Creating and Chaining Camera Moves for Quadrotor Videography

KE XIE, Shenzhen University
HAO YANG, Shenzhen University
SHENGQIU HUANG, Shenzhen University
DANI LISCHINSKI, The Hebrew University of Jerusalem
MARC CHRISTIE, Univ Rennes, INRIA, CNRS
KAI XU, Shenzhen University and National University of Defense Technology
MINGLUN GONG, Memorial University of Newfoundland
DANIEL COHEN-OR, Shenzhen University and Tel Aviv University
HUI HUANG*, Shenzhen University

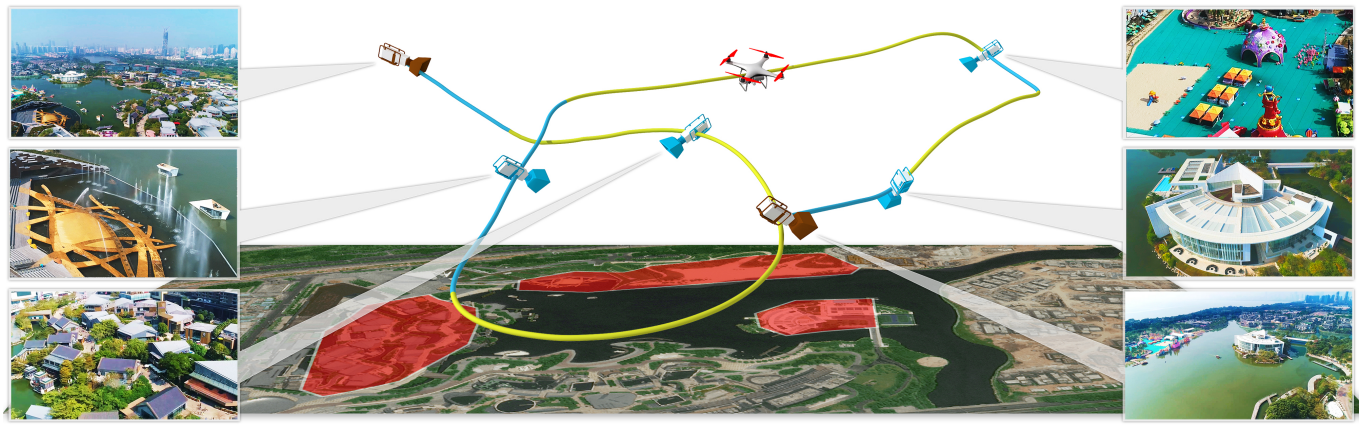


Fig. 1. Given several landmarks (red areas on the satellite image) and a pair of start and end views (brown camera icons), we generate a suitable camera move (in yellow) for capturing each landmark, and optimally connect them into a continuous and smooth path using transition trajectories (in blue).

Capturing aerial videos with a quadrotor-mounted camera is a challenging creative task, as it requires the simultaneous control of the quadrotor's motion and the mounted camera's orientation. Letting the drone follow a pre-planned trajectory is a much more appealing option, and recent research has proposed a number of tools designed to automate the generation of feasible camera motion plans; however, these tools typically require the user to specify and edit the camera path, for example by providing a complete and ordered sequence of key viewpoints.

In this paper, we propose a higher level tool designed to enable even novice users to easily capture compelling aerial videos of large-scale outdoor scenes. Using a coarse 2.5D model of a scene, the user is only expected to specify starting and ending viewpoints and designate a set of landmarks, with or without a particular order. Our system automatically generates a diverse set of candidate local *camera moves* for observing each landmark, which are collision-free, smooth, and adapted to the shape of the landmark. These moves are guided by a landmark-centric view quality field, which combines visual interest and frame composition. An optimal global camera trajectory is then constructed that chains together a sequence of local camera moves, by choosing one move for each landmark and connecting them with suitable transition trajectories. This task is formulated and solved as an instance of the Set Traveling Salesman Problem.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; Interest point and salient region detections; Graphics systems and interfaces;

Additional Key Words and Phrases: Quadrotor videography, trajectory planning, camera control, scene navigation, visual exploration

ACM Reference Format:
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 Trans. Graph.* 37, 4, Article 88 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201286

*Corresponding author: Hui Huang (hhzhiyan@gmail.com)

Authors' addresses: Ke Xie, Shenzhen University; Hao Yang, Shenzhen University; Shengqiu Huang, Shenzhen University; Dani Lischinski, The Hebrew University of Jerusalem; Marc Christie, Univ Rennes, INRIA, CNRS ; Kai Xu, Shenzhen University , National University of Defense Technology; Minglun Gong, Memorial University of Newfoundland; Daniel Cohen-Or, Shenzhen University , Tel Aviv University; Hui Huang, College of Computer Science & Software Engineering, Shenzhen University.

0730-0301/2018/8-ART88 $15.00
https://doi.org/10.1145/3197517.3201286

1 INTRODUCTION

As quadrotor unmanned aerial vehicle (UAV) technology advances and the prices of drones go down, their use is becoming increasingly ubiquitous for a wide variety of tasks. In particular, quadrotors are increasingly used by professional, as well as amateur, photographers and cinematographers for capturing aerial imagery and videos.

However, manual capture of high quality aerial video footage with drones is highly challenging, even for experienced cinematographers. In addition to controlling the trajectory of the drone, one must simultaneously control the camera's degrees of freedom. The common practice requires two persons to capture a drone video, one to pilot the drone, and another to control the camera at the same time [Diaz 2015]. Furthermore, limited battery life and quickly changing lighting conditions make it difficult to rehearse the flight, or repeat the capture, if mistakes were made during the first attempt.

Recently, researchers have addressed problems such as designing smooth drone trajectories that interpolate user-specified keyframes [Joubert et al. 2015], and ensuring that planned trajectories are dynamically feasible [Roberts and Hanrahan 2016]. In this work, we also consider drone trajectory design. We, however, introduce a higher level design tool, intended to enable even novice users to plan a complete and smooth camera trajectory for capturing the desired visual content, and based on fairly minimal user input.

More specifically, the goal of this work is to automate the drone videography process for complex large-scale outdoor scenes consisting of several landmarks of interest. The input to our system is a coarse 2.5D model of a scene, a set of designated landmarks, with or without a particular order in which they should be visited, and the starting and ending camera positions. The model should also include any potential obstacles or areas that must be avoided. From this input, our approach automatically generates a continuous drone trajectory, which is collision-free, visits all of the landmarks, and satisfies several quality criteria. An example of such a trajectory is shown in Fig. 1, and more in Section 6.

Our approach starts by automatically generating a diverse set of candidate local *camera moves* for observing each landmark, which effectively sample the huge search space of all possible trajectories around the landmark. The camera moves are guaranteed to avoid collisions with the landmarks and any of the other specified obstacles. The trajectories are smooth, adapted to the overall shape of the landmark, and are guided by a novel landmark-centric quality view field, which accounts for visual interest and frame composition.

Next, we produce a single continuous trajectory that visits all of the landmarks, by selecting a single camera move for each landmark, and chaining them together using smooth transition trajectories (shown in blue in Fig. 1). These transitions are also designed with collision avoidance, smoothness, and view quality in mind. The task of selecting an optimal ordered sequence of alternating transition and local trajectories is formulated as an instance of the Set Traveling Salesman Problem, a generalized version of the Traveling Salesman Problem, where the set of cities is partitioned into several clusters, and the salesman must follow a minimal cost path that visits exactly one city from each cluster.

We tested our planning tool on five large-scale outdoor scenes. To evaluate the quality, we have also conducted a user study, where for each of four scenes, three different versions of aerial videos produced in three different ways are compared. The analysis drawn from 160 samples (80 participants with 2 repetitions) clearly shows the superiority of our technique over manually created trajectories, as well as trajectories created with a classical GPS waypoint tool.

2 RELATED WORK

2.1 Virtual Camera Control

The control of a camera in a virtual 3D environment is strongly guided by the type of tasks to perform together with the target application, and has been addressed by a wide variety of techniques; see [Christie and Olivier 2009; Christie et al. 2008]. Below, we review a few techniques most closely related to our approach.

The automated computation of single viewpoints has first been addressed by Blinn [1988], who proposed an efficient iterative technique to compute the position and orientation of a camera from the specification of on-screen properties. The problem has then been expressed in a more general framework, where visual properties in the image space (position and orientation of targets) are translated into constraints or costs applied on the degrees of freedom of the camera, and solved through a range of optimization techniques [Bares et al. 2000; Drucker and Zeltzer 1994; Ranon and Urli 2014]. Aspiring to generate more cinematographic viewpoints and trajectories, researchers have been formalizing elements of the filmic language into properties to be enforced on the cameras. He et al. [1996] first presented a set of heuristics to pose virtual cameras based on visual composition principles. The use of different camera representations such as the Toric Space simplifies the expression and solving of such problems [Lino and Christie 2015].

Computing sequences of viewpoints (camera trajectories) imposes new challenges, such as collision and visibility over time, but also smoothness along trajectories. When a priori knowledge of the 3D environment is available, techniques rely on the construction of environment abstractions such as roadmaps, and then compute paths through roadmap traversal techniques and trajectory smoothing; see [Salomon et al. 2003] or [Nieuwenhuisen and Overmars 2004]. Several visibility-aware camera planning techniques have been proposed. Oskam et al. [2009] attempted to enforce visibility of targets along the path, by precomputing visibility between pairs of targets and computing a camera path using an A* planner, which is then smoothed. Lino et al. [2010] perform a real-time potential visibility set computation.

