Optimization-based User Support for Cinematographic Quadrotor Camera Target Framing

Christoph Gebhardt
Otmar Hilliges
AIT Lab, ETH Zürich

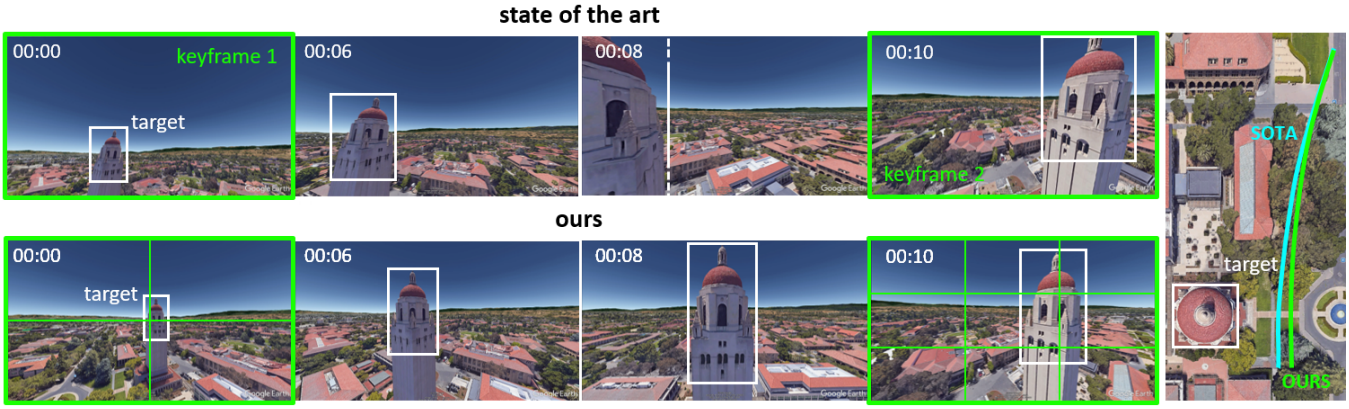

Figure 1: Quadrotor camera tools generate trajectories based on user-specified keyframes. Existing tools interpolate keyframes to frame camera targets. This can lead to visually unappealing results. *Top row*: existing tools cause the target object (the tower) to be featured at varying screen positions during the duration of the video. *Bottom row*: our method optimizes the camera pose such that the tower is positioned according to videographic compositional rules and and it is entirely visible throughout the shot (illustrative example).

ABSTRACT

To create aesthetically pleasing aerial footage, the correct framing of camera targets is crucial. However, current quadrotor camera tools do not consider the 3D extent of actual camera targets in their optimization schemes and simply interpolate between keyframes when generating a trajectory. This can yield videos with aesthetically unpleasing target framing. In this paper, we propose a target framing algorithm that optimizes the quadrotor camera pose such that targets are positioned at desirable screen locations according to videographic compositional rules and entirely visible throughout a shot. Camera targets are identified using a semi-automatic pipeline which leverages a deep-learning-based visual saliency model. A large-scale perceptual study (N≈500) shows that our method enables users to produce shots with a target framing that is closer to what they intended to create and more or as aesthetically pleasing than with the previous state of the art.

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445568

CCS CONCEPTS

• **Computing methodologies** → **Robotic planning**; • **Computer systems organization** → *External interfaces for robotics.*

KEYWORDS

Computational Design; Aerial Videography; Quadrotor Camera Tools; Trajectory Optimization

1 INTRODUCTION

The framing of landmarks is an important factor in the visual quality of scenic aerial video shots [18]. Professional videographers emphasize that during flight, it is necessary to continuously control and fine-tune the framing of a camera target, precisely positioning it on the image plane of the video, in order to create desired compositional effects (e.g., a simultaneously moving foreground and background, see [video]). Similarly, literature of cinematography highlights the importance of target framing for creating visually appealing cinematographic compositions [2].

Quadrotor camera tools have been proposed to bring the creation of aerial videos to end-users [17, 25]. These tools allow users to design videos by specifying keyframes in a realistic virtual environment (e.g., Google Earth), using them as references to generate physically feasible trajectories. Trajectories can then be previewed in these tools and executed with a robot in the real world. Extensions have been proposed that provide users with optimization-based support for parts of the design process that are difficult to conduct,

such as designing flight plans that are physically feasible [37] or feature globally smooth and visually appealing camera motion [19].

However, no existing design tool computationally supports users in framing camera targets. Previous approaches [18, 19] simply interpolate camera angles between keyframes. Other methods [17, 26, 37] frame targets by optimizing the camera pose such that its center forward vector points towards a look-at position trajectory. Look-at trajectories are constructed by interpolating the intersections of the center rays of user-specified keyframes with the virtual environment of the tool. Both types of approaches do not consider the 3D extent of actual camera targets in their optimization formulation. As a result, these approaches cannot constrain the image space position of camera targets between keyframes. This results in target objects that move freely in the video's image plane, potentially causing visually unappealing aerial videos. Furthermore, existing approaches cannot optimize trajectories such that camera targets adhere to specific screen locations. However, positioning targets at desirable image space locations constitutes aesthetically pleasing framing [2].

Figure 1 illustrates this problem (top row). A novice user sets two keyframes that capture the intended target, the tower's upper part, at undesirable image space positions. Both keyframes show the tower in a tilted orientation aligned at random positions on the screen. State-of-the-art methods leave these keyframes unadjusted and allow image space positions to be unconstrained between keyframes when generating a trajectory. This causes the video to depict the tower on various screen positions throughout the video, even cutting it off in parts.

In this paper, we tackle this problem by supporting users in the creation of aesthetically pleasing target framing. We propose a pipeline to identify landmarks of interest from coarse user input and an optimization formulation to generate trajectories such that target objects are continuously framed at desirable screen positions. Using our approach on the same input of the tower shot produces a video where the target is entirely visible through out the shot and aligned between keyframes (see bottom row, Figure 1). In addition, user's target framing is corrected to adhere to videographic compositional rules (e.g., Rule of Thirds in keyframe 2).

To achieve this behavior, we extend the approach by Gebhardt et al. [19] to optimize the camera pose according to image space constraints on the reprojected screen position of camera targets. More specifically, we propose an objective term that corrects the target position in the image space to align with desirable screen positions as defined by videographic compositional rules, e.g., Rule of Thirds [41]. Between keyframes, our approach constraints the movement of camera targets in the image space throughout the shot. To ensure that a generated target framing follows the user intent, optimized target screen positions are snapped to the compositional rule that is closest to user-specified target screen locations. Furthermore, we introduce a cost term which ensures that the entire camera target is visible during the duration of the video. This is achieved by optimizing the camera pose such that a target is always located within the extended camera frustum. The algorithm is formulated as well-behaved non-convex problem.

Our second contribution is a semi-automatic camera target identification pipeline. The pipeline leverages a deep-learning-based visual saliency model [28] to estimate a likely 2D bounding box of a camera target. The estimated area can be refined by the user if desired. Using ray casting and a point cloud clustering method, 3D camera targets are identified based on the bounding boxes and incorporated into the optimization problem.

We demonstrate the benefit of our method in a large-scale perceptual study ($N \approx 500$). Results show that viewers consider videos generated with our method to better match videographers' intent. Furthermore, they feature a target framing that is more or as aesthetically pleasing than the framing of videos generated with the state of the art. In addition, we show the positive effect of our method on visual quality in simulated and real-world aerial videos.

In summary, we contribute: (i) an optimization formulation that adjusts user-specified keyframes to feature camera targets at desirable image space positions and constraints their screen position between keyframes, (ii) an additional objective term that ensures targets to be in the field of view of the camera throughout the shot, (iii) a semi-automatic pipeline that identifies the 3D camera targets that users intend to capture from their specified keyframes, and (iv) the empirical evidence that our approach improves the perceived visual quality of end-user-designed aerial videos.

2 RELATED WORK

Camera Control in Virtual Environments: Camera placement [30] and motion planning [29, 44] have been studied extensively in virtual environments [9]. A recent work uses a deep learning approach to learn automated camera control from real film sequences [24]. These works share our goal of assisting users in the creation of camera motion and introduced the idea to define viewing constraints in image space (c.f., [13, 20, 31, 32]). Similarly, several works propose systems that support users in the creation of virtual film by providing information about cinematographic rule violations [10] or by taking users' high-level description of the film they intend to create and generating a video that adheres to their specifications as well as cinematographic conventions [4, 14]. However, it is important to consider that virtual environments are not limited by real-world physics or robot constraints and thus can produce arbitrary camera paths that are potentially not executable with a quadrotor.

