DOI: 10.1111/cgf.15179
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2024
M. Skouras and H. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 8

Generating Flight Summaries Conforming to Cinematographic Principles

Christophe Lino[1,2] Marie-Paule Cani[1]

[1]Ecole Polytechnique, CNRS (LIX), IP Paris, France
[2]Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France

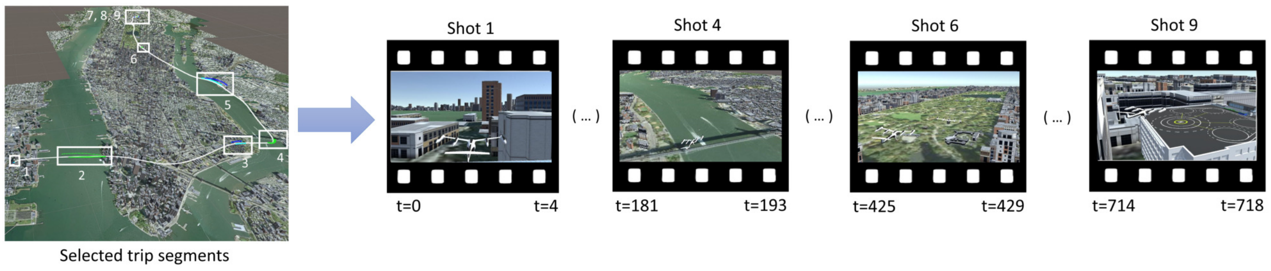


Figure 1: *From an input 3D trajectory, our method automatically generates a video preview of a user-specified duration, including well chosen time ellipses (i.e. gaps in the story). To this end, we jointly tackle two interrelated problems: (i) a best-summary problem, i.e. conveying the relevant parts of a trip within a limited duration and (ii) a virtual camera control problem, i.e. choosing the parameters and trajectories for each camera over time to best convey the trip, while enforcing both camera dynamics and the main cinematographic principles.*

Abstract

We propose an automatic method for generating flight summaries of prescribed duration, given any planed 3D trajectory of a flying object. The challenge is to select relevant time-ellipses, while keeping and adequately framing the most interesting parts of the trajectory, and enforcing cinematographic rules between the selected shots. Our solution optimizes the visual quality of the output video both in terms of camera view and film editing choices, thanks to a new optimization technique, designed to jointly optimize the selection of the interesting parts of a flight, and the camera animation parameters over time. To our best knowledge, this solution is the first one to address camera control, film editing, and trajectory summarizing at once. Ablation studies demonstrate the visual quality of the flights summaries we generate compared to alternative methods.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Procedural animation;*** *Temporal reasoning;* • ***Applied computing*** $\rightarrow$ *Media arts;*

1. Introduction

While many companies are developing new flying vehicles such as VTOL (Vertical Take-off and Landing Vehicles) to transport passengers for short flights above cities, the ability to preview such flights, given an input duration for the video summary, is crucial for advertisement and would help improving future acceptance by the public. Generating such flight previews could also be extremely useful for planning and pre-visualizing drones flights. These two applications motivated this work.

The Computer Graphics (CG) community has shown a growing interest in the development of autonomous, yet expressive virtual cameras, as well as in automating film editing techniques. They typically inspire from cinematographic principles to automatically generate camera shots (*i.e.* the parameters of each camera over time), then edit and combine shots between multiple cameras towards a coherent storyline. However, producing 3D summaries of prescribed duration has for now been restricted to road-trips, where camera control is a 2-dimensional problem, and the summary is achieved by non-linearly accelerating time to focus more on interesting trip parts. This has the disadvantage of giving a distorted perception of changes in vehicle speed and is particularly ill-suited to flights, where frequent accelerations and decelerations of the camera could make the observer ill.

In contrast, this work relies on ***time-ellipses*** to generate flight summaries. This technique, borrowed to cinematography where is is commonly used, is defined as the fact to cut out a part of a story between two specific instants. In our case, time-ellipses allow to exclude the less interesting parts of a flight, but bring two new challenges: (i) the best-summary problem, *i.e.* how to convey the relevant parts of the journey in a limited duration and (ii) a virtual camera control problem, *i.e.* how to choose the parameters and trajectories of each camera over time to best convey the journey. The latter requires enforcing cinematographic principles, such as viewpoints aesthetics and film editing rules across shots.