In dynamically changing environments, local motion strategies for camera control have been explored, based on incremental hierarchical solving [Halper et al. 2001], local roadmaps [Li and Cheng 2008], or cinematographic behavior strategies [Galvane et al. 2013].

For visualization of animated volume data, Hsu et al. [2013] proposed a dynamic camera motion planning mechanism that incorporates multiple quality criteria into a single system.

2.2 Scene Navigation and Exploration

Scene navigation approaches address the problem of generating a guided tour of a 3D environment, generally constrained by visiting a set of given landmarks. By contrast, scene exploration provides means for the user to interactively explore 3D environments, while

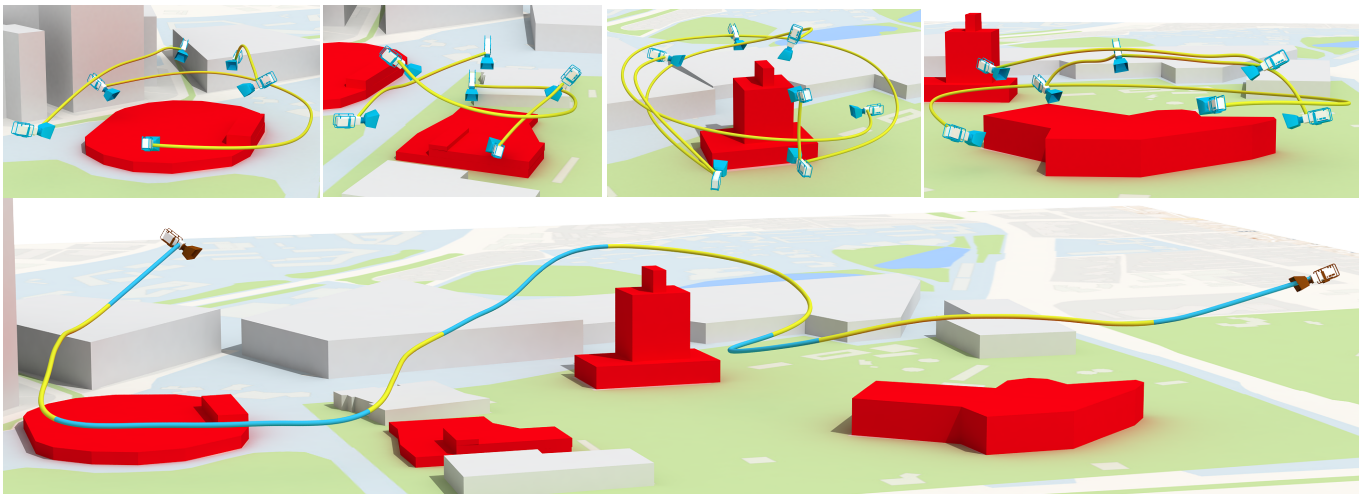


Fig. 2. A 2.5D model of a large-scale scene with landmarks highlighted in red. Top: several candidate moves generated for each of the four landmarks. Bottom: a continuous drone trajectory generated by chaining together the most suitable local trajectories (in yellow) using smooth transition trajectories (in blue).

avoiding non-relevant areas or viewpoints. Scene exploration has largely been addressed in the context of interactive virtual museum tours [Andújar et al. 2004; Chittaro et al. 2003; Drucker and Zeltzer 1994]. Scene navigation and exploration techniques are founded on the *visual interest* of the 3D environment, a characteristic that may be predefined (e.g., a preset of good views of landmarks), or computed automatically through visual interest metrics.

Vázquez et al. [2001] introduced *viewpoint entropy* in an attempt to quantify the amount of information that a viewpoint conveys about a 3D scene, and use this concept to automate the computation of good views. In scene navigation, similar visual metrics can be exploited to automate the computation of camera paths, an approach followed by Sokolov et al. [2008]. Given a set of good viewpoints, the authors generate a path that interpolates the viewpoints by solving a Traveling Salesmen Problem (TSP) where the cities to traverse are viewpoints and the cost is a combination of the Euclidean distance between the viewpoints and the visual quality along the path. The approach has been extended by considering a *semantic distance* metric between the good views, whose goal is to avoid transitions between unrelated landmarks. Serin et al. [2012] applied a similar approach for the specific task of navigation on a 3D terrain. TSPs have also been used in scene exploration to design interactive guided tours [Elmqvist et al. 2007].

While our contribution also relies on the use of TSP techniques to construct a camera path, the problem we address is more complex: first a large collection of suitable camera moves around landmarks is generated, which attempt to satisfy a number of safety and quality requirements, and then a global path is designed, which selects the best camera move for each landmark and connects them together.

2.3 Aerial Cinematography and Trajectory Planning

Mounting cameras on UAVs has triggered the development of techniques to assist users in the complex task of simultaneously controlling the drone and the camera. Applications range from automated surveillance tasks to area coverage [Fan 2014], scanning of unknown environments [Dunkley et al. 2014], capture of aesthetic aerial shots of buildings [Joubert et al. 2015], or human subjects performing an activity outdoors [Joubert et al. 2016]. All approaches compute camera paths that must be physically realizable by the UAV.

Assisting the design of camera trajectories for aesthetic aerial videography has received increasing attention [Gebhardt et al. 2016; Joubert et al. 2015; Roberts and Hanrahan 2016]. The process consists of prototyping a trajectory in a 3D simulator before executing it automatically in the real environment. The virtual trajectory is designed by creating an ordered collection of manually positioned look-from/look-at viewpoints (keyframes). Joubert et al. [2015] also required the user to specify the timing of the keyframes. A specific C^4 continuous trajectory, represented as a 7th degree piecewise polynomial, is created between the keyframes. This representation was found to produce the smoothest and most reasonably bounded control signals for quadrotors [Joubert et al. 2016, 2015]. 7th degree polynomials are also used in the robotics literature [Mellinger and Kumar 2011; Richter et al. 2016]. The feasibility of this trajectory is then analyzed and reported to the user, so he/she can iteratively alter the keyframe timings. More recent work addresses the feasibility issue in an automated way [Roberts and Hanrahan 2016] using an optimized time-warping of the trajectory.

Similarly to Joubert et al. [2015], Gebhardt et al. [2016] proposed a design tool where a camera path can be drawn and edited in a virtual environment and then optimized to ensure its feasibility. Given the multiple constraints (such as avoiding collisions), the optimization process does not guarantee to respect the user inputs: a trade-off is necessary between user inputs and conflicting constraints.

To account for more cinematographic properties on quadrotors, Fleureau et al. [2016] presented a tool to automatically maintain visual on-screen properties (orientation, composition) on moving targets, and automatically compute transitions between viewpoints with moving targets. The approach relies on the Toric Space representation [Lino and Christie 2015] to express cinematographic

properties (distance to target, angle on target, screen positions of targets) and perform interpolations in the Toric Space to maintain or transition between visual properties. Galvane et al. [2016] introduce an interactive tool that allows a user to produce well composed shots of moving actors in real time, by only specifying high level cinematographic commands; however, this tool assumes a fully captured indoor environment. Nägeli et al. [2017] described a high-level drone trajectory planning tool aimed at dynamic and cluttered environments. We target a different scenario of large-scale outdoor scenes containing static landmarks of interest.

While our approach shares some ideas with [Gebhardt et al. 2016; Joubert et al. 2015], we propose a higher level and more automated design tool. Our tool requires users to provide neither viewpoints (except starting and ending), nor timings. Only the set of landmarks of interest needs to be specified, and our tool automatically proposes and selects among camera moves typical of aerial video sequences. This makes our tool particularly well suited for novice users.

Andersson et al. [2017] proposed an active learning approach for quadrotor collision avoidance in the presence of non-cooperative moving obstacles. In this work, we assume that obstacles are static and compute a collision-free optimal trajectory offline.

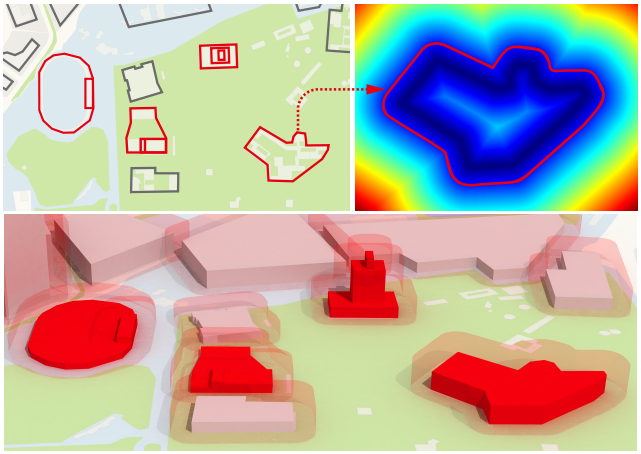


Fig. 3. No-fly zone generation. Top left: based on the 2D contours of landmarks (red) and obstacles (gray) provided as input, we dilate each contour curve (top right) to account for map or GPS inaccuracies. Each dilated contour is extruded based on the height of the corresponding structure to form the no-fly zones (transparent red in bottom figure).