Real-time Aerial Videography in Dynamic Scenes: Recently, several works address the generation of quadrotor camera trajectories in real-time to record dynamic scenes. Rousseau et al. [38] and Sabetghadam et al. [39] propose Model Predictive Control (MPC) approaches to follow a target and Poiesi and Cavallaro [36] introduced a particle swarm method that captures a moving target with multiple quadrotors. However, these works do not optimize a target's image space position. Galvane et al. [15, 16] and Joubert et al. [25] plan camera motion in a lower dimensional subspace considering quadrotor to target orientation and target positioning in image space. Similarly, Nägeli et al. [34] use an MPC to optimize visibility and position on the screen of camera targets, subject to robot constraints for a single quadrotor. The authors extend this work for multiple drones and allow actor-driven tracking on a geometric path using a Model Predictive Contour Control (MPCC) approach [35]. Ashtari et al. [3] expand the capability of such controllers by proposing a quadrotor model that enables subjective first-person-view shots, normally attained with steadicams or shoulder rigs. All of these works assume the camera targets to be known. In contrast,

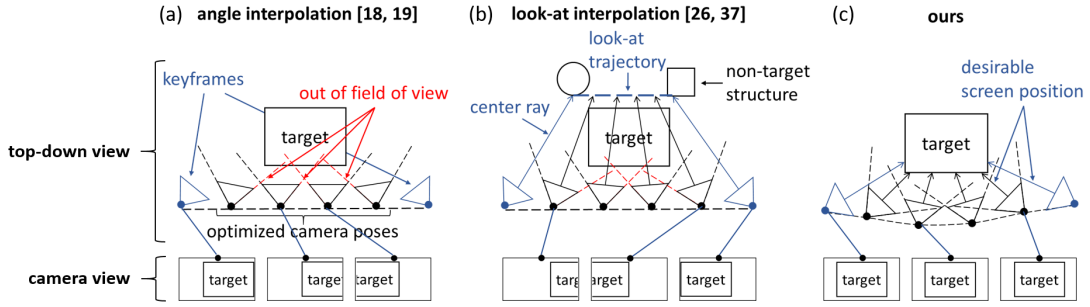

Figure 2: Schematic illustration of the difference between existing target framing approaches that (a) interpolate angles between keyframes [18, 19] or (b) the intersections of keyframe center rays with the environment [26, 37] and (c) our method that considers the actual position of a target in the optimization scheme and generates trajectories that ensure that the target is featured at desirable image space positions and that the target is visible through the entire shot (by being located within the extended camera frustum).

we propose a semi-automatic approach that identifies camera targets in a scene from sparse user input. Furthermore, we introduce a cost term that ensures that targets are not cropped but fully visible through the entire shot.

In addition, several different learning-based approaches have been proposed. Gschwindt et al. [21] develop a Reinforcement Learning approach that learns to automatically chose aesthetically favorable view points depending on a camera's position with respect to a moving human in simulation. Bonatti et al. [5, 6] and Huang et al. [22] propose pipelines that enable the autonomous capturing of videos of a single moving target. Their methods automatically detect the target and generate trajectories that smoothly follow it while avoiding collisions and preserving artistic guidelines.

Complementary to these works, we focus on the global planning aspects of aerial videography to support users in the design and recording of aerial videos of city- and landscapes. Our method aims at supporting users by optimizing the target framing such that it adheres to videographic compositional rules as well as their intent.

Quadrotor Camera Tools: A number of tools for the planning of quadrotor camera motion exist in both, industry and research. Commercially available applications usually place waypoints on a 2D map or allow to interactively control the quadrotor's camera as it tracks a pre-determined path [1, 11, 42, 45]. These tools generally do not provide means to ensure feasibility of the resulting plans and do not consider aesthetic objectives. In research, the planning of physically feasible quadrotor camera trajectories has recently received a lot of attention. Design tools allow for planning of aerial shots in 3D virtual environments [17, 18, 26, 43] and employ optimization to ensure that aesthetic objectives and robot modeling constraints are satisfied. Extensions of these tools help users to stay within the feasible range of quadrotor motion [37] and create videos that feature aesthetically pleasing camera dynamics [19].

Gebhardt et al. [17] introduced a tool where users specify the position of the quadrotor in space and for each of these positions select a camera target. Their method then optimizes the camera rotation such that the center of the target is depicted in the center of the video frame. Similarly, Xie et al. [43] propose an optimization-based approach that requires user to only specify a quadrotor's start and end position as well as landmarks of interest and then generates a trajectory utilizing a landmark-centric view quality field. The view field optimizes the camera pose such that the salient parts of landmarks are positioned on the image plane according the Rules of Thirds. These approaches automate the framing of camera targets and, hence, do not leave compositional control to users. In contrast, our method aims at optimizing target framing to be aesthetically pleasing while adhereing to users' intent.

Other tools [18, 19, 26, 37] allow users to specify keyframes by controlling the position and orientation of a camera in the virtual environment. To attain a reference camera orientation between keyframes, Joubert et al. [26] and Roberts and Hanrahan [37] interpolate the intersection points of keyframe center rays with the 3D environment. The resulting look-at-trajectory is then optimized such that the current position on the target trajectory is shown in the center of the camera's image frame. Similarly, Gebhardt et al. [18, 19] optimize camera orientations by following the interpolation of keyframes' pitch and yaw angles. These approaches do not model camera targets or approximate them as a single point which can yield to undesirable target framing (see Section 3). In contrast, our approach considers the targets users intend to capture in the optimization formulation. Our method optimizes camera position and orientation such that targets are captured at desirable image space locations and entirely visible throughout the shot.

3 METHOD OVERVIEW

Quadrotor camera tools should support users to design aerial videos that feature an aesthetically pleasing target framing. In order to achieve this, they should fulfill a set of requirements. This includes (a) facilitating users to position camera targets at visually appealing screen positions according to videographic compositional rules [2, 41]. In addition, approaches should ensure that (b) screen positions of camera targets are constrained between keyframes and that (c) targets are fully visible throughout the entire shot.

Existing quadrotor camera optimization schemes do not fulfill these requirements. The approach of Gebhardt et al. [18, 19] simply interpolates camera angles between the user-specified keyframes (see Figure 2a). This does not allow to position targets at desirable image space positions and results in video frames where the target

moves freely on the screen, even partly out of sight (red extended camera frustums in Figure 2a). Other approaches [26, 37] model camera targets as look-at trajectories that are constructed by interpolating the intersections of keyframes' center rays with the virtual environment. This is only a valid model for camera targets that are located in the center of the keyframe image. If a camera target is positioned on other screen locations, the center ray will intersect with other volumes. This causes the look-at trajectory to specify counterintuitive directional camera references, resulting in videos that also frame targets in an undesirable manner (e.g., moving out of sight, see Figure 2b).

To satisfy above requirements, we propose an optimization formulation that considers the position of a camera target on the image plane when optimizing the pose of a quadrotor camera (Section 4). This allows to correct the target framing to follow well-established compositional rules of photo- and videography like the Rule-of-Thirds, Golden Ratio, or Phi Grid [41]. In addition, our approach optimizes a trajectory such that the target screen position interpolates desirable screen locations between keyframes. Furthermore, it ensures that targets are entirely visible throughout the shot. This generates trajectories that overcome the problems of the previous state of the art (see Figure 2c). Thus, we minimize a cost functional over an infinite time horizon according to a dynamical robot model to find optimized values for system states $\mathbf{x}$ and inputs $\mathbf{u}$. As a high-level abstraction, the cost function

$$\underset{\mathbf{x},\mathbf{u}}{\text{minimize}} \sum \underbrace{C_{\text{smooth-motion}}(\mathbf{x},\mathbf{u})}_{[19]} + \underbrace{C_{\text{framing}}(\mathbf{x})}_{\text{Eq. 3}} + \underbrace{C_{\text{visibility}}(\mathbf{x})}_{\text{Eq. 11}},$$

serves three main purposes: 1) ensuring globally smooth camera motion ($C_{\text{smooth-motion}}$) by building on an existing optimization algorithm [19], 2) it optimizes the screen position of the targets to follow cinematographic compositional rules (C_{framing}), and 3) ensures that camera targets are entirely visible throughout the shot ($C_{\text{visibility}}$). Our optimization method requires knowing the 3D position and size of camera targets that users intend to capture. To aid users in this process, we contribute a semi-automatic target identification pipeline that determines 3D camera targets from sparse user input (Section 5).

4 OPTIMIZING CAMERA TARGET FRAMING

We propose two objective terms to frame targets at desirable screen positions and ensure their visibility throughout a shot. Thus, we expand an existing variable, infinite horizon, contour-following algorithm [19], using an approximated physical model of the quadrotor camera [17] (see Appendix C and D for details on model and algorithm). The optimization is considered an infinite-horizon algorithm because the time step along the trajectory can vary. In this section, we first introduce our target model (Section 4.1). We discuss then the details of our algorithm (Sections 4.2-4.4). See Appendix A for a table of notations and Appendix E for an explanation of the coordinate frames of our algorithm.

4.1 Target Model

For our target model, we assume that in scenic aerial video shots the quadrotor camera focuses on one target per frame and targets change between frames. We believe this is a valid assumption for a moving camera as human perceptual constraints prevent us from focusing on more than one point at a time. This is also reflected in popular aerial video techniques which work with a single point of focus [8]. Using our target identification pipeline (Section 5), we attain the assigned camera target t_j of each keyframe k_j. To incorporate these targets into our optimization, we fit a bounding cylinder with radius r_t, height h_t, and a center at position $\mathbf{p}_t \in \mathbb{R}^3$. We use this primitive as its geometric properties allow us to consider the 3D expansion of a target in our formulation without needing to sample vertices during optimization.

To be able to optimize a trajectory for globally smooth motion, the algorithm [19] in which we include our objective terms uses a time-free reference parameterization. Therefore, a chord length parameterized spline is used, where the parameter θ describes progress on the spatial reference path of the quadrotor defined as $\mathbf{r}_d(\theta) \in \mathbb{R}^3$. We use the same parameterization to compute a target reference spline that interpolates position, radius and height of the consecutive targets t_j of the video. The spline is defined as $\mathbf{f}_t(\theta) = [\mathbf{p}_t(\theta), r_t(\theta), h_t(\theta)] \in \mathbb{R}^5$. Figure 3 illustrates the spline of the target position $\mathbf{p}_t(\theta)$ that is defined between θ_0 and θ_N.

4.2 Framing Optimization

With our method, we optimize the pose of the quadrotor camera to position 3D targets at desirable locations in the image frame. Therefore, we need a reference function that specifies desirable image space locations for our target reference spline $\mathbf{f}_t(\theta)$ when observed from the according quadrotor position $\mathbf{r}_d(\theta)$.

