Immersive Aerial Cinematography

Botao (Amber) Hu
81 Adam Way, Atherton, CA 94027
botaohu@cs.stanford.edu

Qian Lin
Department of Applied Physics, Stanford University
348 Via Pueblo, Stanford, CA 94305
linqian@stanford.edu

Abstract

We build an immersive aerial cinematography system combining programmed aerial cinematography with route planning and preview in 3D virtual reality (VR) scene. The user will have a 3D Oculus-Rift-based VR experience previewing the Google Earth model of the scene they plan to videotape. Switching between camera first-person-view and a global view, was well as multi-user interacting in the virtual world is supported. The user will be able to specify keyframes while viewing the scene from camera first-person view. These keyframes are subsequently used to construct a smooth trajectory, whose GPS coordinates are streamed to a quadrotor to execute the shot in autopiloting mode with GPS tracking.

1. Introduction

With the emergence of highly-stabilized quadrotors in consumer product space, aerial cinematography using quadrotors has became very popular for both professional and recreational photograph and filming use. However, in the main stream market, quadrotor filming is still currently done manually, usually requiring two people to operate (one controlling the quadrotor and the other the on-board camera). Such manual remote control is challenging, especially given the six degrees of freedom of the quadrotor and three degrees of freedom of the camera, as well as controlling all these nine degrees of freedom with proper timing. Drone companies like DJI and 3D Robotics provide commercial 2D waypoint mission planner. However, even with such autopiloting tool it is still difficult to visualize the resulting footage prior to execution.

Recent work by our collaborators at Stanford Computer Graphics Lab [1] provide a design tool that overcomes many of the previously mentioned drawbacks in quadrotor route planning. The designing tool provides camera first-person-view on Google Earth, and fine grain control over the trajectory defined by keyframes. The unique technical contribution of [1] is in providing a physical quadrotor camera model allowing the computation of the quadrotor and the gimbal driving the camera jointly. As a result a smooth trajectory can be computed from user-defined keyframes with specific time intervals. Basically, [1] solves the problem of route planning given a few keyframes. [1] also provides a platform of previewing and planning routes based on a combination of camera look-from and look-at control using mouse on a 2D Google map, and previewing of the scene from camera first-person-view in Google Earth. In our project, we replace this design platform with an more immersive and interative one.



Figure 1. Screen shot of the system in operation. The upper panel is a (monocular) camera first-person view. Also shown in red curve is a pre-planned camera route arcing the Hover tower. The two lower panels are the binocular views of a stand-by user in the Google Earth, also showing the quadrotor highlighted in yellow circle. These two views will be rendered in the left and right eye view on Oculus.

The unique contribution of our current project is to make the scene previewing and keyframe selection experience more intuitive and immersive. The originality of our project can be summaries in three-fold

- We track the physical motion of the user and adjust the Google Earth scene that the user sees to match the physical motion. Thus, physical motion of the user is translated into a visual feedback that matches the user expectation in a virtual world. This provides the user an authentic and interactive experience of actually be in the scene he or she plans to videotape.

- The keyframe selection process is integrated into the VR scene preview experience of the user. That is, instead of drawing the keyframe camera position on a 2D google map with a mouse on a browser window [1], the user can now "walk" to camera look-from positions and orient themselves to the look-at directions. If they are satisfied with the scene they are seeing in the virtual world, they can add a new keyframe to the route by pressing a bottom on the joystick.

- Our system allows multiple users to simultaneously be in the virtual world and share the VR experience. Our tracking system can record the positions of multiple objects in the same physical tracking space and translate they into the virtual world with the correct relative space. As a result, every user can see the virtual world from their own perspective, as well as other users at their proper position in the virtual world. For example, user A can be the quadrotor in the virtual world and sees the camera first-person-view, while user B can be a by-stander seeing a global view and the quadrotor/user A moving in the scene as it plans its trajectory. This is depicted in Fig. 1.

2. Previous Work

Our project is closely related to our collaborators' work [1]. Their interface provides a 3D preview showing the current camera view in Google Earth, and a 2D map showing a top down view of the camera position in Google Maps. The views allow users to preview shots in a 3D environment, as well as localize the camera with respect to the environment. In their software interface, a user designs camera shots by specifying sequence of look-from and look-at coordinates at specific execution time. The look-from and look-at latitude and longitude can be defined with mouse clip on a 2D Google map showing the top down view of the scene, and the altitude value needs to be entered manually. A real time preview of the current camera view in Google Earth is also provided on a separate panel. Once the keyframes and timing are specified by the user, a separate look-at and look-from curve is generated through the 3D space. A smooth camera trajectory is calculated from a model of the quadrotor coupled with camera gimbal that [1] introduces. Such a trajectory is subsequently executed by a quadrotor equipped with position and rotation feedback provided by on-flight GPS, barometer and gyro.