3 OVERVIEW

The goal of our tool is to enable even novice users to easily capture smooth and visually compelling aerial videos covering multiple landmarks in a complex large-scale outdoor environment. We assume that a rough 2.5D or 3D scene model that includes all the landmarks of interest is available. This model should also include any potential obstacles that must be avoided. Fig. 2 shows an overview of our method, which is comprised of two main phases: creating local (landmark-centric) camera moves (Section 4), and chaining them together into a single continuous trajectory (Section 5).

In the first phase, we compute, for each landmark, a set of camera trajectory candidates. The guiding principles for creating such local camera moves are driven by requirements identified by the media production industry [Messina et al. 2017] and include: i) avoiding any collisions with the landmark or any other obstacles in the scene; ii) adapting to the overall shape of the landmark; iii) maximizing the visual interest of video frames captured along the trajectory; and iv) maximizing smoothness and minimizing rotations along the trajectory. To meet these goals, we construct a volumetric landmark-centric *view quality field* in the obstacle-free space around the target landmark (Section 4.1). Guided by this field, a number of possible local shape-aware camera moves are generated that maximize the view quality and trajectory smoothness (Section 4.2).

In the second phase, our goal is to select a single local camera move for each landmark and generate a continuous and smooth trajectory that chains together the selected local moves. To this end, we compute a set of possible *transition trajectories* that connect together endpoints of local moves from different landmarks (Section 5.1). Each local move and each transition trajectory has an associated cost, with smaller cost corresponding to higher quality. The task of selecting a maximal quality ordered sequence of alternating transition and local trajectories is formulated as an instance of the Set Traveling Salesman Problem (STSP), where the local moves are nodes in the same set, while the transitions are edges connecting nodes from different sets (Section 5.2). A near-optimal solution for the STSP is then obtained using an existing algorithm [Helsgaun 2015]. Examples of the resulting global drone camera trajectories are shown in Figs. 1 and 2.

4 CREATING LOCAL CAMERA MOVES

In this section, we describe the construction of local camera moves around a given landmark. Our challenge here is to produce safe (collision-free) trajectories, which could be considered good from a cinematographic standpoint, based only on a coarse model of the scene. Our key ideas consist of: (i) assessing the visual interest of a given view on a landmark by computing a saliency field on the landmark surface, which may then be rendered from any viewpoint; (ii) using a weight map based on well known photographic composition principles to weigh the visual interest map for a given view, yielding a landmark-centric view quality field; (iii) reducing the search space of possible trajectories around each landmark to a manageable size; and (iv) proposing a strategy for generating a small number of diverse promising camera move candidates.

4.1 Landmark-Centric View Quality Field

No-fly zone. When planning drone trajectories, safety is our first priority. To avoid the drone colliding into landmark structures or other potential obstacles under inaccurate GPS readings, we generate a *no-fly zone* for each indicated structure in the scene. Under the assumption that each such structure a is represented using its 2D footprint polygon P_a and its height h_a, the no-fly zone is generated by first computing the Minkowski sum in 2D between P_a and a disk with a radius of r_s (20 meters by default). The resulting dilated 2D contour is then extruded into a 3D volume with a height of $h_a + r_s$. The drone trajectory is prohibited from entering any of the no-fly

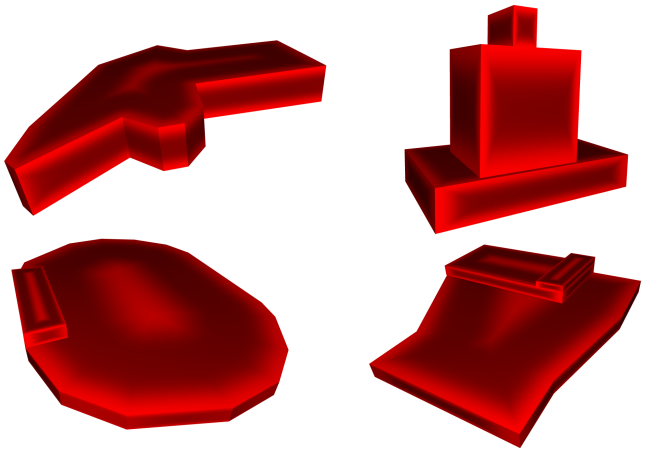


Fig. 4. Visualization of saliency features of the landmarks from Fig. 3. Darker shades indicate lower saliency.

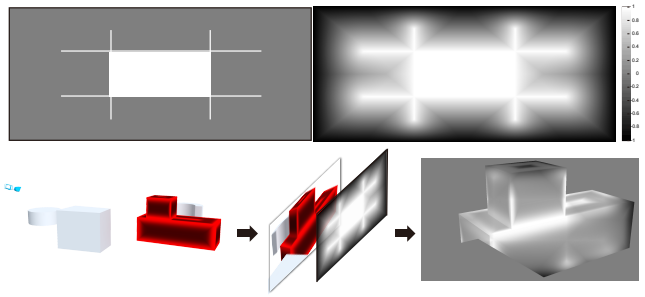


Fig. 5. Estimating the view quality of a given frame based on a rough model of the landmark. Top: a weight map is generated that assigns higher weights based on proximity to the center and the Rule of Thirds lines. Bottom: the saliency scores assigned to the landmark's 3D surfaces (shown in red) are projected to the camera view and multiplied by the weight map.

zones in the scene (Fig. 3), while the volume outside of these no-fly zones is considered safe for flying.

Landmark viewing space. When generating a local trajectory that focuses on a given landmark m, it does not make sense to place the drone too far away from m. Hence, a maximum viewing distance is computed based on the dimensions of the landmark m and the camera's field of view, such that the landmark occupies no less than 2/3 of the frame, when viewed from the maximum viewing distance. The volume within the maximum viewing distance while excluding all no-fly zones is referred as the landmark viewing space $S(m)$.

To find a variety of good viewpoints within the viewing space, we discretize $S(m)$ into cubic cells (voxels) and evaluate the view quality at each voxel center. To simplify the task, we assign a default viewing direction for each voxel center v. That is, so long as v is higher than the geometric centroid of landmark m (referred to as C_m), the view direction is set to point at C_m. Otherwise, the view direction is set to point horizontally towards the ground plane projection of C_m, since most drone cameras cannot be tilted upward.

Salient features. Based on the rough model for landmark m, we now predict the quality of the view captured from the center of each voxel v under the default view direction. This yields a *view quality field*, denoted as Q_m, which fills the viewing space $S(m)$. For each voxel center $v \in S(m)$, the view quality $Q_m(v)$ measures the landmark's shape-aware visual interest based on its visually important features (*salient features*). That is, we favor those views from which more salient features of the landmark's shape are visible. The salient features include the landmark's *silhouettes*, *sharp edges* of the landmark shape's surface, as well as the medial region of the shape's top surface(s), which we refer to as the *skeletal region*. The reason for including the latter is that for large area landmarks, such as a stadium or a harbor, observing the landmark from around its perimeter may not be as effective as flying above the landmark's area, which affords better views of the central region of the landmark.

The silhouette can be directly extracted from the landmark's 2D footprint and sharp vertical edges can be detected based on the angle between the footprint edges, or the dihedral angle between faces, if the landmark's 3D mesh is given. The skeletal region captures the medial region of the top area (flat field or rooftop) of a landmark. One could simply compute the medial axis of the top face polygon. However, due to the noise-prone nature of medial axis, we instead opt to erode the top polygon(s). The resulting eroded region approximates the skeleton of the original polygon.

The silhouette edges, sharp edges, and skeletal regions are assigned with a saliency value of 1, inducing a saliency field across the landmark's shape via a Gaussian decay mapping; see e.g., Fig. 4.

View quality. To compute the view quality at a given view point v, we render each landmark's shape using the intrinsic and extrinsic parameters of the drone camera, with each point colored according to its saliency value, as shown in Fig. 4. This results in a visual interest map, $I_m(v)$, for each view point v.

To achieve an aesthetically pleasing composition, we encourage interesting content to appear in the central region of the frame or to follow the well-known Rule of Thirds in photography, which suggests dividing an image into nine parts by equally spaced horizontal and vertical gridlines, and placing important elements along these lines or their intersections. To this end, we define a composition weight map, I^{ω}, representing the importance of the central region and the vertical and horizontal gridlines, as shown in Fig. 5 (top). The final view quality score is then computed as the dot product of the visual interest and the composition weight maps:

$$Q_m(v) = I_m(v) \cdot I^{\omega}, \tag{1}$$

e.g., as the sum of the values in the bottom right image in Fig. 5.

4.2 Field-Guided Local Camera Moves

Guided by the landmark-centric view quality field, our goal now is to generate a few promising local camera moves passing through the viewing space $S(m)$. Since it is infeasible to enumerate and rank all of the possible trajectories through $S(m)$, we apply a number of *a priori* constraints from the literature [Messina et al. 2017] to define a more manageable search space. First, given that the camera moves

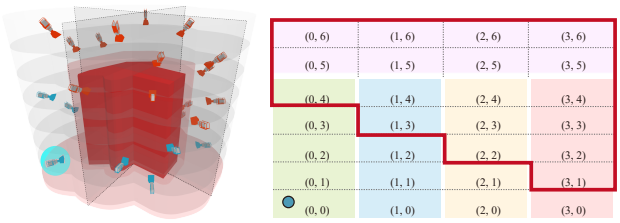


Fig. 6. (a) The viewing space is partitioned into multiple stacks of pie-shaped cells defined using cylindrical coordinates with origin at the centroid of the landmark footprint. A single candidate key view is selected for each cell and shown with a camera icon. (b) A 2D table arrangement of the view cells with rows corresponding to layers and columns to radial sectors. Different colors indicate the grouping of cells into five regions. A candidate local trajectory that starts from the bottom left cell must traverse at least four cells before reaching its ending point. Thus, it must end in one of the cells surrounded by the red frame, also illustrated with red camera icons in (a).