We start by specifying a set of vectors L in the camera clip space C that define desirable screen positions according to cinematographic rules. In our implementation, L contains the center of the image plane ($\mathbf{l}^c = [0, 0, 0, 1]^T$) and the intersection points of the Rules of Thirds (e.g., $\mathbf{l}^r = [0, \frac{1}{3}, \frac{1}{3}, 1]^T$, see blue dashed lines in Figure 3).

Second, we compute the directional vector $\mathbf{m}_j^v$ between the position $\mathbf{r}_{d,j}$ of each keyframe and the position of its target $\mathbf{p}_{t,j}$ in camera view space V. This vector is computed as

$$\mathbf{p}_j^v = \mathbf{R}_{\psi,\phi}(\psi_{d,j}, \phi_{d,j})(\mathbf{r}_{d,j} - \mathbf{p}_{t,j}) \tag{1}$$

$$\mathbf{m}_j^v = \frac{\mathbf{p}_j^v}{||\mathbf{p}_j^v||}$$

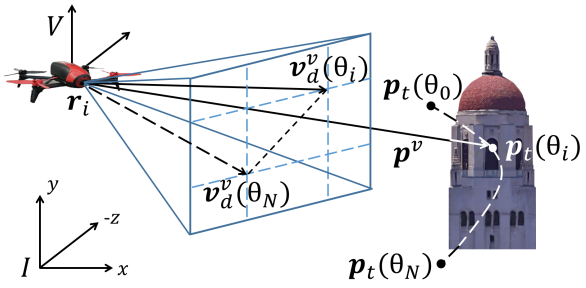

Figure 3: Schematic illustration of the framing cost term that optimizes the pose of the quadrotor camera such that the relative vector between camera and target $\mathbf{p}_v$ aligns with the reference vector $\mathbf{v}_d$ that positions the target at a desirable image space position.

where $\mathbf{p}_j^v$ the view space position of t_j, $\psi_{d,j}$ and $\phi_{d,j}$ are the pitch and yaw orientation of k_j and $\mathbf{R}_{\psi,\phi} \in SO(3)$ is the rotation matrix from world frame I to camera view space C.

For each keyframe, we identify the vector in L that is closest to the actual directional vector between t_j in k_j as

$$\mathbf{v}_j^v = \underset{\mathbf{l} \in L}{arg\ min}\ ||\mathbf{m}_j^v - w(\mathbf{R}_c^{-1}\mathbf{l}^c)||\,, \tag{2}$$

where $\mathbf{v}_j^v$ is the closest vector, $\mathbf{R}_c \in SO(4)$ is the camera matrix that performs rotations from the camera's view space V to its clip space C and w the function that normalizes homogeneous to Cartesian coordinates. This selects an image space position which follows cinematographic rules while still being in close distance to users' originally defined target screen position. Using the θ parameterization, the reference directional vectors $\mathbf{v}_j^v$ are linearly interpolated to attain the function $\mathbf{v}_d^v(\theta)$ (see Figure 3). In case users want to keep the exact framing they specified, one can compute $\mathbf{v}_d^v(\theta)$ by directly interpolating the $\mathbf{m}_j^v$ vectors.

With the reference function for desired target framing in place, we can now define the term that optimizes camera target framing:

$$\mathbf{p}^v = \mathbf{R}_{\psi,\phi}(\psi_{q,i} + \psi_{g,i}, \phi_{g,i})(\mathbf{p}_t(\theta)) - \mathbf{r}_i) \tag{3}$$
$$c_f(\theta), \mathbf{p}^v) = ||\frac{\mathbf{p}^v}{||\mathbf{p}^v||} - \mathbf{v}_d^v(\theta)||^2,$$

where $\mathbf{p}^v$ is the position of the target in V and $\mathbf{r}_i$ the position of the quadrotor at a specific stage i of the optimization horizon. Figure 3 illustrates the cost term. It optimizes the pose of the quadrotor camera such that the relative vector between quadrotor and target $\mathbf{p}^v$ aligns with the directional reference vector $\mathbf{v}_d^v(\theta)$ to position the target at the desired image space position.

4.3 Visibility Maximization

With the framing optimization cost term in place, one can ensure that the camera target is at the desired position in the image plane across the entire trajectory. However, if the distance between camera and camera target is small, it is possible that although the center of the camera target is at the desired position on the image plane, large parts of the target are not captured because they get cropped. While this is most likely not the case at keyframes, they are specified by users, it can occur between them where reference positions are attained via interpolation (as with the state of the art, Figure 1 top row). To address this problem, we propose a second cost term that ensures that camera targets are entirely visible in each frame of the video. Figure 4 illustrates the intuition behind the objective term. It introduces a penalty when any part of the bounding cylinder of a camera target intersects with a plane of the extended camera frustum and, hence, would not be visible.

For this cost term, we first calculate four points on the edges of the cylinder. Since the roll of the quadrotor camera is always zero, one can attain the points at top $\mathbf{p}_t^v$ and bottom $\mathbf{p}_b^v$ of the cylinder as

$$\mathbf{p}_t^v = \mathbf{p}^v + \left[0,\ \frac{h_t(\theta)}{2},\ 0\right]^T \tag{4}$$

$$\mathbf{p}_b^v = \mathbf{p}^v - \left[0,\ \frac{h_t(\theta)}{2},\ 0\right]^T. \tag{5}$$

To find the left and the right edge of the cylinder from the perspective of the camera, we search the points that have the z value

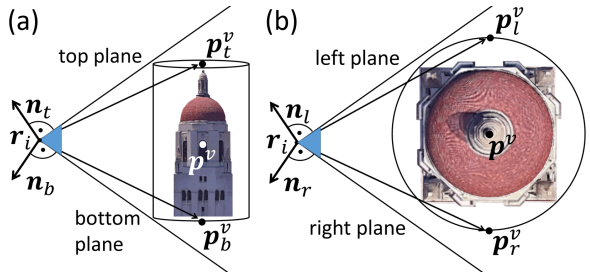

Figure 4: Schematic illustration of the visibility maximization cost term showing a camera target and its bounding cylinder from (a) side and (b) top. The term introduces a penalty if the cylinder intersects with a plane of the extended camera frustum.

of $\mathbf{p}^v$, the distance r_t from $\mathbf{p}^v$ and are on the plane that has $\mathbf{p}^v$ as its normal. By substituting given values into the point-normal form of this plane and simplifying it, we attain the x-values of these points:

$$x = x_{p^v} + \frac{y_{p^v}^2 - y_{p^v} y}{x_v}, \tag{6}$$

where x_{p^v}, y_{p^v} are the respective x- and y-values of $\mathbf{p}^v$. Substituting x into the 2-norm equation that computes the distance between $\mathbf{p}^v$ and one of the points on the edge we attain the y-values.

$$y_{1,2} = y_{p^v} \pm \sqrt{\frac{r_t^2}{\left(\frac{y_{p^v}^2}{x_{p^v}^2} + 1\right)}} \tag{7}$$

Substituting the respective y-values back into Equation 6 we can the specify $\mathbf{p}_l^v = [x_2, y_2, z_v]^T$, $\mathbf{p}_r^v = [x_1, y_1, z_v]^T$.

To attain the planes that describe the camera frustum, we first project the corners of the image plane into camera view space:

$$\mathbf{q}_{tl}^v = \mathbf{R}_c^{-1}\mathbf{c}_1, \mathbf{q}_{tr}^v = \mathbf{R}_c^{-1}\mathbf{c}_2, \mathbf{q}_{bl}^v = \mathbf{R}_c^{-1}\mathbf{c}_3, \mathbf{q}_{br}^v = \mathbf{R}_c^{-1}\mathbf{c}_4 \tag{8}$$

where $\mathbf{c}_{1-4}$ are the four corners of the image plane defined in camera clip space and $\mathbf{v}_{tl}, \mathbf{v}_{tr}, \mathbf{v}_{bl}, \mathbf{v}_{br}$ are the top-left, top-right, bottom-left and bottom-right vectors that form the edges of the camera frustum. We then compute the normals of the planes of the frustum:

$$\mathbf{n}_t^v = \mathbf{q}_{tl}^v \times \mathbf{q}_{tr}^v, \mathbf{n}_b^v = \mathbf{q}_{bl}^v \times \mathbf{q}_{br}^v, \tag{9}$$
$$\mathbf{n}_l^v = \mathbf{q}_{tl}^v \times \mathbf{q}_{bl}^v, \mathbf{n}_r^v = \mathbf{q}_{tr}^v \times \mathbf{q}_{br}^v.$$

And we define a visibility cost function that returns the squared minimum of two distances and zero if they have different signs:

$$f_{vis}(d_1, d_2) = \begin{cases} 0 & \text{if } |d_1 + d_2| < |d_1| + |d_2| \\ min(d_1^2, d_2^2) & \text{otherwise.} \end{cases} \tag{10}$$

The visibility cost term is computed by summing the result of Equation 10 for the distances between edge points and planes:

$$f_{all}(\mathbf{n}_{t-r}, \mathbf{p}_{t-r}^v) = \sum_{\mathbf{p}_m}^{M} f_{vis}(\mathbf{n}_t^T \mathbf{p}_m, \mathbf{n}_b^T \mathbf{p}_m) + f_{vis}(\mathbf{n}_l^T \mathbf{p}_m, \mathbf{n}_r^T \mathbf{p}_m)$$

$$c_v(\mathbf{n}_{t-r}, \mathbf{p}_{t-r}^v, x_{\mathbf{p}^v}) = \begin{cases} 0 & \text{if } x_{\mathbf{p}^v} < 0 \\ f_{all}(\mathbf{n}_{t-r}, \mathbf{p}_{t-r}^v) & \text{otherwise,} \end{cases} \tag{11}$$

where $M = [\mathbf{p}_t^v, \mathbf{p}_b^v, \mathbf{p}_l^v, \mathbf{p}_r^v]$ is the set of cylinder points and $\mathbf{n}^T\mathbf{p}$ is the distance between a point $\mathbf{p}$ and a plane specified by the normal

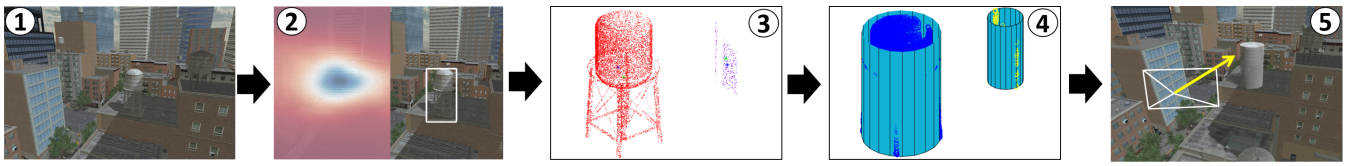

Figure 5: The target identification pipeline: (1) for each user-specified keyframe k_j, (2) the visually most interesting 2D area is predicted and refined if necessary. (3) Rays are casted through these areas of all keyframes and intersections with the 3D environment are saved in a point cloud. This point cloud is segmented and (4) a cylinder is fit to each cluster. (5) The cylinder with the lowest distance from the center forward vector of a keyframe k_j is selected as its target t_j.

