be used to infer future motion. We train and test the network with a custom off-road driving dataset, which includes over 1000 trajectories with 20-40 m average length.

We experimentally verify that the forecasts calculated based on the the inferred reward map match the expected behaviors of a human driver, with trajectory probabilities concentrated on traversable paths. In addition, multi-modal behavior are present at intersections and when approaching open areas. Figure 7 summarizes representative results, and a supplementary video shows more examples: `https://goo.gl/xpSvvb`. We also compare our method with four baselines: extended Kalman filter (EKF), behavior cloning (BC), random policy, and IRL with out kinematics information, shown in Table I. Our method has the best forecast results on the test set using both negative log-likelihood (NLL) and Hausdorff distance (HD) as the comparing metric.

The forecasted probability distribution of the future motion of a vehicle can provide richer and more accurate prediction compared with filtering-based methods when planning the motion of the UAV in the filming task.

Method	EKF	BC	Random	IRL w.o. kinematics	Ours
NLL	N.A.	0.87	1.35	1.33	**0.69**
HD	9.12	10.54	25.62	25.46	**6.71**

TABLE I: Forecasting accuracy comparison on the test set using NLL and HD. For both NLL and HD, lower numbers represent better predictions. Our method obtains the best results in both evaluation metrics.

V. UAV Motion Planning

Using a prediction for the actor trajectory (ξ_a), we can now generate a drone trajectory (ξ_q) that results in artistically pleasing videos. Following literature in cinematography [16], [4], we can treat the autonomous cinematographer as an agent that executes trajectories which follow the director's artistic guidelines as closely as possible while moving

smoothly, avoiding obstacles and keeping a clear view of the actor (minimizing occlusion). We can then define total trajectory cost:

$$J\left(\xi_q\right) = J_{\text{shot}}\left(\xi_q, \xi_a\right) + J_{\text{smooth}}\left(\xi_q\right) + J_{\text{occ}}\left(\xi_q, \xi_a\right) + J_{\text{obs}}\left(\xi_q\right)$$

Our key insight is that this motion planning problem can be efficiently solved in real-time as a smooth trajectory optimization. Following the derivation of [17], we define smoothness ($J_{\text{smooth}}\left(\xi_q\right)$) and obstacle avoidance ($J_{\text{obs}}\left(\xi_q\right)$) costs. In addition, we define cost functions specifically for cinematography (detailed descriptions found in [5]):

$$J_{\text{shot}}\left(\xi_q, \xi_a\right) = \frac{1}{t_f}\frac{1}{2}\int_0^{t_f} ||\xi_q(t) - \xi_{shot}(t, \xi_a)||^2 dt$$

Shot quality: written in a quadratic form, it measures the average squared distance between ξ_q and an ideal trajectory ξ_{shot} that only considers positioning via cinematography parameters. ξ_{shot} can be computed analytically: for each point $\xi_a(t)$ in the actor motion prediction, the drone position lies on a sphere centered at the actor.

$$J_{\text{occ}}\left(\xi_q, \xi_a\right) = \int_{t=0}^{t_f}\int_{\tau=0}^{1} c(p(\tau))\frac{d}{d\tau}p(\tau)d\tau\,\frac{d}{dt}\xi_q(t)dt$$

Occlusion: even though the concept of occlusion is binary, *i.e*, we either have or don't have visibility of the actor, a major contribution of our work is to define a differentiable cost that expresses a viewpoint's occlusion intensity for arbitrary obstacle shapes. Mathematically, occlusion is the integral of the truncated signed distance cost c over a 2D manifold connecting both trajectories ξ_q and ξ_a in a path $p(\tau)$.

With the camera orientation controlled separately, the drone's trajectory is represented as $\xi_d(t) = [x_d(t) \quad y_d(t) \quad z_d(t)]^T$ with the drone heading always assumed to be pointing towards the actor. We build our trajectory planner on CHOMP [18], as its update rule is amenable to new cost functions, and pre-compute an approximation of the Hessian using analytic second-order derivatives from the smoothness and shot quality costs.