Our project replace the scene previewing and route planning part in [1] with an immersive experience. Once the keyframes are selected, we channel the data into [1]'s platform to perform trajectory computation, virtual footage generation and quadrotor controlling.



Figure 2. The physical setup of the mocap system and the work station. Four of the eight mocap cameras we used are captured in the photo and highlighted with red circles.

3. Approach

3.1. Motion capture tracking system

The motion-capture (mocap) system, as shown in Fig. 2, is used for tracking the physical motion of the user. The system we use is a Optitrack Flex 13 with a frame rate up to 120 fps and latency of 8.33 ms. The mocap system measures in real time the position and orientation of one or more infrared markers in the measurement space covered by the cameras. These markers are attached to the user so that they follow the motion of the user. In our application, we use two infrared trackers. On is attached to the Oculus Rift headset wore by the user. The other is attached to a handheld monitor screen to provide first person view of the camera.

Using a software package provided by Optitrack with their Mocap system, the measured position and orientation data of the markers are broadcast. Our web-based program subsequently use these data to calculate the corresponding camera position in the virtual world for properly displaying in Google Earth.

3.2. Physical tracking to Google Earth camera position

The infrared marker attached to either the camera monitor screen or Oculus headset tracks the motion of the rigid body to which it is attached, In this case the Google Earth camera view need to follow either the camera monitor or the motion of the user's head to provide authentic 3D experience. One important intermediate step is the convert the translation and rotation that the Mocap system measures in the physical space into proper Google Earth camera look-from coordinate and viewing direction [Fig. 3].

Consider rotation first. This means rotating in the physical world maps to change of camera angle in Google



Figure 4. Google Earth camera view controlled by marker position. From left to right: original view, change heading, change tilt, change roll, walk away.

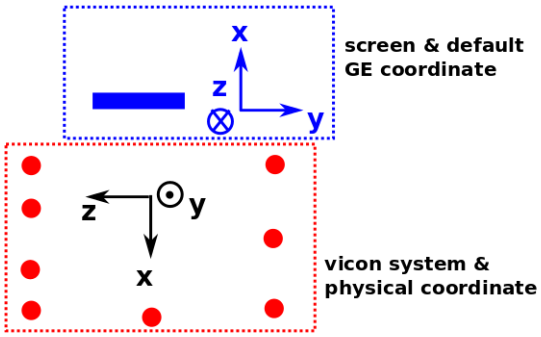


Figure 3. Sketch of the physical space of the Mocap tracking volume. Showing the relative orientation of the physical coordinate (south-up-west) and the Google Earth coordinate (north-east-down). Red dots represent the Mocap camera positions.

Earth. The Mocap system measures the rotate of the infrared marker relative to the physical frame, with an unknown initial orientation. In the initialization (calibration) step, we point the infrared marker towards the forward direction of the physical space (horizontally towards the computer monitor screen). In this direction the mocap system reads out a rotation $R_{P,0}$ representing the rotation between the (intrinsic) market coordinate with respect to the physical coordinate. By definition maps to the north pointing direction of the camera in Google Earth, representing by rotation matrix $R_{G,0} = I$ (the identity matrix). The physical coordinate and the default Google Earth coordinate is related by a rotation matrix

$$C = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad (1)$$

Thus for any subsequent mocap measurement R_P, the camera orientation in the Google Earth is

$$G_P = (C^{-1} R_{P,0}^{-1}) R_P (C G_{P,0}) \quad (2)$$

From G_P we can compute the three TaitBryan ZYX angles, corresponding to heading, tilt and roll in the Google Earth camera parameters.

Now let's consider translation. This means a user walking in the physical space maps to change of camera longitude, latitude and altitude in Google Earth. At initialization, the Google Earth latitude and longitude is set to the coordinate of the Hover tower, and the altitude is set to 0 (relative to ground). The initialized scale is $100 : 1$, meaning that moving by 1 m in the physical space moves the camera by 100 m in the Google Earth. Let $T_{G,0}$ be the initialized Google Earth coordinate and $T_{P,0}$ be the corresponding physical coordinate (position of the marker in the mocap system at initialization), and $S = 100$ be the initialized scaling factor, then

$$T_G = T_{G,0} + S(T_P - T_{P,0}) \quad (3)$$

We allow the user to reset the scaling by touching the up and down bottom on a joystick connected to the handheld monitor, or the up and down bottom on a keyboard. When rescaling, $T_{G,0}$ and $T_{P,0}$ is reset to the current position, and subsequent update of the position will use these new values and the new scaling factor. Rescaling enables zooming out of a scene to have an overview, or zoom in to finer details.