$\mathbf{n}$. Revoking the costs if $x_{\mathbf{p}^v} < 0$ ensures that the term is only active if the target is in front of the camera.

Intuitively, this objective ensures that the four points on the hull of the cylinder (M) are located within the extended camera frustum. This is satisfied if the points on two opposing sides of the cylinder are located on the opposite sides of the respective planes (Equation 10). For instance, $\mathbf{p}_l^v$ is located to the right of the left plane and $\mathbf{p}_r^v$ is located to the left of the right plane (see Figure 4). If not, the distance between the point on the side of the cylinder that cuts the camera frustum and the intersected plane gets minimized. The term achieves the desired outcome as the framing objective (Equation 3) constraints the target to be located in the viewing direction of the camera.

4.4 Optimization Problem

We construct our overall objective function by linearly combining the equations of the *variable infinite horizon contour following algorithm* J_i (Equation 16, [19]), the *framing optimization objective* c_f (Equation 3) and the *visibility maximization objective* c_v (Equation 11). The final optimization problem is then:

$$\underset{\mathbf{x},\mathbf{u},\boldsymbol{\Theta},v}{\text{minimize}} \sum_{i=0}^{N} J_i + w_f c_f(\theta, \mathbf{p}_v) + w_v c_v(\mathbf{n}_{t-r}, \mathbf{p}_{t-r}^v, x_{\mathbf{p}^v}) \tag{12}$$

$$\begin{aligned}
\text{subject to } & \mathbf{x}_0 = \mathbf{k}_0 && \text{(initial state)}\\
& \boldsymbol{\Theta}_0 = \mathbf{0} && \text{(initial progress)}\\
& \boldsymbol{\Theta}_N = \mathbf{L} && \text{(terminal progress)}\\
& \mathbf{x}_{i+1} = A\mathbf{x}_i + B\mathbf{u}_i + g && \text{(dynamical model)}\\
& \boldsymbol{\Theta}_{i+1} = C\boldsymbol{\Theta}_i + Dv_i && \text{(progress model)}\\
& \mathbf{x}_{min} \le \mathbf{x}_i \le \mathbf{x}_{max}, && \text{(state bounds)}\\
& \mathbf{u}_{min} \le \mathbf{u}_i \le \mathbf{u}_{max}, && \text{(input limits)}\\
& \mathbf{0} \le \boldsymbol{\Theta}_i \le \boldsymbol{\Theta}_{max} && \text{(progress bounds)}\\
& \mathbf{0} \le v_i \le v_{max} && \text{(progress input limits)}\\
& -\pi \le d(\psi_{q,i} + \dot{\psi}_{g,i}, \phi_{g,i}) \le \pi && \text{(angular change limits),}
\end{aligned}$$

where J_i is quadratic in $\mathbf{x}$, $\mathbf{u}$, v and linear in $\boldsymbol{\Theta}$ and c^f and c^v are non-linear in $\mathbf{x}$. When flying a generated trajectory we follow the optimized positional trajectory $\mathbf{r}$ with a standard LQR-controller and use velocity and accelerations states of $\mathbf{x}$ as feed-forward terms.

5 TARGET IDENTIFICATION

For our optimization method to work, we need to know the position and size of the camera targets users intend to capture with a specific shot. In contrast to related work in dynamic aerial videography that estimates the 3D pose of a single human actor [5, 6, 22], we are interested in landmarks (buildings, trees, mountains, etc.). As our tool builds upon realistic 3D environments like Google Earth, we do not need to reconstruct the real world but want to identify the 3D structures within the scene users intend to capture. Our goal is to develop an approach that eases the process of specifying the 3D camera targets that are used as reference in the optimization method. Thus, we attempt to infer the targets a user intends to capture from their specified keyframes. Figure 5 shows the design of our target identification pipeline. (1) The image of each keyframe k_j is feed into Deep Gaze II [28], a deep visual saliency (DVS) model. (2) The network estimates the visually most interesting area of this image. For each keyframe, the bounding box of this area is presented to users and can be adjusted if it does not frame the intended camera target well enough. (3) To attain the 3D camera targets in the scene, we cast rays through the 2D bounding boxes of all keyframes and save the intersections with the virtual environment in a point cloud. Using a point-cloud-clustering method [33], we segment the resulting point cloud to identify the point clouds of the individual targets. (4) For each target point cloud, we use a heuristic approach to compute a bounding cylinder by finding the maximal spread of its points in the left-right and top-bottom direction (see Appendix F for details). (5) Finally, we assign the cylinder as target t_j to keyframe k_j that has the lowest distance from the center forward vector of the keyframe.

During the development of the pipeline, we tested two approaches to infer 3D camera targets from keyframes without additional user input. Thus, we replaced step 2 of the pipeline to either solely rely on the DVS model (no user refinement) or the landmark-centric view quality field [43]. In a pre-study, both approaches did not perform significantly better than a simple baseline that assumed the target to always be located in the center of the keyframe image (see Supplementary Material). Only by allowing users to manually refine the target areas in keyframe images it was possible to attain the 3D camera targets they intended to capture.

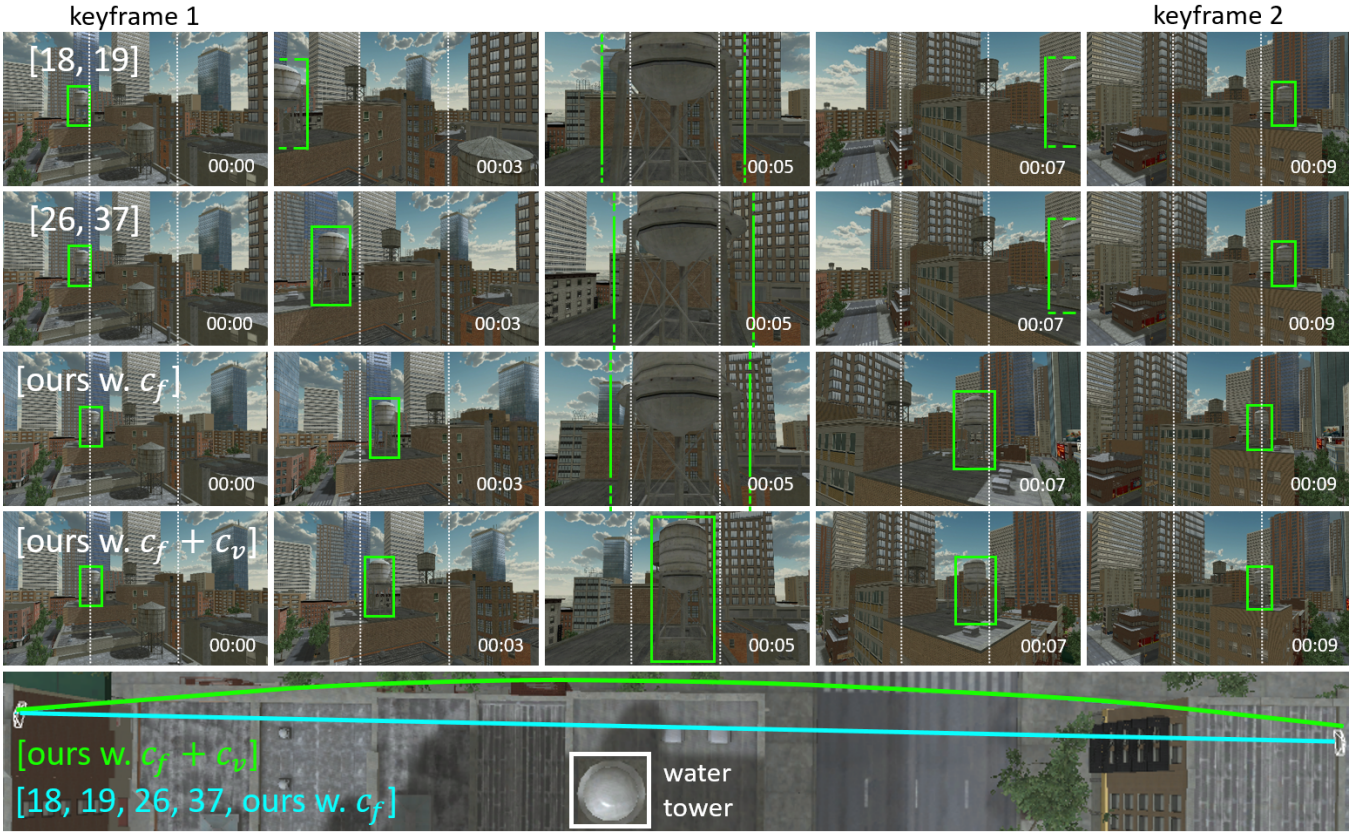

Figure 6: Illustrates the effect of different framing approaches on videos generated from two keyframes. For methods that interpolate angles [18, 19] or the intersection of the center rays of keyframes with the environment [26, 37], the camera target (the water tower) moves freely on the image plane of the videos. When generated with our method using the cost term c_f, the target is aligned with the (dashed white) vertical lines of the Rules of Thirds at the keyframes and transitions between them. Using both cost terms of our method, $c_f + c_v$, produces the same result and additionally corrects the framing such that the water tower is visible through the entire shot. The bottom row displays a top-down view on the generated trajectories.