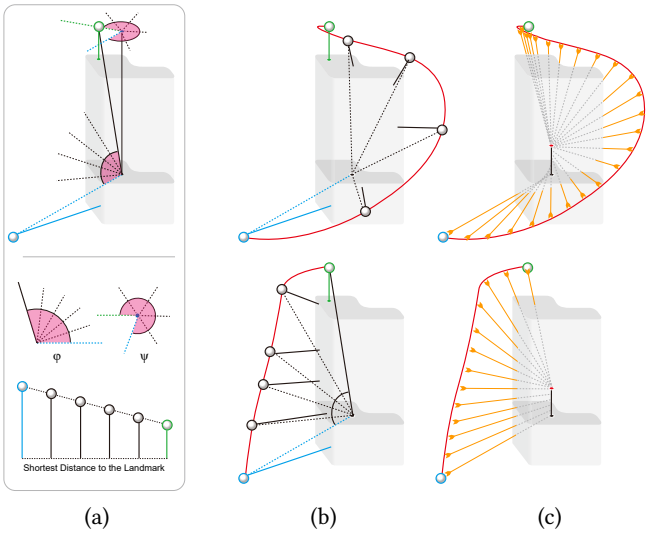


Fig. 7. Trajectory interpolation. (a) Given a pair of starting (in blue) and ending (in green) camera poses, we compute four intermediate poses in order to fit a 5th degree B-spline curve. The intermediate poses are defined by linearly interpolating the tilt angle ϕ to the landmark centroid, the heading ψ to the landmark centroid, and the shortest distance to the landmark's no-fly zone surface. Since the heading may be interpolated in two different ways (clockwise and counter-clockwise), two possible outcomes of camera pose (b) and view direction (c) interpolation are shown.

are intended to explore the landmark, we require that each trajectory should be sufficiently long and feature a significant change either in the altitude or the angle from which the landmark is viewed. Second, view quality along the trajectory should be high. Third, the trajectories should adapt to the shape of the landmark, while being smooth and avoiding sharp or sudden turns. Finally, a diverse set of moves satisfying the constraints above should be considered.

Viewing regions, cells, and key views. Taking the above considerations into account, we divide the viewing space $S(m)$ into several *viewing regions* defined by radial sectors around the landmark, and an additional region above the landmark. Each of these regions is further split into vertically stacked *view cells*, and a single locally best *key view* is selected inside each view cell; see Fig. 6(a).

To ensure that the landmark is observed from a substantial range of views, we only consider smooth trajectories between pairs of key views, which traverse no less than a prescribed minimal number of view cells along the way. For each trajectory in this reduced set, we compute a cost which accounts for the view quality along the trajectory, compatibility with the landmark's shape, and the amount of camera view changes. Finally, we classify all the trajectories into five categories, based on how many regions they traverse. Trajectories traversing a small number of regions are mostly vertical, thereby covering a variety of viewing altitudes, while those traversing more regions are more horizontal, covering a wider variety of radial views. From each category we select the trajectory with the highest score as a candidate local camera move to be considered by the trajectory chaining stage (Section 5).

More specifically, we divide $S(m)$ into $\mathsf{S}+1$ viewing regions: S radial sectors around the landmark, and one region above it. The radial regions are further divided into L vertically stacked slices, forming L view cells per radial view region. The vertical stacking starts from the lowest flying altitude of the drone, $h_{\min} = r_s$ by default, and the height of each layer is set by default to $\max\{h_{\min}, (h_m + r_s)/\mathsf{L}\}$, where h_m is the height of the landmark m. Two more layers of the same height are then added onto the top above the landmark, forming the top view region, which is divided into $2\mathsf{S}$ view cells, as shown in Fig. 6. In practice, we found that $\mathsf{S} = 4$ radial sectors split into $\mathsf{L} = 5$ slices suffice for generating a sufficiently diverse set of moves. Thus, we have at most 28 view cells (up to 7 layers times 4 sectors) arranged in 5 viewing regions around each landmark. This grouping is visualized using different colors in Fig. 6(b).

Within each view cell, we select a single locally best key view, while avoiding choosing two key views which are too close to each other. This is achieved using a greedy algorithm, which first picks the viewpoint with the highest quality score across all cells. Then all candidate viewpoints within r_s radius of selected viewpoints are eliminated, before the next best viewpoint is selected for another cell. Fig. 6(a) shows the candidate key views selected within surrounding viewing space of a given landmark.

Trajectory generation. Based on the key views computed in the previous stage, we generate candidate camera moves around the landmark. Considering each key view as a starting point, we generate all of the trajectories that traverse at least four cells before reaching an ending key view. Thus, we avoid considering short trajectories, and all of the considered trajectories traverse a significant range of altitudes, or angles around the landmark, or both. Fig. 6(b) shows all the possible ending cells for a trajectory starting at the bottom left cell (indicated with a blue dot).

Given a pair of starting and ending camera key views, we generate a smooth trajectory represented by a 5th degree B-spline curve, which goes from the starting pose to the ending pose, through 4 intermediate poses. While previous works [Joubert et al. 2016, 2015] advocated the use of 7th degree splines in order to generate smooth

control signals for quadrotors, we chose to use 5th degree splines, since we use the manufacturer's waypoint SDK to control the drone, rather than generating the control signals directly ourselves. This is explained in more detail in Section 6.

Each camera pose is specified using the tilt angle ϕ and the heading angle ψ with respect to the landmark's centroid, as well as the distance of the camera from the landmark's no-fly zone. The angles and the distances of the intermediate poses are obtained by linearly interpolating those of the starting and ending camera poses (Fig. 7(a)), and a 5th degree B-spline curve is fit to the resulting sequence of six camera poses. The camera's view direction is also linearly interpolated. Since the heading may be interpolated in two different ways (clockwise and counter-clockwise), each pair of starting and ending poses gives rise to two possible camera trajectories, as shown in Figs. 7(b-c).

Each camera trajectory generated as described above is adapted to the shape of the landmark, since the intermediate poses are obtained by interpolating the distances to the landmark's no-fly zone. Nevertheless, this alone cannot guarantee that the trajectory does not penetrate the no-fly zone of the landmark or those of other obstacles in the scene. Thus, for each generated trajectory, we verify that it is indeed collision-free, and discard it otherwise. Only collision-free trajectories are considered from this point onward.

Trajectory cost. Given a trajectory $T_{s,e}$ from viewpoint s to e, we define its associated cost $E_{\text{local}}(T_{s,e})$ as a sum of three terms:

$$E_{\text{local}}(T_{s,e}) = E_{\text{quality}} + E_{\text{axis}} + E_{\text{rot}} \tag{2}$$

The first term E_{quality} defines the cost based on the average quality of views sampled along $T_{s,e}$:

$$E_{\text{quality}} = 1 - \frac{1}{|\mathcal{V}_{s,e}|} \sum_{v \in \mathcal{V}_{s,e}} Q_m(v), \tag{3}$$

where $|\mathcal{V}_{s,e}|$ is the cardinality of the set of sample views along trajectory $T_{s,e}$ and $Q_m(v)$ is the view quality field value at v. We have also considered computing the cumulative view quality along the trajectory, rather than using the average view quality, however, we found that cumulative quality tended to result in somewhat more boring and redundant trajectories.

The second term measures how well the trajectory is aligned with the dominant axis of the landmark:

$$E_{\text{axis}} = 1 - \frac{|(p_s - p_e) \cdot D_m|}{\max(\|p_s - p_e\|^2, \|D_m\|^2)}, \tag{4}$$

where p_s and p_e are the trajectory's starting and ending positions, and D_m is the dominant principle direction of landmark m.

The third term E_{rot} penalizes the rate of change in the camera view direction, as it travels from s to e along the trajectory $T_{s,e}$:

$$E_{\text{rot}} = \frac{1}{\gamma(T_{s,e})}(1 - (q_s \cdot q_e)), \tag{5}$$

where q_s and q_e are unit view direction vectors at s and t, respectively, and $\gamma(T_{s,e})$ denotes the arc length of the trajectory.

We classify all trajectories into at most S + 1 categories, based on the number of viewing regions they crossed. Then for each category, we pick the trajectory with the smallest score in Eq. (2) as a local camera move candidate. In other words, for each landmark, the final output of this stage is the top $K \leq S + 1$ camera moves that sample a diverse set of trajectories through the landmark's safe viewing space. As mentioned earlier, in practice $K \leq 5$; see e.g., Fig. 2(top).

5 CHAINING CAMERA MOVES

Having established a set of top-K candidate local camera moves $\mathcal{T}^m = \{T_j^m\}_{j=1}^K$ for each landmark m, our next task is to compute a set of *transition trajectories* connecting pairs of local trajectories of two given landmarks. An optimal trajectory for the entire flyby is then generated by chaining together selected local and transition trajectories in alternating order, so as to maximize the total quality (i.e., minimize the cost) of the resulting trajectory.