The effect of changing heading, tilting, rolling and backing up is shown in Fig. 4, as a demonstration of the motion-controlled view change.

3.3. 3D Google Earth view creation in Oculus Rift

In our previous considerations, we model the rigid body movement of a monocular camera. To render this camera view into stereo view for Oculus Rift, we assume a default eye separation in the physical world of 64 mm (the fixed Oculus lens separation). This translates into a $64\text{ mm} \times S = 10$ m separation between the two stereo cameras along the initial west direction. The two camera views are render into the left and right screen of Oculus Rift, as shown in Fig. 5. This results in a default stereo focus at infinity. Since we are not looking at close-up objects anyway, the rough stereo image generation provides good enough approximation of 3D experience.

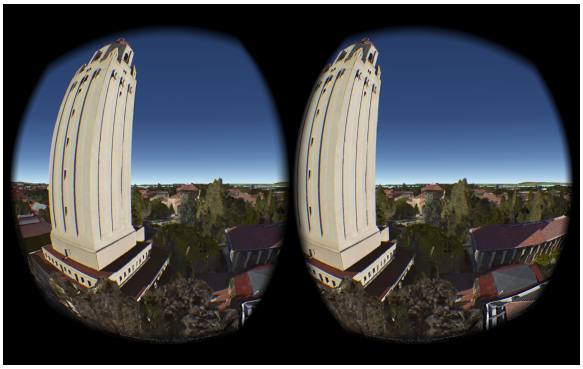


Figure 5. Stereo view created for Oculus display.

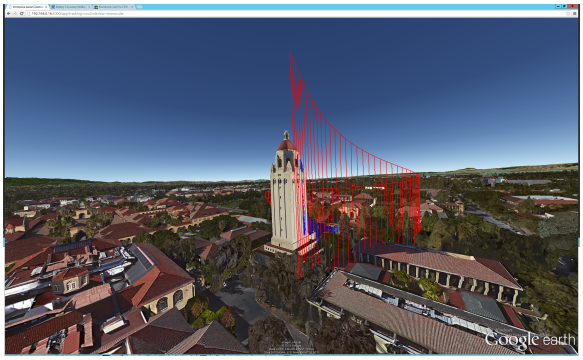


Figure 6. A view showing the overlay of planned look-from (red) and look-at (blue) curve. The camera position can be obtained from the look-from GPS coordinate, and the camera orientation is calculated from both curves.

3.4. Route planning

Using bottoms on a joystick, the current camera position and orientation can be added as a keyframe on a quadrotor/camera trajectory. We have a separate curve representing the camera look-from and look-at coordinate. The camera look-from is the current camera position, and the look-at is generated by a ray hit-test on the Google Earth. This is demonstrated in Fig. 6.

Whenever a keyframe is added, an basic algorithm produce a polynomial fitting to produce a planned trajectory that smoothly connects the added keyframe to the previous one. The user will be able to preview the footage based on Google Earth following the planned trajectory. We also enable keyframe deleting from the joystick.

4. Hardware and Software platforms

Our immersive aerial cinematography system includes four parts: (1) motion capture system; (2) immersive display (oculus rift and handheld monitor); (3) keyframes editing system; (4) aerial cinematography platform.

We use 8 Optitrack Flex 13 cameras to build up our motion capture system. The system will provide us a live stream of the pose estimation of all marked rigid bodies in the tracking volume with a frame rate up to 120 fps. The tracking volume we setup is 16ft x 16ft x 8ft. Inside the volume, the average tracking error is under 1mm and the average latency is under 8.33 ms. We attached markers to the Oculus Rift headset wore by the user and to the handheld monitor screen to provide first person view of the virtual camera.

Based on the streaming of pose estimation from motion capture system, we developped a web-based applciation to implement the immersive aerial cinematography system. The system registered the tracking volume to a space volume in the world. It will build a one-to-one correspondance between points in the tracking volume to a physical space. Based on the correspondance, we can compute the position and orientation of the virtual camera in Google Earth to present the scene of the tracked objects, e.g. the Oculus Rift or handheld monitor. Then, based on WebGL, we render the scene of the virtual camera, transfering a Google Earth camera view to a warped, shaded and chromatic aberration corrected stereo scene streamming to Oculus-rift head mount display or directly render a monocular view streamming to a handheld monitor.