6 IMPLEMENTATION

Our design tool is implemented using UNITY 3D. For the DVS model, we used a pre-trained PYTHON model of Deep Gaze II [27]. We chose this model as it is the state of the art according to the MIT Saliency Benchmark [7]. Point cloud segmentation and cylinder fitting are implemented in MATLAB. The different components of our tool communicate via network. Also the optimization problem is implemented with MATLAB and solved with the FORCES Pro software [12] which generates fast solver code, exploiting the special structure in the non-linear program. We set the horizon length of our problem to be $N = 100$. Furthermore, we introduced the function $d(\psi, \phi)$ to constrain the change of pitch and yaw between two consecutive stages of Equation 12. The function is defined as

$$d(\psi, \phi) = \begin{bmatrix} \dot{A}_\psi(\dot{\psi}) + \ddot{A}_\psi(\ddot{\psi}) + B_\psi(M_\psi) \\ \dot{A}_\phi(\dot{\phi}) + \ddot{A}_\psi(\ddot{\phi}) + B_\phi(M_\phi) \end{bmatrix}, \tag{13}$$

where $\dot{A}_{\psi/\phi}$ and $B_{\psi/\phi}$ are the forward propagation factors of the particular state or input and M_ψ, M_ϕ are torques acting on pitch and yaw of the quadrotor camera. This constraint is necessary to prevent the angular distance of pitch or yaw between two consecutive stages to be larger than 2π. In our tests, this happened in early iterations of the solver when Δt was arbitrary large and resulted in non-convergence of the problem.

6.1 Runtime Performance

We evaluated the optimization runtime of our method. Therefore, we generated all of the 60 trajectories of our study (Section 8) with our approach and the method of [19]. In both optimization schemes, we use the the approximated linear quadrotor model of Appendix C. We measured runtime on a standard desktop machine (Intel Core i7 4GHz CPU). On average, it took 14.16 s (SD: 6.63 s) to generate a trajectory with our method and 5.87 s (SD: 2.88 s) with the method of [19]. The increase in computation time can be explained with our method optimizing camera orientation in $SO(5)$ resulting in a non-linear objective function, while [19] tracks references per dimension resulting in an objective function that stays quadratic.

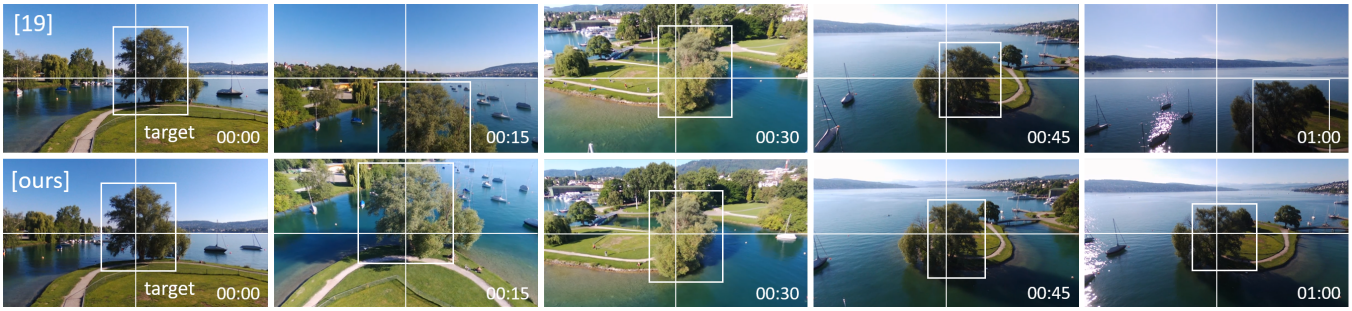

Figure 7: Qualitative comparison of video frames of two trajectories generated with [19] (top row) and our method (bottom row). The white cross illustrates the desired screen position and the bounding box frames the camera target.

7 QUALITATIVE RESULTS

To evaluate if our method has the desired effect on target framing, we design a challenging shot and generate trajectories with four different optimization schemes. The first is generated with the method that interpolates angles between keyframes [18, 19]. The second trajectory is generated with the target framing approach that interpolates the intersection of the center rays of keyframes with the environment [26, 37]. The third trajectory is generated with our method using only the framing optimization (c_f). The last is generated by using both cost terms (c_f and c_v) of our method.

Figure 6 shows a frame by frame comparison of the resulting videos of the four trajectories. For the previous methods [18, 19, 26, 37], the camera target (the water tower) moves freely around the image plane, partly disappearing in some frames. In contrast, the video generated with c_f of our method corrects the camera orientation of the keyframes to align the target with the closest vertical line of the Rules of Thirds (see dashed white lines). Between keyframes, the tower nicely transitions between these two lines. However, when the camera approaches the target closely, the tower is not entirely visible in some of the video frames. This is corrected by using both cost terms, c_f and c_v.

We also conducted a qualitative comparison by recording two videos with the same consumer grade drone (Parrot Bebop 2). The quadrotor followed trajectories generated with our method and with the approach of Gebhardt et al. [19] (same framing approach as [18]) using the same keyframes. The input data was designed in Google Earth using KLM-files and simulates a novice user who struggles to frame targets in an aesthetically pleasing manner when specifying keyframes. Figure 7 shows resulting video frames (also see [video]). Our method keeps the intended target nicely framed in the center of the image while the image space positions of the target for the baseline method [19] change drastically between frames. This provides evidence that the positive perceptual effect of our framing terms can even be seen in real-world shots where GPS-accuracy and other factors influence the resulting video.

8 PERCEPTUAL STUDY

We investigated the effect of our approach on the perceived match between generated and user-intended target framing compared to existing methods [18, 19, 26, 37]. In addition, we analyzed if our approach positively affects the aesthetic perception of aerial videos compared to the state of the art. To answer both questions, we conducted a pairwise perceptual comparison of videos generated with the mentioned approaches.

8.1 Experimental Design

We conducted the perceptual study in two steps. First, we invited a group of participants to design aerial videos using our quadrotor camera tool. Second, we created two video questionnaires for which we produced videos by simulating trajectories generated with the different methods. The input of all methods was the participant-designed keyframes from the first step. Each questionnaire item showed a video of a trajectory of our method and a representative of the state of the art. The questionnaires were answered by a new group of participants who, per item, indicated their preference for one of the two videos. In the following, we provide more details on the experimental design of our study.

Trajectory design: We recruited 15 participants from our institution (10 female, 5 male) to design aerial videos. The average age was 26.2 years (SD=3.3). Three of them were working in a photo- or videography-related job. Six were hobby photo- or videographers. The remaining participants reported no experience in photo- or videography. Using our quadrotor camera tool, participants were asked to design four free-form videos in two scenes showing two different virtual cities. For each video, we asked participants to specify four or more keyframes focusing on objects they find interesting. Participants used our target identification pipeline to specify the camera targets they intended to capture. To ensure that participants do not adjust keyframes to a particular target framing approach, they were only allowed to see their designed video after they finished editing keyframes. Finally, participants were asked to describe where on the image plane they wanted to position a particular camera target.

On average, participants spent 12.85 s (SD=9.52 s) to refine the bounding box of a camera target in step 2 of our target identification pipeline (see Figure 5). To refine the bounding boxes of all targets per trial, they needed on average 59.94 s (SD=28.61 s). The mean trial duration was 202.15 s (SD=84.54 s).

Conditions: Variations of trajectories generated from different optimization schemes can stem from differences in the underlying dynamical model, positional reference tracking, and target framing approach. In this study, we isolated variations in trajectories to stem

from differences in target framing. Thus, we used the algorithm of Gebhardt et al. [19] as the base algorithm of all conditions, ensuring that they are subject to the same dynamical model and positional reference tracing. We investigated three conditions:

(1) *Angle interpolation* (ANG) generated trajectories with the unmodified base algorithm [19] that interpolates pitch and yaw angles of the camera between keyframes (same framing approach as [18]).
(2) *Look-at trajectory* (LOOK-AT) represented the target framing approach [26, 37] that constructs a look-at trajectory by interpolating the intersections of center rays of keyframes with the virtual environment. The camera pose is then computed by orienting it such that the current position on the look-at trajectory is in the center of its screen. For this condition, we implemented this framing approach as cost term in the base algorithm [19]. The difference to the original method is that look-at and quadrotor trajectory (i.e., look-from trajectory) are parameterized by θ and progress is determined according to Equation 15.
(3) OURS generated trajectories using our proposed framing optimization formulation. This condition used the participant-specified camera targets as input.