5.1 Constructing Transition Trajectories

A good transition trajectory $T_{jj'}^{mm'}$ that connects two local camera moves, T_j^m for the source landmark m and $T_{j'}^{m'}$ for the destination landmark m', should satisfy several requirements. It should be collision-free and efficient (i.e., short), saving time and battery power. To better convey the overall structure of the scene, we prefer to have either the source or the target landmark to be visible during most of the transition. Finally, as with local camera moves, we would like the camera motion to be smooth, avoiding fast camera rotations and unnecessary turning. Below we describe how we formalize these requirements.

Safety and efficiency. As with local camera moves, we require each transition trajectory to be collision-free. To this end, we construct a visibility graph between the starting and ending points of the transition and the no-fly zone surfaces of the landmarks and the obstacles in the scene. Over this graph, we compute the shortest path between the starting and ending points using a classic algorithm proposed in [Alt and Welzl 1988]. The two local moves for landmark m and m' together with this shortest path in-between are smoothed to form one transition trajectory, which is obstacle-avoiding.

Transition trajectory cost. Similarly to local trajectories, each transition trajectory is also assigned an associated cost E_{trans}, defined as a sum of three terms:

$$E_{\text{trans}}(T_{jj'}^{mm'}) = E_{\text{quality}} + E_{\text{rot}} + E_{\text{turn}}. \tag{6}$$

The view quality term E_{quality} differs from the local move case in that it takes into account both the start landmark m, and the destination landmark m'. Specifically, we define for each viewpoint v along the trajectory a joint view quality:

$$Q_{mm'}(v) = w_m Q_m(v) + w_{m'} Q_{m'}(v), \tag{7}$$

where $w_m = e^{-\frac{d^2(v,m)}{\sigma d^2(m,m')}}$ and $w_{m'} = e^{-\frac{d^2(v,m')}{\sigma' d^2(m,m')}}$ are weights used to balance the influence of the two landmarks. Here d is the Euclidean distance between a viewpoint and a landmark's center. We set $\sigma = 0.05$ and $\sigma = 0.3$, thereby biasing the quality score to be influenced more by the destination landmark.

The E_{quality} term of a transition trajectory is then defined as the average of the joint view quality along the trajectory:

$$E_{\text{quality}} = 1 - \frac{1}{|\mathcal{V}_{m,m'}|} \sum_{v \in \mathcal{V}_{m,m'}} Q_{mm'}(v), \tag{8}$$

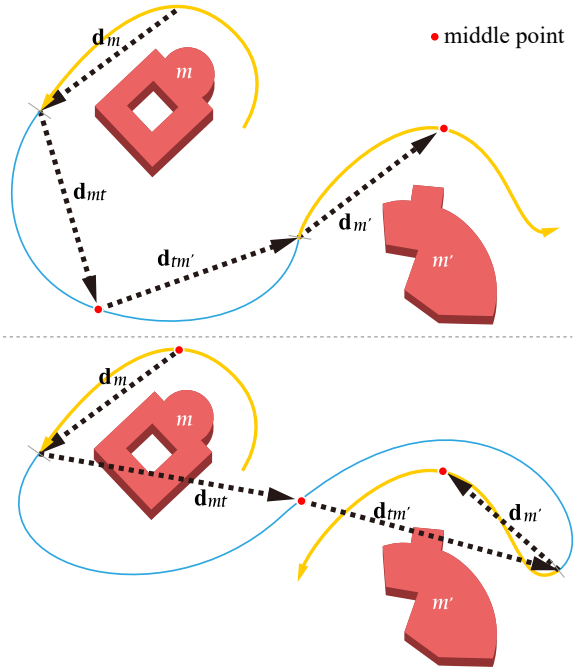


Fig. 8. Estimation of the amount of turning for transition trajectories (blue). The overall trajectory is approximated by a coarse four-segment polyline. The opposing directions in the first and the second pair of line segments in the bottom example indicate a significant amount of turning, corresponding to an undesirable zig-zag trajectory.

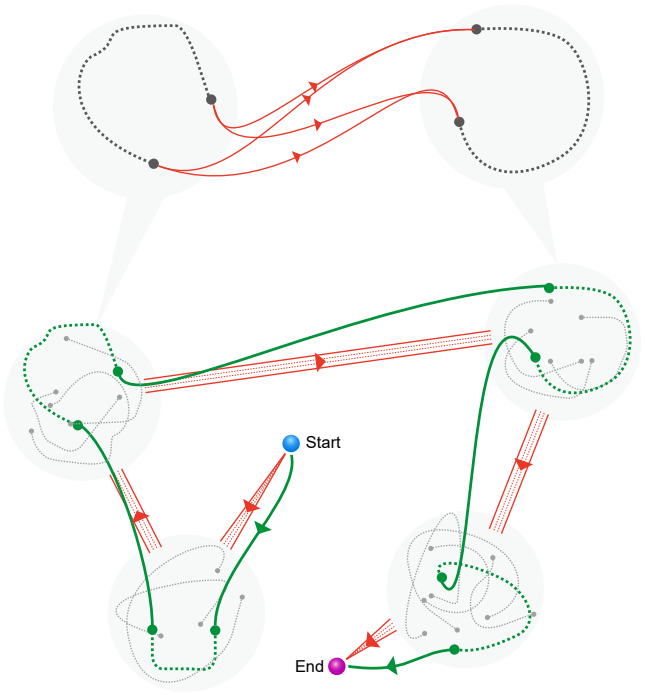


Fig. 9. Top: each pair of local trajectories may be connected using four different transition trajectories. Bottom: The local trajectories for each landmark define a cluster of nodes in the graph. Given the start (big blue dot) and end (big purple dot) views, as well as a landmark visiting order (optional), the final global optimal path shown in green is computed via solving the STSP, which visits exactly one node in each cluster.

where $|\mathcal{V}_{m,m'}|$ is the cardinality of the set of sample views along the trajectory.

The second term is identical to the rotation speed penalty term that we use for local moves in Eq. (5).

The third term, E_{turn}, is introduced in order to discourage zig-zag trajectories. This term penalizes the amount of turning when transitioning from one local trajectory to another. As an extremely simple measure of the amount of turning we examine the angles of a coarse polyline approximation of the transition. Specifically, let $\mathbf{d}_m$, $\mathbf{d}_{mt}$, $\mathbf{d}_{tm'}$ and $\mathbf{d}_{m'}$ denote the vectors of a polyline connecting the midpoints and endpoints of the starting, transition, and ending trajectories, as shown in Fig. 8. The term E_{turn} is then defined as:

$$E_{\text{turn}} = \frac{1}{4}(2 - (\mathbf{d}_m \cdot \mathbf{d}_{mt}) - (\mathbf{d}_{tm'} \cdot \mathbf{d}_{m'})). \tag{9}$$

As illustrated in Fig. 8, this term discourages the first pair of segments $(\mathbf{d}_m, \mathbf{d}_{mt})$ from having opposing directions, and similarly for the second pair $(\mathbf{d}_{tm'}, \mathbf{d}_{m'})$.

5.2 Global Trajectory Planning

At this point we have several local camera move candidates around each landmark, and a collection of transition trajectories connecting different pairs of local moves. Given a starting and an ending camera position, our goal is to produce a single trajectory which includes one local move for each landmark, in an optimal order, with the appropriate transition trajectories between each pair of moves. We solve this difficult combinatorial optimization problem by formulating it as a generalized version of the Traveling Salesman Problem (TSP), also known as the *Set TSP* (STSP).

Given a graph with several disjoint subsets (clusters) of nodes, as well as a set of edges for each pair of adjacent node clusters, the goal of STSP is to find a minimal cost (i.e., maximal quality) tour, which visits each cluster exactly once. The standard TSP may be viewed as a special case of STSP, with only one node per cluster. This implies that the STSP problem is also NP-hard.

In our case, each landmark is to be visited once, and therefore the local moves around a single landmark form a single cluster of nodes in the graph. More specifically, each local camera move contributes to two nodes in the cluster, since its trajectory may be traversed in two directions. The cost of a node is given by $E_{\text{local}}(T_{s,e})$ in Eq. (2), computed along the corresponding local candidate trajectory.

The transition trajectories $T_{jj'}^{mm'}$ that connect the end of a local trajectory T_j^m of landmark m to the beginning of a local trajectory $T_{j'}^{m'}$ of landmark m', correspond to edges connecting nodes from two different clusters, one cluster containing the local moves around landmark m and another containing the local moves around landmark m'; see an illustration in Fig. 9. The cost of an edge is defined by $E_{\text{trans}}(T_{jj'}^{mm'})$ in Eq. (6), evaluated along the corresponding transition trajectory.

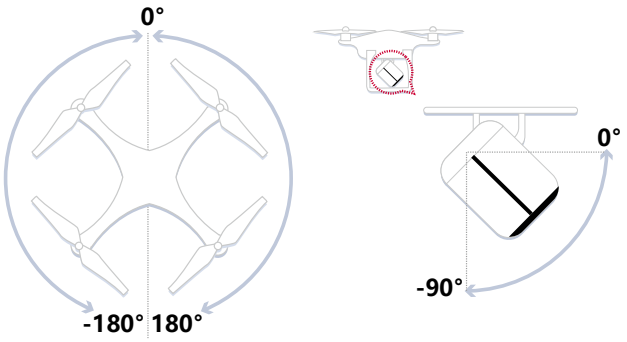


Fig. 10. For a DJI drone camera, the controllable heading range (left) is $[-180°, 180°]$ and the tilt range (right) is $[-90°, 0°]$.