In our system, we adapts a joystick, Xbox 360 Wireless controller, for user to wirelessly edit keyframes while the user is viewing the scene from the virtual camera. The user could add a keyframe by pressing a button on the wireless controller at the current position and orientation of the virtual camera which is reflected as her first person view. Also, she could remove the previous keyframe. Once the keyframe list has been edited, the system will recalculate the new smooth trajectory based on [1]. The user also could interactively dragging previous keyframe by holding a button on joystick, to adjust the camera view of the keyframe. Finally, the user could preview the planned trajectory by pressing a start button on joystick and export the planned trajectory to a way-point file, which will be streammed into the quadcopter when executing the plan.

Our aerial cinematography platform is based on IRIS+ quadrotor from 3DRobotics [2]. This quadrotor is an entry-level ready-to-fly quadrotor. It's equipped with a 2-axis gimbal for independent roll and pitch camera control. We attached a consumer GoPro Hero 4 Black camera to the gimbal. We used 900MHz telemetry radios for communication between the quadrotor and the ground station running our tool.

The IRIS+ is equiped with a standard GPS receiver, a barometer, and an inertial measurement unit (IMU) consisting of accelerometer and gyroscope. These sensors are used to produce a state estimate for the vehicles global position, velocity, and orientation. The global positioning accuracy depends on the onboard GPS and barometer. GPS position-

ing error has a magnitude of 2.8m horizontally with 95% certainty, assuming an ideal configuration of satellites [Kaplan and Hegarty 2006]. Altimiter accuracy and precision suffer from local atmospheric effects. In informal bench testing weve found the on-board altimeters altitude estimate drifts in excess of 2 meters over 30 minutes. In contrast, the quadrotors GPS-based velocity measurements are accurate to 0.1m/s [4].

This quadrotor runs the open source ArduPilot software on its onboard Pixhawk autopilot computer [3], which provides a set of command to control the quadrotor over the telemetry radio link using the MAVLink protocol. So we could stream a sequence of GUIDED and SET ROI commands from a ground station to the quadcoper via the MAVLink protocol. This stream commands the quadrotor to move to a given position and to point the camera to a given region of interest. We start our autonomous flight by sending the quadrotor a TAKEOFF then GUIDED message to fly to the start of the user's camera trajectory. Once our design tool detects that the quadrotor has reached the start of the trajectory, and has less than 1m/s velocity, we trigger this sequence of messages to fly the quadrotor along the camera trajectory [1].

5. Evaluation

We have not carrier out quantitative evaluation of our system performance. The current aim of our project is to provide a correct and comfortable user experience. We have visually make sure the motion-caption and real-time view rendering is correct. We also tested the Oculus view.

6. Conclusion

Our system provides an intuitive way for an aerial cinematographer with no experience of operating quadrotor to easily control and plan the trajectory to film a predictable and high quality footage of a professional and smooth curved camera motion, which is usually hard to be done by pure manual control by an entry-level pilot.

7. Collaboration

Qian contributes to build the registration and correspponding algorithm, and the infrastructure of the system. Botao developped the web application, solved the hardware issues, and conducted the arial filming. Especially, we thank N. Joubert, who provides us the trajectory smooth algorithm.

8. Discussion and Future Work

At the moment, we use a Oculus Rift headset for the 3D view of the planner and a separate monitor screen for the first-person camera view. To provide a complete 3D route planning and preview experience, we want to overlay image of the FPV screen onto the Oculus view, so that the user can look at the first-person-view monitor in the virtual 3D world.

We will also replace the mocap motion capturing system with a consumer-friendly solution, like HTC Vive which have active SLAM indoor tracking, or Oculus Cresent Bay with a consumer-level $5'' \times 5'' \times 5''$ tracking volume.

9. Appendix

Two short videos of our project demo have been uploaded to Youtube.
Motion control: `https://youtu.be/YTZLxA6VP0E`.
Demo: `https://youtu.be/YdTbDrqGXa4`

References

[1] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan. Designing Feasible Trajectories for Quadrotor Cameras, 2015. *Manuscript in preparation*.

[2] 3DRobotics. IRIS+. http://3drobotics.com/iris/, 2014.

[3] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. Autonomous Robots 33, 12, 2012.

[4] U-BLOX. LEA-6 data sheet, docu. no. GPS.G6-HW-09004-E2. http://www.u-blox.com/images/downloads/Product_Docs/LEA-6_DataSheet_%28GPS.G6-HW-09004%29.pdf, 2013.