Questionnaires: We designed two video questionnaires: INTENT and AESTHETICS. Both showed the same 10 video comparisons, 5 of which compared two videos rendered from trajectories of ANG and OURS. The other 5 comparisons were between videos of trajectories of LOOK-AT and OURS. We generated trajectories using the participant-designed keyframes. To ensure visual differences in videos, we chose the 5 trajectories that have the highest discrepancy in angular and positional distance between the respective condition and OURS. The discrepancy function is defined as

$$\underset{\mathbf{t}^k_{ours},\mathbf{t}^k_{baseline}\in T}{\arg\max} \sum_{i=0}^{N} ||\mathbf{p}^i_{ours} - \mathbf{p}^i_{baseline}|| + \text{angle}(\mathbf{q}^i_{ours}, \mathbf{q}^i_{baseline}) \,,$$

where $\mathbf{p}^i$ is the position of the quadrotor and $\mathbf{q}^i$ is its rotation as quaternion at stage i of the trajectory. $\text{angle}(\mathbf{q}_1, \mathbf{q}_2)$ is a function that returns the angular distance in degrees between two quaternions. $\mathbf{t}^k_{ours}$ and $\mathbf{t}^k_{baseline}$ are the trajectories of OURS and the respective baseline generated on the same set of keyframes $k \in K$.

For each question of both surveys, we showed participants images of the camera targets the videographer intended to capture. This was done to ensure that they focused on the right objects when watching the video despite the visual density of the virtual scenes. In both surveys, the two videos are placed side-by-side, randomly assigned to the left or right, and participants stated which video they preferred on a 5-point Likert scale. The question was: "which shot frames the objects aesthetically more pleasing?". The five responses are: "shot on the left side frames the objects much more pleasing", "shot on the left side frames the objects more pleasing", "both frame the scene the same", "shot on the right side frames the objects more pleasing", and "shot on the right side frames the objects much more pleasing".

The two questionnaires examined different objects of investigation. INTENT investigated if video viewers perceive differences in how good the videos of conditions match with the intended result of the videographer. Thus, we additionally displayed the image space position at which the videographer intended to position the target in its videos. These positions were specified according to the description of the respective participant that designed a video and were shown as a thin cross. AESTHETICS examined the effect of methods on the general aesthetic perception of videos and, hence, displayed videos unmodified.

Participants: We recruited 518 participants via mailing lists from our institution (202 female, 314 male, 2 preferred not to say). 279 participants answered the questionnaire INTENT and 239 answered AESTHETICS. The average age was 22.6 years (SD=3.2). 12 participants were working in a photography- or videography-related job, 116 were hobby photographers or videographers and 390 reported no experience in photo- or videography. Participants were not paid. We split the investigation into two questionnaires to keep the time necessary for their completion below 15 min.

8.2 Results

Assuming equidistant intervals, we mapped survey responses of both questionnaires onto a scale from -2 to 2, where negative values mean that the video of ANG or LOOK-AT is perceived as more pleasing, 0 indicates no difference and a positive value means that the video of OURS is perceived as more pleasing. To attain interval data, samples were build by taking the mean of the Likert-type results. As data normality was violated, we ran a Wilcoxon signed-rank test to estimate if means differ from zero.

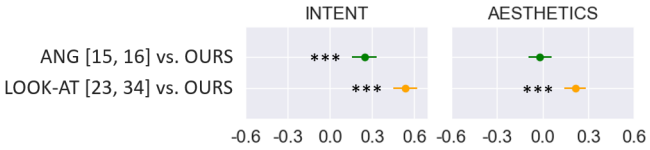

Figure 8: Mean and 95% confidence interval of the effect of optimization scheme on perceived match with videographer intent and aesthetics. Positive values indicate a preference for videos generated with our framing extension. Negative values indicate that videos generated with ANG or LOOK-AT are preferred. Significance notation is with respect to the null effect (zero).

Evaluating the responses of INTENT revealed a mean with a positive value and a high confidence for both comparisons (see Figure 8). Statistical testing showed that our method allows videographers to produce videos that better match with what they intended to create compared to ANG (M=0.25, SD=0.74; Z=23754.5, p<0.001) and LOOK-AT (M=0.53, SD=0.72; Z=28347.5, p<0.001). The mean of the data of AESTHETICS has a positive value with a high confidence for the comparison with LOOK-AT (M=0.21, SD=0.56; see Figure 8). The effect of our method on the aesthetics of target framing for this comparison is a significant (Z=16061.5, p<0.001). However, there is no significant difference between ANG and OURS in terms of the aesthetic perception of target framing (M=-0.02, SD=0.59; Z=10789.5, p=0.66).

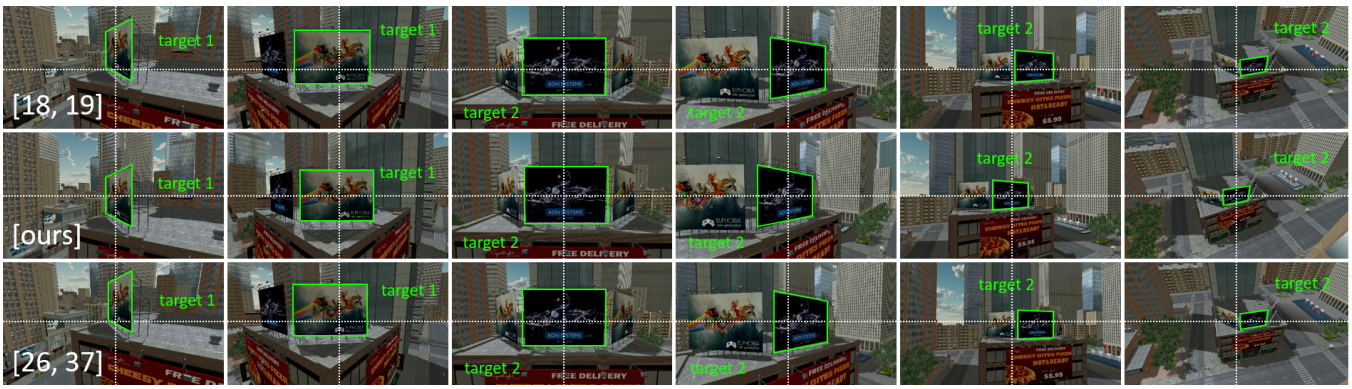

Figure 9: Qualitative comparison of a participant-designed video shot: OURS keeps the two targets at their intended screen position throughout the shot. ANG [18, 19] and LOOK-AT [26, 37] exhibit slight deviations between desired and generated target image space positions.

The qualitative inspection of the videos revealed that the video-designing participants benefited from our approach's capability to keep targets positioned at desirable image space locations. One example is shown in Figure 9 where OURS keeps the sequential targets of a video shot at their indented target position while ANG (top) and LOOK-AT (bottom) show slight offsets with respect to the intended image space positions.

Finally, we compared the difference between video ratings of self-declared expert and novice participants. A Mann-Whitney test did not reveal significant differences between their responses ($n_{experts}$=58, $n_{novices}$=181, ANG-AESTHETICS: U=5689, p=0.3, ANG-INTENT: U=6555, p=0.2, LOOK-AT-AESTHETICS: U=5199, p=0.9, LOOK-AT-INTENT: U=7668.5, p=0.5).

9 DISCUSSION

The results of the study indicate that participants perceived the videos of our approach to better match with videographers' intent compared to the state of the art [18, 19, 26, 37]. In terms of the aesthetic perception of videos, results are not as clear. Participants found our method to produce aesthetically more pleasing videos than the framing method of Joubert et al. [26] and Roberts and Hanrahan [37]. However, no significant differences are found in the comparison with the method of Gebhardt et al. [18, 19]. Nevertheless, the significant results in the other comparisons encourages us to suggest that our approach better supports users in framing camera targets.

The aesthetic perception of videos depends on factors that are not modeled in the objectives of the investigated optimization schemes, e.g., visual density or the lighting of the scene. We assume that the null result and the lower effect of comparisons in terms of aesthetics can be explained by participants that attribute differences in the videos to a variety of such unmodeled factors. In particular, we believe that due to the densely populated surroundings of camera targets in the virtual city scenes, the positioning of a target on the screen lost its perceptual importance. Leveraging this factor could allow to trade-off framing precision with computational costs by relaxing constraints in visually dense scenes.

Our framing extension had access to more user input compared to the other approaches. Participants spent ca. 30% of the time per trial on defining the 2D target areas. The equivalent of that in the other conditions would be to provide participants with more time to specify additional keyframes to refine the target framing. Setting additional keyframes would allow participants to work around methods' lack of explicitly modeling camera targets and their screen position. In contrast, specifying 2D areas on keyframe images to indicate a camera target provides additional domain-relevant information for the quadrotor camera trajectory optimization. Furthermore, we are convinced that the time needed to specify these bounding boxes can be significantly reduced by using appropriate interaction techniques, e.g., direct manipulation [23, 40].

Using the landmark-centric view quality field or the deep visual saliency model without additional user input were not sufficient for our target identification pipeline to identify camera targets users intended to capture. This indicates the difficulty of computationally identifying user intent and that even complex models, such as the DVS network, fail to capture all subjective differences. Our approach of manually refining desired camera targets highlights the potential of hybrid methods where easy interaction mechanisms allow users to communicate their individual intent to optimization-based adaptive user interfaces to better individualize support.

In future work, we will investigate the effect of our algorithm on the workflow of videographers. For instance, we will examine if using our algorithmic extension to design a video shot speeds up the process or helps users to create videos with higher visual quality. Furthermore, we will investigate if our algorithm differently effects workflow and design of experts, hobbyists or novices.

To advance the design of our algorithm, we will generalize the method to model the true shape of a camera target rather than approximating it with a cylinder. This could be achieved by sampling points from the 3D environment during iterations of the solver. Furthermore, we will incorporate a cost term for obstacle avoidance (similar to [17]) to further increase user support when designing aerial videos.

10 CONCLUSION

In this paper, we propose an approach that better supports users in designing aerial videos with quadrotor camera tools. The previous state of the art failed to produce videos with visually appealing target framing that adheres to what users intended to capture. This is due to these methods not considering the 3d extent of camera targets and simply interpolating keyframes when generating trajectories. In response, we propose a quadrotor camera tool that models target objects, allowing our approach to generate trajectories according to established framing objectives. Our approach identifies areas of landmarks users intend to capture from sparse user input using a semi-automatic target identification pipeline. Integrating these camera targets into a novel optimization formulations allows the generation of trajectories that frame targets at desired screen positions even between keyframes. In addition, we optimize the camera pose such that the whole volume of a target is located within the extended camera frustum ensuring its visibility throughout the shot.