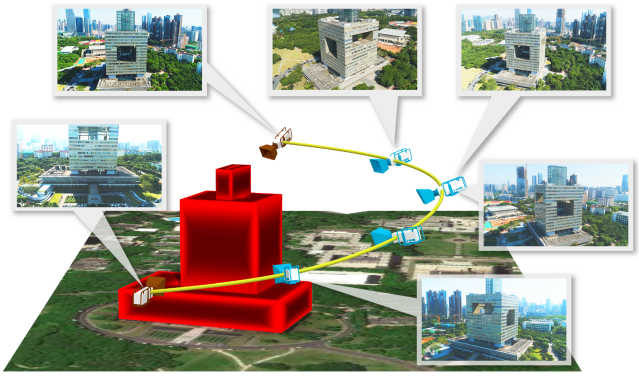


Fig. 11. Given a pair of starting and ending views (brown camera icons) looking at a building, a spiral-like smooth camera move can be easily interpolated using our method illustrated in Fig. 7.

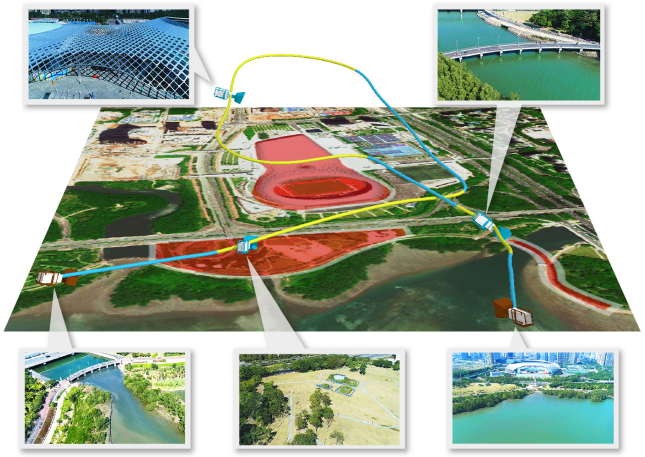


Fig. 12. City Bay: this flyby has three landmarks (highlighted in red) of very different shapes. Please see the supplementary video.

Each transition trajectory corresponds to one directed edge if the visiting order of landmarks is given by the user, otherwise to two directed edges in the graph, each with a different cost (6), since it is non-symmetric. Therefore, the graph contains at most $2KM + 2$ nodes, where each of the M landmarks has $2K$ nodes representing the endpoints of its K local trajectories. Two additional nodes are used to represent the starting and ending point of the drone's flight. The trajectories connecting these points to each of the local moves can be computed and evaluated similarly to the transition trajectories, as described in Section 5.2. The number of edges is capped at $(2K)^2(M-1) + 4K$ when the visiting order is specified by the user, and $(2KM)^2 + 4MK$ when the order is unknown.

In summary, each local move defines a node in the graph, whose cost is given by Eq. (2) and each transition defines an edge between different node clusters, whose cost is given by Eq. (6). A minimal cost path in this graph corresponds to an alternating sequence of transition trajectories and local moves, whose quality is maximal. Such a path is approximately found by solving the STSP problem, using the software package developed by Helsgaun [2015].

6 RESULTS AND EVALUATION

6.1 Drone System

We use the DJI Phantom 4 Pro, a portable yet powerful drone, to capture our aerial videos. The drone's flying movements consist of forward, backward, left or right along a horizontal plane, increasing or decreasing its altitude, and changing its heading clockwise or counterclockwise (Fig. 10(left)). It is equipped with a 4K/60fps 20 megapixel camera, stabilized by a 3-axis (pitch, roll, yaw) mechanical gimbal. The camera tilt, i.e., pitch angle, may be controlled in the range of $[-90°, 0°]$; see Fig. 10(right)).

Our current implementation uses the *DJIWaypointMission* SDK in order to program the drone and the camera to follow the trajectory generated by our method. With this SDK it is possible to specify a sequence of up to 99 waypoints (physical locations to which the drone will fly). The desired camera heading and tilt may be specified for each waypoint. The drone then travels from one waypoint to the next at a preset speed, adjusting altitude, heading, and camera tilt as it advances. Thus, given a trajectory to follow, we place up to 99 samples along the trajectory. The overall speed at which the trajectory is executed is therefore controlled by the DJI software. The locations, camera headings and tilts at these sample points are computed and used to set waypoints. Effectively, this means that the drone follows a roughly piecewise linear approximation of our smooth planned trajectory.

Furthermore, the accuracy and the smoothness of the actual camera motion also depends on the accuracy of the drone's GPS. While our results demonstrate that the resulting aerial videos are reasonably smooth, in the future we plan to implement and use our own lower level drone control program that would be able to follow a planned trajectory in a more precise manner. The tool we designed also proposes a virtual preview of the flight sequence by using the smoothed computed trajectory. The speed of the preview is set to the *autoFlightSpeed* value of the DJI SDK.

6.2 Aerial Videos

Let us start by re-iterating that even a seemingly simple camera move around a single landmark can be difficult to execute when

Fig. 13. Sunny Beach: this flyby has four landmarks (highlighted in red) of very different sizes. Please see the supplementary video.

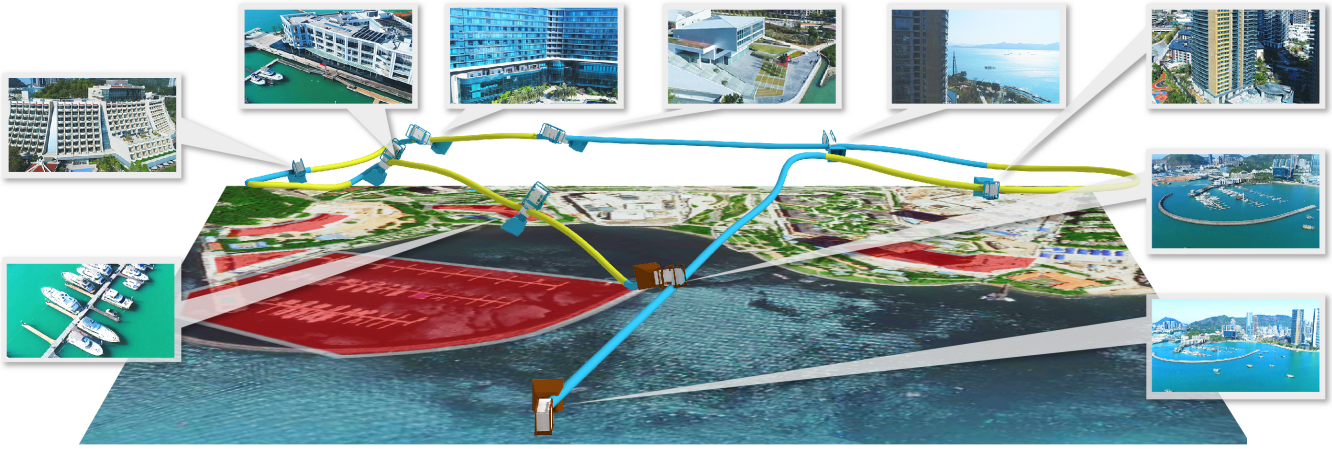


Fig. 14. Sea World: this flyby has five landmarks (highlighted in red), where four of them are very close to each other. Please see the supplementary video.

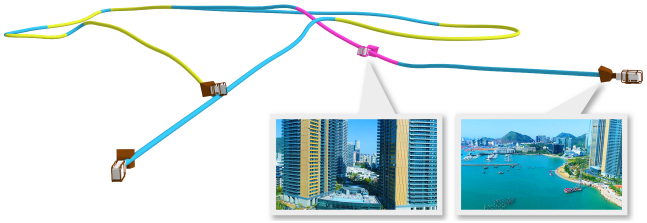


Fig. 15. User can also specify a mandatory key view that he/she wants to include in the flyby, e.g., here indicated by the pink camera icon. He/She can also change the starting and/or ending view accordingly, as shown by the rightmost brown camera icon. This will result in a must-choose local camera move and thus alter the whole global optimal trajectory. Compare with the fully auto one shown in Fig. 14, and the corresponding video has been submitted as supplementary material.

piloting the drone using manual controls. For example, Fig. 11 shows a simple task of viewing around a building. An experienced drone operator was not able to pilot the drone in a smooth and accurate manner along a desirable trajectory, even after several attempts, the best of which is included in the supplementary video. In contrast, our tool successfully produced a smooth spiral-like trajectory by using interpolation in the safe viewing space.

We further tested our trajectory design tool on five large-scale outdoor scenes; see Figs. 1, 2, 12, 13, and 14, respectively. As shown, the test scenes contain landmarks of different shapes and sizes. More detailed statistics about these five tests are summarized in Table 1.

In addition to a fully auto planning, our algorithm can easily incorporate the user's preference. For example, as shown in Fig. 15, if a user indicated a mandatory key view that he/she wants to include in the resulting flyby, we add it into our candidate set of key views and select the best five (by default) local camera moves that pass through it. The global optimal trajectory generated accordingly will then contain the views that users prefer to see.