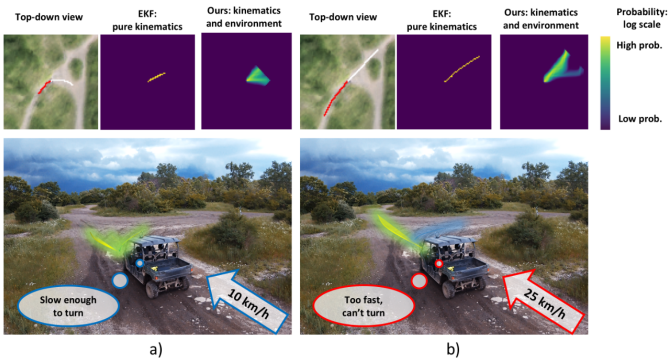

Fig. 7: Motion forecasting results on the same environment comparing different kinematic speeds. The slow-speed case shows a stronger multi-modal forecast (a), while the high-speed case predicts a more rectilinear path (b). The top-down view shows the trajectory of the past 5 s (red) and ground truth (white).

We evaluated our planner in a real-world scenario, performing several types of shots following different types of actors: humans, cars and bicycles at different speeds and motion types. In total, we collected over 1.25 hours of flight time while re-planning at 5 Hz with a 10 s horizon. In the test, the actor wears a GNSS/INS module to provide his pose, and a Kalman filter predicts the actor's future motion. The environment map was generated offline. Figure 8 summarizes the most representative experimental results, and a supplementary video depicts more examples: `https://youtu.be/QX73nBBwd28`.

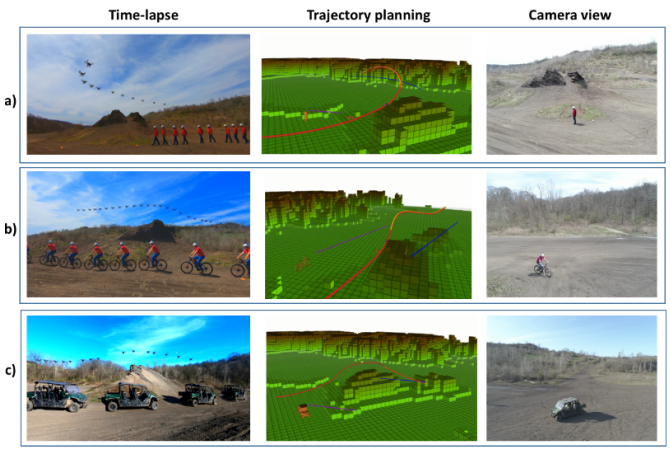

Fig. 8: Preliminary results: a) circling shot around person; b) side shot following biker; c) side shot following vehicle. The planned trajectory (red) avoids colliding with and being occluded by the mountain, while being smooth. The actor's motion forecast is in purple, and the desired artistic shot is in blue.

VI. CONCLUSION AND FUTURE WORK

In this work we describe a system which enables autonomous aerial filming using UAVs. We provide preliminary experimental results for each module and insights found during development. We are currently working on integrating all modules into a complete system and will report results in the future.

Additional future topics to be addressed in our work are: 1) Online mapping of the environment using an onboard LiDAR, 2) Refining the UAV path planning algorithm, reasoning about the uncertainty of the actor motion forecasts, and 3) Modelling artistic shot selection for different environmental contexts based on learned aesthetics metrics.

ACKNOWLEDGEMENTS

We thank Lentin Joseph, Aayush Ahuja, Delong Zhu, and Greg Armstrong for the assistance in field experiments and robot construction. Research presented in this paper was funded by Yamaha Motor Co., Ltd. .

REFERENCES

[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," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, 2017.

[3] M. Roberts and P. Hanrahan, "Generating dynamically feasible trajectories for quadrotor cameras," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 61, 2016.

[4] C. J. Bowen and R. Thompson, *Grammar of the Shot*. Taylor & Francis, 2013.

[5] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer, "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming," in *International Symposium on Experimental Robotics*, 2018.

[6] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," 2017.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[9] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint*, 2017.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[12] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[13] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[14] Y. Zhang, W. Wang, R. Bonatti, D. Maturana, and S. Scherer, "Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories," in *submission*, 2018.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[16] D. Arijon, "Grammar of the film language," 1976.

[17] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[18] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *ICRA*, pp. 489–494, IEEE, 2009.