We conducted a large-scale perceptual study (N≈500) where we compared our method with the state of the art [18, 19, 26, 37]. Results indicated that our approach produces videos that perceivably better match with the video users intended to create than other methods. Results also hint that the videos of our method are aesthetically more pleasing.

ACKNOWLEDGMENTS

We thank Roman Sattler for his work in the exploratory phase of the project and Seonwook Park for providing the video voice-over. We are also grateful for the valuable feedback of David Lindlbauer on study design and paper drafts.

REFERENCES

[1] APM. 2016. APM Autopilot Suite. Retrieved January 6, 2020 from http://ardupilot.com

[2] Daniel Arijon. 1976. Grammar of the film language. (1976).

[3] Amirsaman Ashtari, Stefan Stevsic, Tobias Naegeli, Otmar Hilliges, and Jean-Charles Bazin. 2020. Capturing Subjective First-Person View Shots with Drones for Automated Cinematography. *ACM Trans. Graph.* 0, ja (2020). https://doi.org/10.1145/3378673

[4] William H. Bares and James C. Lester. 1998. Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds. In *Proceedings of the 4th International Conference on Intelligent User Interfaces* (Los Angeles, California, USA) *(IUI '99)*. Association for Computing Machinery, New York, NY, USA, 119–126. https://doi.org/10.1145/291080.291101

[5] Rogerio Bonatti, Cherie Ho, Wenshan Wang, Sanjiban Choudhury, and Sebastian Scherer. 2019. Towards a Robust Aerial Cinematography Platform: Localizing and Tracking Moving Targets in Unstructured Environments. *arXiv preprint arXiv:1904.02319* (2019).

[6] Rogerio Bonatti, Wenshan Wang, Cherie Ho, Aayush Ahuja, Mirko Gschwindt, Efe Camci, Erdal Kayacan, Sanjiban Choudhury, and Sebastian Scherer. 2019. Autonomous Aerial Cinematography In Unstructured Environments With Learned Artistic Decision-Making. *arXiv preprint arXiv:1910.06988* (2019).

[7] Zoya Bylinskii, Tilke Judd, Ali Borji, Laurent Itti, Frédo Durand, Aude Oliva, and Antonio Torralba. 2012. MIT Saliency Benchmark.

[8] Eric Cheng. 2016. *Aerial Photography and Videography Using Drones*. Vol. 1. Peachpit Press.

[9] Marc Christie, Patrick Olivier, and Jean-Marie Normand. 2008. Camera Control in Computer Graphics. *Computer Graphics Forum* 27, 8 (Dec. 2008), 2197–2218. https://doi.org/10.1145/1665817.1665820

[10] Nicholas Davis, Alexander Zook, Brian O'Neill, Brandon Headrick, Mark Riedl, Ashton Grosz, and Michael Nitsche. 2013. Creativity Support for Novice Digital Filmmaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 651–660. https://doi.org/10.1145/2470654.2470747

[11] DJI. 2020. Ground Station Pro. Retrieved January 6, 2020 from https://www.dji.com/ground-station-pro

[12] Alexander Domahidi and Juan Jerez. 2017. FORCES Pro: code generation for embedded optimization. Retrieved September 4, 2017 from https://www.embotech.com/FORCES-Pro

[13] Steven M. Drucker and David Zeltzer. 1994. Intelligent Camera Control in a Virtual Environment. In *In Proceedings of Graphics Interface '94*. 190–199.

[14] David K. Elson and Mark O. Riedl. 2007. A Lightweight Intelligent Virtual Cinematography System for Machinima Production. In *Proceedings of the Third AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Stanford, California) *(AIIDE'07)*. AAAI Press, 8–13.

[15] Q. Galvane, J. Fleureau, F. L. Tariolle, and P. Guillotel. 2016. Automated Cinematography with Unmanned Aerial Vehicles. In *Proceedings of the Eurographics Workshop on Intelligent Cinematography and Editing* (Lisbon, Portugal) *(WICED '16)*. Eurographics Association, Goslar Germany, Germany, 23–30. https://doi.org/10.2312/wiced.20161097

[16] Quentin Galvane, Christophe Lino, Marc Christie, Julien Fleureau, Fabien Servant, Fran Tariolle, Philippe Guillotel, et al. 2018. Directing cinematographic drones. *ACM Transactions on Graphics (TOG)* 37, 3 (2018), 34.

[17] Christoph Gebhardt, Benjamin Hepp, Tobias Nägeli, Stefan Stevšić, and Otmar Hilliges. 2016. Airways: Optimization-Based Planning of Quadrotor Trajectories According to High-Level User Goals. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (Santa Clara, California, USA) *(CHI '16)*. ACM, New York, NY, USA, 2508–2519. https://doi.org/10.1145/2858036.2858353

[18] Christoph Gebhardt and Otmar Hilliges. 2018. WYFIWYG: Investigating Effective User Support in Aerial Videography. (2018). arXiv:1801.05972

[19] Christoph Gebhardt, Stefan Stevsic, and Otmar Hilliges. 2018. Optimizing for Aesthetically Pleasing Quadrotor Camera Motion. 37, 4, Article 90 (2018), 90:1–90:11 pages.

[20] Michael Gleicher and Andrew Witkin. 1992. Through-the-lens camera control.. In *Siggraph*, Vol. 92. 331–340.

[21] Mirko Gschwindt, Efe Camci, Rogerio Bonatti, Wenshan Wang, Erdal Kayacan, and Sebastian Scherer. 2019. Can a Robot Become a Movie Director? Learning Artistic Principles for Aerial Cinematography. *arXiv preprint arXiv:1904.02579* (2019).

[22] Chong Huang, Fei Gao, Jie Pan, Zhenyu Yang, Weihao Qiu, Peng Chen, Xin Yang, Shaojie Shen, and Kwang-Ting Tim Cheng. 2018. Act: An autonomous drone cinematography system for action scenes. 7039–7046.

[23] Edwin L Hutchins, James D Hollan, and Donald A Norman. 1985. Direct manipulation interfaces. *Human–computer interaction* 1, 4 (1985), 311–338.

[24] Hongda Jiang, Bin Wang, Xi Wang, Marc Christie, and Baoquan Chen. 2020. Example-driven Virtual Cinematography by Learning CameraBehaviors. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* XX, X (8 2020), 14.

[25] Niels Joubert, Dan B Goldman, Floraine Berthouzoz, Mike Roberts, James A Landay, Pat Hanrahan, et al. 2016. Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles. (2016). arXiv:1610.01691

[26] Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. 2015. An Interactive Tool for Designing Quadrotor Camera Shots. *ACM Trans. Graph.* 34, 6, Article 238, 11 pages. https://doi.org/10.1145/2816795.2818106

[27] Matthias Kümmerer. 2016. DeepGaze II. Retrieved September 3, 2020 from https://deepgaze.bethgelab.org/

[28] Matthias Kummerer, Thomas SA Wallis, Leon A Gatys, and Matthias Bethge. 2017. Understanding low-and high-level contributions to fixation prediction. In *Proceedings of the IEEE International Conference on Computer Vision*. 4789–4798.

[29] Tsai-Yen Li and Chung-Chiang Cheng. 2008. *Real-Time Camera Planning for Navigation in Virtual Environments*. Springer Berlin Heidelberg, Berlin, Heidelberg, 118–129. https://doi.org/10.1007/978-3-540-85412-8_11

[30] Christophe Lino and Marc Christie. 2012. Efficient Composition for Virtual Camera Control. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 65–70. https://doi.org/10.1145/1409060.1409068

[31] Christophe Lino and Marc Christie. 2015. Intuitive and Efficient Camera Control with the Toric Space. *ACM Trans. Graph.* 34, 4, Article 82 (July 2015), 12 pages. https://doi.org/10.1145/2766965

[32] Christophe Lino, Marc Christie, Roberto Ranon, and William Bares. 2011. The Director's Lens: An Intelligent Assistant for Virtual Cinematography. In *Proceedings of the 19th ACM International Conference on Multimedia* (Scottsdale, Arizona, USA) *(MM '11)*. ACM, New York, NY, USA, 323–332. https://doi.org/10.1145/2072298.2072341

[33] MathWorks. 2017. Segment point cloud into clusters based on Euclidean distance. Retrieved January 8, 2020 from https://mathworks.com/help/vision/ref/pcsegdist.html

[34] T. Naegeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges. 2017. Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization. *IEEE Robotics and Automation Letters* PP, 99 (2017), 1–1. https://doi.org/10.1109/LRA.2017.2665693

[35] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. 2017. Real-time Planning for Automated Multi-view Drone Cinematography. *ACM Trans. Graph.* 36, 4, Article 132 (July 2017), 10 pages. https://doi.org/10.1145/3072959.3073712
[36] Fabio Poiesi and Andrea Cavallaro. 2015. Distributed vision-based flying cameras to film a moving target. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2453–2459.
[37] Mike Roberts and Pat Hanrahan. 2016. Generating Dynamically Feasible Trajectories for Quadrotor Cameras. *ACM Trans. Graph.* 35, 4, Article 61 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925980
[38] Gauthier Rousseau, Cristina Stoica Maniu, Sihem Tebbani, Mathieu Babel, and Nicolas Martin. 2018. Quadcopter-performed cinematographic flight plans using minimum jerk trajectories and predictive camera control. In *2018 European Control Conference (ECC)*. IEEE, 2897–2903.
[39] Bahareh Sabetghadam, Alfonso Alcántara, Jesús Capitán, Rita Cunha, Aníbal Ollero, and Antonio Pascoal. 2019. Optimal Trajectory Planning for Autonomous Drone Cinematography. In *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 1–7.
[40] Ben Shneiderman. 1997. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*. 33–39.
[41] Angie Taylor. 2011. Composition. In *Design Essentials for the Motion Media Artist*, Angie Taylor (Ed.). Focal Press, Boston, 99 – 142. https://doi.org/10.1016/B978-0-240-81181-9.00010-8
[42] VC Technology. 2019. Litchi Tool. Retrieved January 6, 2020 from https://flylitchi.com/
[43] Ke Xie, Hao Yang, Shengqiu Huang, Dani Lischinski, Marc Christie, Kai Xu, Minglun Gong, Daniel Cohen-Or, and Hui Huang. 2018. Creating and chaining camera moves for quadrotor videography. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 88.
[44] I-Cheng Yeh, Chao-Hung Lin, Hung-Jen Chien, and Tong-Yee Lee. 2011. Efficient camera path planning algorithm for human motion overview. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 239–250. https://doi.org/10.1002/cav.398
[45] YUNEEC. 2018. Data Pilot. Retrieved January 6, 2020 from https://us.yuneec.com/comm-en-datapilot

A NOTATION

For completeness and reproducibility of our method we provide a summary of the notation used in the paper in Table 1.