The resulting captured aerial video sequences are included in the supplementary materials, sped up by a factor of ×4 or ×8. Even with the accelerated playback, the camera motion is quite smooth, and most of the transitions between landmarks appear natural.

It takes our current implementation roughly 5-10 minutes to generate each final global optimal trajectory. As may be seen from

Table 1. Test scene statistics: number of landmarks (#m), the total time for computing view quality fields, local trajectory construction time, global optimization time, and distance of the global optimal trajectory in meters.

Figure	#m	Time$_f$	Time$_l$	Time$_g$	Distance
Fig. 1	3	5m	15s	10s	2475
Fig. 2	4	7m	37s	15s	2179
Fig. 12	3	5m	18s	15s	3190
Fig. 13	4	10m	41s	38s	3806
Fig. 14	5	8m	50s	31s	2998

Table 1, almost all of the computation time is spent on the computation of the view quality fields, a process that can be performed once during the pre-processing and re-used for multiple local and global trajectory construction.

6.3 User Study

User evaluations are conducted to measure the quality of the videos generated by our tool in comparison with i) videos created by an experienced drone pilot, and ii) videos created through the DJI GS Pro design tool[1].

We considered four different locations (Sea World, University, City Bay, Sunny Beach). For each location a number of landmarks were identified; see Table 1. Landmarks were communicated to our experienced drone pilots whose task was to fly the drone to *"highlight the landmarks and to perform good transitions between the landmarks"*. Each drone pilot had a direct visual feedback from the camera, using a tablet connected to the DJI GO APP[2]. For each location, the drone pilot shot multiple takes and proposed his preferred one for the user study. Resulting videos represent the *Manual* condition. Two different but both experienced pilots created the videos, either working alone or together.

This landmark information was also given to our experienced drone pilots who used the DJI GS Pro iPad application to create a virtual trajectory by designing a sequence of waypoints on a 3D map (*tap and waypoint flight* mode) and then flying along them automatically. The resulting videos represent the *DGS* condition. Finally, we used these landmarks to generate videos with our tool, referred to as the *Auto* condition. The videos for the three conditions were shot in different weather and lighting conditions, which may have affected the perception of viewers. To perform such recordings with very similar color and lighting conditions remains challenging. It is important to note that for each question, there were multiple scenes compared, therefore the lighting conditions may have influenced both our method as well as the others.

Our hypotheses are: the videos created by our tool are more pleasing to watch than the others (H1), provide a clearer overview of the landmarks than the others (H2), follow a more reasonable route than the others (H3), provide better transitions between landmarks (H4), and create smoother drone trajectories (H5).

To evaluate these hypotheses, we ran multiple side-by-side video comparisons rather than individual ratings [Yannakakis and Hallam 2011]. Each side-by-side comparison consisted in watching two videos of the same location generated by different flying methods and then answering 5 questions on a 5-point Likert scale. A 2D view of the scene was displayed above the videos with the landmarks highlighted in red. Users had full control over the videos (start, pause, stop and navigate in time). A total of 80 participants were recruited (age varies from under 20 to over 40, with majority fall into the range of 20-30). There is a total of 12 videos (4 locations and 3 methods per location). Each participant watched 6 side-by-side video comparisons:

- two versions of *Auto* vs. *Manual*, one version per location
- two versions of *Auto* vs. *DGS*, one version per location
- two versions of *Manual* vs. *DGS*, one version per location

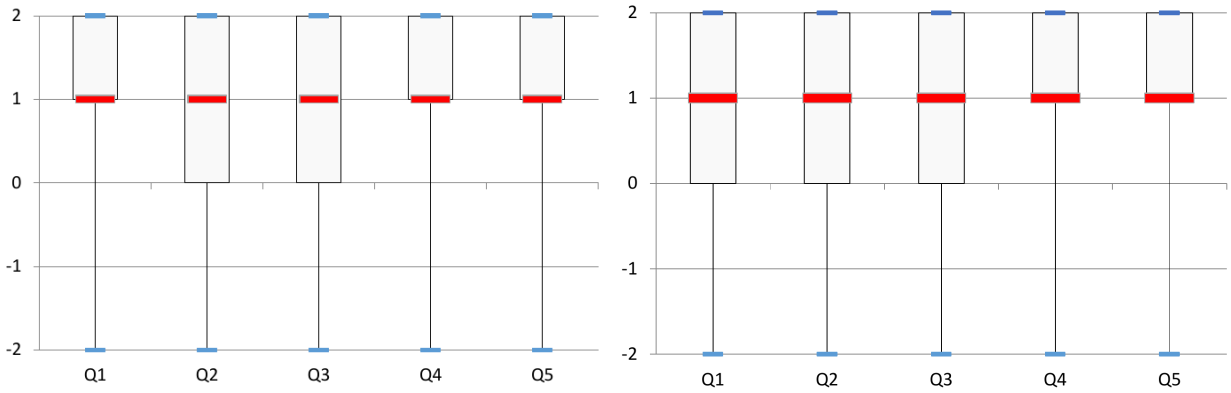
Each video is 1-2mins in time, and the total experiment time for a user was around 25mins. Each user answered a total of 30 questions.

In these experiments, we emit the hypothesis that the results are not influenced by the locations and therefore aggregated the results over different locations. Running the comparisons on two versions ensured repeatability. While we were not interested in the results of *Manual* vs. *DGS*, we ran these comparisons to ensure each condition was viewed exactly 4 times by each user. The left-to-right ordering of videos was determined randomly. Order of comparisons was shuffled between participants. For one given condition, the size of the statistical sample is 160 (80 participants with 2 repetitions).

The questions were: (Q1) the left video was more pleasing to watch than the right video on a 5 point Likert scale ranging from "completely agree" to "completely disagree", displaying "neutral" in the middle; (Q2) the left video provided a clearer overview of the landmarks than the right one; (Q3) the left video follows a more reasonable route than the right one; (Q4) the transitions between the landmarks are more pleasing on the left video and (Q5) the trajectory of the drone was smoother in the left video.

Fig. 16 displays the results of our user evaluation for questions Q1 to Q5, in comparing our *Auto* approach v.s. the *Manual* approach (left chart), and our *Auto* approach vs. the *DGS* approach (right chart). Given that we are performing side by side comparisons, we relied on Wilcoxon signed rank tests. A post-hoc Bonferroni correction was applied with $n = 3$ (the number of pairwise comparisons for the 3 conditions). The applied correction value is therefore $p = 0.013$. On question Q1, the hypothesis that our video is preferred over the *Manual* version is ensured (Wilcoxon signed rank test with $p = 0.0127$). On the same question Q1 when comparing with *DGS*, the null hypothesis could not be rejected (Wilcoxon signed rank test with $p = 0.0281$) however being above the non-corrected threshold value ($p = 0.05$). For H2 (providing a better overview), the null hypothesis can be rejected both when comparing *Auto* v.s. *Manual* ($p = 0.0088$), and *Auto* v.s. *DGS* ($p = 0.0101$). For H3 (following a more reasonable route), the null hypothesis can also be rejected (*Auto* v.s. *Manual* with $p = 0.0071$, *Auto* v.s. *DGS* with $p = 0.0092$). For H4 (providing better transitions), the conclusions are the same (*Auto* v.s. *Manual* with $p = 0.0040$, *Auto* v.s. *DGS* with $p = 0.0031$), as well as for H5 (creating smoother drone trajectories) with *Auto* v.s. *Manual* with $p = 0.0001$, *Auto* v.s. *DGS* with $p = 0.0113$). We also ensured that the location does not influence the results of the users. A Wilcoxon rank sum test shows that the hypothesis of a difference does not hold on all questions ($p = 0.01052$).

[1] https://www.dji.com/ground-station-pro
[2] https://www.dji.com/goapp



(a) *Auto* vs. *Manual* (b) *Auto* vs. *DGS*

Fig. 16. Boxplot visualizations of the user evaluation comparing (a) *Auto* vs. *Manual* videos, and (b) *Auto* vs. *DGS* videos. On the scale, 0 corresponds to a neutral answer, −2 corresponds to completely disagree and +2 corresponds to completely agree on questions Q1 to Q5. Blue dashes represent minimum and maximum values answered, and the red dash represents the median value. The boxes stretch from the first quartile (bottom) to the third quartile (top), meaning that 25% of the values are below the first quartile and 75% are equal or above the third quartile.

In addition to these statistical results, the experienced drone pilots (1.5 years experience each) mentioned during discussions that the design of the drone trajectories in the *Manual* condition was a complex task to perform due to the combination of multiple objectives: avoiding collision, maintaining visibility on the drone, maintaining drone velocity, and framing the landmarks while ensuring a smooth trajectory. Smoothness in the trajectory and camera angle was the most difficult task over such long flights, and many takes were required. In opposition, the DJI GS Pro application simplified the process, however, some iterations between designing the path in the tool and flying the drone were required to adjust camera angles around landmarks. One experienced drone pilot commented on the results to our *Auto* approach by mentioning *"Some generated moves are quite impressive and well respect the shapes of landmarks"*, highlighting the benefits of our solution.

7 CONCLUSION AND FUTURE WORK

We have introduced a new high level drone trajectory design tool, which generates a smooth trajectory for capturing a continuous flyby video of a collection of nearby landmarks. The tool takes as input a rough 2.5D models for the landmarks to be visited and additional obstacles that the drone should avoid.