Symbol	Description
$\mathbf{r}$, $\dot{\mathbf{r}}$, $\ddot{\mathbf{r}}$, $\dddot{\mathbf{r}}$	Quadrotor position, velocity, acceleration and jerk
ψ_q, $\dot{\psi_q}$, $\ddot{\psi_q}$, $\dddot{\psi_q}$	Quad. yaw and angular velocity/acceleration/jerk
ψ_g, $\dot{\psi_g}$, $\ddot{\psi_g}$, $\dddot{\psi_g}$	Gimbal yaw and angular velocity/acceleration/jerk
ϕ_g, $\dot{\phi_g}$, $\ddot{\phi_g}$, $\dddot{\phi_g}$	Gimbal pitch and angular velocity/acceleration/jerk
$\mathbf{x}$, $\mathbf{u}$	Quadrotor states and inputs
A, B	System matrices of quadrotor
g	Gravity
$\mathbf{\Theta}$, v	Progress state and input
C, D	System matrices of progress
T	Trajectory end time
N	Horizon length
θ	Progress parameter
$\mathbf{f}_t(\theta)$	Target reference spline ($\mathbb{R}^5$)
$\mathbf{r}_t(\theta)$	Positional target reference ($\mathbb{R}^3$)
$r_t(\theta)$	Target radius reference
$h_t(\theta)$	Target height reference
$\mathbf{v}_d^v(\theta)$	Reference of desirable camera directions
$\mathbf{p}^v$	Camera target position in view space
c_f	Framing optimization cost term
c_v	Visibility maximization cost term

Table 1: Summary of notation used in the body of the paper

B OPTIMIZATION WEIGHTS

The values for the weights of the objective function we used in the user study and the online survey are listed in Table 2.

Weight (layed on)	Value
w_f (target framing)	1
w_v (visibility maximization)	1
w_p (quadrotor position)	1
w_j (jerk)	0.1
w_T (end-time)	0.1

Table 2: Values for weights used in Equation 16.

C DYNAMICAL MODEL

We use the approximated quadrotor camera model of Gebhardt et al. [17]. This discrete first-order dynamical system is incorporated as equality constraint into our optimization problem:

$$\begin{aligned} \mathbf{x}_{i+1} &= A\mathbf{x}_i + B\mathbf{u}_i + g,\ \mathbf{u}_{min} \leq \mathbf{u}_i \leq \mathbf{u}_{max}, \\ \mathbf{x}_i &= [\mathbf{r}, \psi_q, \psi_g, \phi_g, \dot{\mathbf{r}}, \dot{\psi_q}, \dot{\psi_g}, \dot{\phi_g}, \ddot{\mathbf{r}}, \ddot{\psi_q}, \ddot{\psi_g}, \ddot{\phi_g}, \dddot{\mathbf{r}}, \dddot{\psi_q}, \dddot{\psi_g}, \dddot{\phi_g}]^T, \\ \mathbf{u}_i &= [\mathbf{F}, M_{\psi_q}, M_{\psi_g}, M_{\phi_g}]^T, \end{aligned} \tag{14}$$

where $\mathbf{x}_i \in \mathbb{R}^{24}$ are the quadrotor camera states and $\mathbf{u}_i \in \mathbb{R}^6$ are the inputs to the system at horizon stage i. Furthermore, $\mathbf{r} \in \mathbb{R}^3$ is the position of the quadrotor, ψ_q is the quadrotor's yaw angle and ψ_g and ϕ_g are the yaw and pitch angles of the camera gimbal. The matrix $A \in \mathbb{R}^{24x24}$ propagates the state $\mathbf{x}$ forward, the matrix $B \in \mathbb{R}^{24x6}$ defines the effect of the input $\mathbf{u}$ on the state and the vector $g \in \mathbb{R}^{24}$ that of gravity for one time-step. $\mathbf{F}$ is the the force acting on the quadrotor, M_{ψ_q} is the torque along its z-axis and M_{ψ_g}, M_{ϕ_g} are torques acting on pitch and yaw of the gimbal.

D VARIABLE INFINITE HORIZON CONTOUR FOLLOWING

In this work, we build upon the optimization scheme of Gebhardt et al. [19] that allows to generate globally smooth trajectories by following a time-free reference. In this section, we summarize the parts of their approach that are relevant for our algorithm. To allow the generation of trajectories of arbitrary temporal length, the authors add the trajectory end time T into the state space of the model ($\mathbf{x} = [\mathbf{x}, T]^T \in \mathbb{R}^{25}$ with $\frac{\delta T}{\delta t} = 0$). At each iteration of the solver the discretization step of the dynamical model is adjusted by computing $\Delta t = \frac{T}{N}$ where N is the number of stages of the infinite horizon. Based on the current Δt the forward propagation matrices A and B are recalculated. To optimize the trajectory spatially and temproally a time-free parameterization of the reference is required. Therefore, a chord length parameterized spline is used, where the parameter θ describes progress on the spatial reference path defined as $\mathbf{r}_d(\theta) \in \mathbb{R}^3$. θ is added into the model and its dynamics are formulated with the following linear discrete system equation:

$$\mathbf{\Theta}_{i+1} = C\mathbf{\Theta}_i + Dv_i,\ 0 \leq v_i \leq v_{max}, \tag{15}$$

where $\mathbf{\Theta}_i = [\theta_i, \dot{\theta}_i]$ is the state and v_i is the input of θ at step i and $C \in \mathbb{R}^{2x2}$, $D \in \mathbb{R}^{2x1}$ are the discrete system matrices.

The error from the spline is then minimized using the 3D-space approximation of lag $\hat{\epsilon}_l$ and contour error $\hat{\epsilon}_c$ of [35] summarized in the cost term $c_p(\theta, \mathbf{r}_i)$. Introducing a cost term that minimizes the end time of the trajectory $c_{end}(T)$, implicitly demands θ to reach the end of the trajectory when an initial θ_0 and terminal state θ_N are specified. Smooth quadrotor motion is attained by minimizing the model's jerk and angular jerk $c_j(\dddot{\mathbf{r}}, \dddot{\psi}_q, \dddot{\psi}_g, \dddot{\phi}_g)$ and penalizing θ's input v. The overall cost-term is then given by

$$J_i = w_p c_p(\theta_i, \mathbf{r}_i) + w_j c_j(\dddot{\mathbf{r}}, \dddot{\psi}_q, \dddot{\psi}_g, \dddot{\phi}_g) + w_{end} c_{end}(T) + w_v ||v||^2, \tag{16}$$

where $w_p, w_j, w_{end}, w_v > 0$ are scalar weight parameters.

E COORDINATE FRAMES

Figure 3 illustrates the coordinate frames we use in our algorithm. We denote the world frame I to be a right handed coordinate system with the y-axis pointing up. The camera view space V has the y-axis as the up vector, the x-axis points from the camera into the scene, and the z-axis points to the right. Image space positions are specified in homogeneous coordinates in the camera clip space C, which defines the image space to range from -1 to 1 in x and y. Superscripts r^v or r^c indicate that the vector r is expressed in frame V or C (no superscript means the vector is expressed in I).

F FITTING CYLINDER TO POINT CLOUD

To fit cylinders to a target point cloud, we use a heuristic approach for which we identify the maximal spread of points in the dimensions x,z and y. Based on these distances, center, radius and height of the cylinder are computed. The algorithm is described as follows

$$\mathbf{P} = [\mathbf{x}, \mathbf{y}, \mathbf{z}],\ \mathbf{x} = [x_1 ... x_N],\ \mathbf{y} = [y_1 ... y_N],\ \mathbf{z} = [z_1 ... z_N]$$

$$\mathbf{p}_i, \mathbf{p}_j = \underset{i,j \in N}{arg\,max} \sqrt{(x_i - x_j)^2 + (z_i - z_j)^2}$$

$$r_t = \underset{i,j \in N}{max} \sqrt{(x_i - x_j)^2 + (z_i - z_j)^2}$$

$$h_t = max(\mathbf{y}) - min(\mathbf{y})$$

$$\mathbf{p}_t = \begin{bmatrix} \mathbf{p}_i + \frac{\mathbf{p}_j - \mathbf{p}_i}{2} \\ min(\mathbf{y}) + \frac{height}{2} \end{bmatrix},$$

where $\mathbf{P}$ is the target point cloud, N the number of points and $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ lists of values of the respective dimensions. $\mathbf{p}_t$ is the center, r_t the radius and h_t the height of the fitted cylinder.