Our method uses landmark-centric cylindrical coordinates to produce a set of local candidate moves that are smooth, collision-free, and adapted to the shape of each landmark. Global combinatorial optimization based on a generalized TSP solver is then used to produce a single continuous flyby trajectory by chaining together a sequence of selected local camera moves, one for each landmark.

While our results indicate that our method can greatly assist users in capturing aerial videos, there are a number of promising directions for future work. First of all, our current approach follows cinematographic composition guidelines by encouraging visually interesting features appear in the center of the frame or along the one-third lines. However, it does not take into account the lighting conditions and the relative positions between foreground and background objects. To film a landmark under a desired composition or from a desired orientation with respect to the lighting at the time, users could insert mandatory key views, but no effort is made to maintain these lighting or composition conditions along the generated trajectory. Future work should develop metrics for assessing lighting condition and quality of frame composition for intermediate views along the generated trajectories. Such metrics should be taken into account by the optimization process in order to yield aerial videos of higher cinematographic quality.

Another promising direction is to perform a study on the camera moves used by professional aerial videographers, and compile a diverse collection of parameterized interesting camera moves. Armed with an arsenal of such moves, given a specific landmark with a set of potential key-views, as well as other constraints, local trajectory candidates could be generated by optimizing the parameters so as to best fit the shape of the landmark and the constraints. Such an approach would make it even easier for inexperienced users to generate high-quality professional looking aerial videos.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments. This work was supported in part by NSFC (61522213, 61761146002, 6171101466, 61532003, 61622212), China Postdoc Foundation (2017M622780), 973 Program (2015CB352501), Guangdong Science and Technology Program (2015A030312015), Shenzhen Innovation Program (KQJSCX20170727101233642, JCYJ20151015151249564), ISF-NSFC Joint Research Program (2217/15, 2472/17), and NSERC (293127).

APPENDIX

Downloads for our high-resolution aerial videos:

http://vcc.szu.edu.cn/drone1/download/

REFERENCES

Helmut Alt and Emo Welzl. 1988. Visibility graphs and obstacle-avoiding shortest paths. *Mathematical Methods of Operations Research* 32, 3 (1988), 145–164.

Olov Andersson, Mariusz Wzorek, and Patrick Doherty. 2017. Deep Learning Quadcopter Control via Risk-Aware Active Learning. In *Proc. AAAI Conf. on Artificial Intelligence*, Vol. 5. 3812–3818.

Carlos Gran Andújar, Pere Pau Alcocer Vázquez, and Marta González Fairén. 2004. Way-Finder: Guided Tours Through Complex Walkthrough Models. *Computer Graphics Forum (Proc. of Eurographics)* 23, 3 (2004), 499–508.

William H. Bares, Somying Thainimit, and Scott McDermott. 2000. A Model for Constraint-based Camera Planning. In *Smart Graphics AAAI Spring Symposium*. 84–91.

Jim Blinn. 1988. Where am I? What am I looking at? *IEEE Computer Graphics and Applications* 8, 4 (1988), 76–81.

Luca Chittaro, Roberto Ranon, and Lucio Ieronutti. 2003. Guiding Visitors of Web3D Worlds Through Automatically Generated Tours. In *Proc. Conf. on 3D Web Technology*. 27–38.

Marc Christie and Patrick Olivier. 2009. Camera control in computer graphics: models, techniques and applications. In *ACM SIGGRAPH ASIA 2009 Courses*.

Marc Christie, Patrick Olivier, and Jean-Marie Normand. 2008. Camera control in computer graphics. *Computer Graphics Forum* 27, 8 (2008), 2197–2218.

T. J. Diaz. 2015. Lights, drone... action. *IEEE Spectrum* 52, 7 (2015), 36–41.

Steven M Drucker and David Zeltzer. 1994. Intelligent camera control in a virtual environment. In *Proc. of Graphics Interface*. 190–190.

Oliver Dunkley, Jakob Engel, Jürgen Sturm, and Daniel Cremers. 2014. Visual-Inertial Navigation for a Camera-Equipped 25g Nano-Quadrotor. In *Aerial Open Source Robotics Workshop*.

Niklas Elmqvist, M. Eduard Tudoreanu, and Philippas Tsigas. 2007. Tour generation for exploration of 3D virtual environments. In *Proc. ACM symposium on Virtual reality software and technology*. 207–210.

Jiankun Fan. 2014. *Optimal path planning and control of quadrotor unmanned aerial vehicle for area coverage.* Ph.D. Dissertation. The University of Toledo.

Julien Fleureau, Quentin Galvane, Francois-Louis Tariolle, and Philippe Guillotel. 2016. Generic Drone Control Platform for Autonomous Capture of Cinema Scenes. In *Proc. Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*. 35–40.

Quentin Galvane, Marc Christie, Rémi Ronfard, Chen-Kim Lim, and Marie-Paule Cani. 2013. Steering behaviors for autonomous cameras. In *Proc. Motion on Games*. 93–102.

Quentin Galvane, Julien Fleureau, Francois-Louis Tariolle, and Philippe Guillotel. 2016. Automated Cinematography with Unmanned Aerial Vehicles. In *Proc. Eurographics Workshop on Intelligent Cinematography and Editing*. 23–30.

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 *Proc. CHI Conf. on Human Factors in Computing Systems*. 2508–2519.

Nicolas Halper, Ralf Helbing, and Thomas Strothotte. 2001. A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence. *Computer Graphics Forum (Proc. of Eurographics)* 20 (2001), 174–183. Issue 3.

Li-Wei He, Michael F Cohen, and David H Salesin. 1996. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *Proc. of SIGGRAPH*. 217–224.

Keld Helsgaun. 2015. Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun algorithm. *Mathematical Programming Computation* 7, 3 (2015), 269–287.

Wei-Hsien Hsu, Yubo Zhang, and Kwan-Liu Ma. 2013. A Multi-Criteria Approach to Camera Motion Design for Volume Data Animation. *IEEE Trans. Visualization & Computer Graphics* 19, 12 (2013), 2792–2801.

Niels Joubert, L. E. Jane, Dan B. Goldman, Floraine Berthouzoz, Mike Roberts, James A. Landay, and Pat Hanrahan. 2016. Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles. *ArXiv e-prints* (Oct. 2016). arXiv:cs.GR/1610.01691

Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. 2015. An interactive tool for designing quadrotor camera shots. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)* 34, 6 (2015), 238:1–238:11.

Tsai-Yen Li and Chung-Chiang Cheng. 2008. Real-Time Camera Planning for Navigation in Virtual Environments. In *Smart Graphics*. Lecture Notes in Computer Science, Vol. 5166. Springer Berlin Heidelberg.

Christophe Lino and Marc Christie. 2015. Intuitive and efficient camera control with the toric space. *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 34, 4 (2015), 82:1–82:12.

Christophe Lino, Marc Christie, Fabrice Lamarche, Guy Schofield, and Patrick Olivier. 2010. A Real-time Cinematography System for Interactive 3D Environments. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 139–148.

Daniel Mellinger and Vijay Kumar. 2011. Minimum snap trajectory generation and control for quadrotors. In *Proc. IEEE Int. Conf. on Robotics & Automation*. 2520–2525.

A. Messina et al. 2017. *Multidrone media production requirements.* Technical Report. University of Bristol.

Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. 2017. Real-time Planning for Automated Multi-view Drone Cinematography. *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 36, 4 (2017), 132:1–132:10.

Dennis Nieuwenhuisen and Mark H. Overmars. 2004. Motion Planning for Camera Movements. In *Proc. IEEE Int. Conf. on Robotics & Automation*, Vol. 4. 3870–3876.

Thomas Oskam, Robert W Sumner, Nils Thuerey, and Markus Gross. 2009. Visibility transition planning for dynamic camera control. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 55–65.

Roberto Ranon and Tommaso Urli. 2014. Improving the Efficiency of Viewpoint Composition. *IEEE Trans. Visualization & Computer Graphics* 20, 5 (2014), 795–807.

Charles Richter, Adam Bry, and Nicholas Roy. 2016. Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments. *Robotics Research* 114 (2016), 649–666.

Mike Roberts and Pat Hanrahan. 2016. Generating Dynamically Feasible Trajectories for Quadrotor Cameras. *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 35, 4 (2016), 61:1–61:11.

Brian Salomon, Maxim Garber, Ming Lin, and Dinesh Manocha. 2003. Interactive navigation in complex environments using path planning. In *Proc. Sym. on Interactive 3D Graphics*. 41–50.

Ekrem Serin, Serdar Hasan Adali, and Selim Balcisoy. 2012. Automatic path generation for terrain navigation. *Computers & Graphics* 36, 8 (2012), 1013–1024.

Dmitry Sokolov and Dimitri Plemenos. 2008. Virtual world explorations by using topological and semantic knowledge. *The Visual Computer* 24, 3 (2008), 173–185.

Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. 2001. Viewpoint Selection using Viewpoint Entropy. In *Proc. of Vision Modeling and Visualization Conference*, Vol. 1. 273–280.

Georgios N. Yannakakis and John Hallam. 2011. Rating vs. Preference: a comparative study of self-reporting. In *Proc. Int. Conf. on Affective Computing and Intelligent Interaction*. 437–446.