Our solution relies on jointly solving a discrete and a continuous optimization problem, by alternating their resolution steps. The discrete problem, expressed in terms of geometric and kinematic criteria along the VTOL trajectory, enables to select viewpoints of interest along the trip. The continuous problem enables to optimize the trajectory of one or multiple cameras, used to render the summarized video, by progressively enforcing viewpoint constraints, camera dynamics and continuity-editing rules. We iterate over both steps to obtain a result which satisfies both the summarizing and camera animation generation steps. In summary, we present a fully-automated system to generate automatic video summaries of prescribed duration, by jointly tackling virtual camera control, film editing, and video summarizing. Our contributions are:

- the use of semantic and geometric criteria to evaluate the interest of trip segments and the relevance of a given camera viewpoint along them, leading to a novel interest score function.
- the first formalization of a virtual trip summarizing problem with time ellipses, as a knapsack problem with conflict graph.
- a comprehensive method to optimize the continuous problem of camera control and the discrete problem of film editing at once.

2. Related Work

Our work relates to both *automatic camera control* in virtual cinematography, and to *video summarizing*, *i.e.* the problem of shortening a video to a prescribed duration. For a broader discussion of these problems, we refer the reader to [CON08], [Ron21] or [AAS*12, ETB15]. Note that deep learning solutions are not relevant to solve the whole problem since they would require large flight datasets with manually crafted summaries on which to learn.

Automating camera placement requires to set all parameters of a camera so that the rendered image meets a set of visual criteria. State of the art solutions either rely on constraint-solving or an optimization-based methods. [GW92] formulate the problem of placing a set of 3D points at desired 2D screen locations as an inverse kinematics problem in the 7-dimensional camera space, parameterized by 3D position, 3D orientation and focal length. More recent works rely on more aesthetic criteria inspired by movie production [TB09b], *e.g.* screen position, projected size or viewing angle of the objects on stage, together with their visibility. [LC12, LC15] provide efficient solutions, yet limited to two target objects, by casting computations into a low-dimensional camera space, called the Toric Space. [RU14] rely on a particle swarm optimization in the camera space to provide a more general solution, including a smart initialization and a lazy search strategy.

Computing camera paths amounts to calculating a continuous series of camera viewpoints, while considering additional constraints to ensure the smoothness of camera trajectories and avoid collisions with static or moving objects. There is an abundance of recent works in this domain. Global planning methods pre-compute a roadmap (as a graph) in which a shortest-path can be planned between two camera positions potentially accounting for visibility along the path [OSTG09, Lin13]. Local planning methods update the camera configuration in a reactive manner, while accounting for the camera dynamics, for instance by relying on motion predictive control (MPC) to smoothly move one or multiple cameras while considering visual properties to satisfy on moving targets and collision avoidance [NAMD*17, NMD*17]. Other works directly cast camera paths as spline curves which are then optimized. [JRT*15] manually specify visual compositions through keyframes (*i.e.* raw camera positions and orientations) and the timing between them through easing curves, while [RH16] enables to re-optimize this timing for infeasible paths, while the path in itself remains fixed. [LC15] define keyframes directly in terms of visual properties to satisfy (*e.g.* screen position, size, and view angle of filmed objects). Intermediate viewpoints are interpolated in image space then reconstructed through computations in Toric space. Spline-based techniques were also applied to target following problems. For instance, [GCLR15] use a viewpoint interpolation similar to [LC15] to generate a raw camera path which they fit to a smooth Bezier-curve camera rail, along which they re-optimize the camera position and orientation. More recent techniques also formulate the target following problem (alone) as learning tasks [GCB*19, JWW*20].

None of these works tackled time ellipses. While we reuse spline-based camera trajectories, we optimize them for the first time in terms of film editing rules between consecutive shots.

Continuity-editing between virtual cameras boils down to finding when to switch between cameras and to which new viewpoint to cut to. Achieving this requires to formalize and combine a set of informal film-editing rules [TB09a]. Several attempts were made to automate this process: [LC08] jointly tackle the planning of camera paths and cuts, through a probabilistic roadmap (PRM) locally defined around a tracked target. They implement an important rule, the line of action (which should not be crossed when cutting) as a partition of the PRM into sub-regions. A cut can be performed between viewpoints only if they lie in compatible sub-regions of the PRM. [LCL*10] also formalize stereotypical viewpoints and potential cuts through a spatial partitioning into director volumes. Cuts are defined as a set of geometric and semantic filters, encoding editing rules applied on computed volumes. Galvane *et al.* [GRLC15] take a precomputed set of camera paths as input. They cast the continuity-editing problem as a semi-Markov process, to find which camera to select at each time frame, and solve it by using a dynamic programming algorithm. In summary, existing techniques rely on pre-computed viewpoints and prune unsatisfying cuts. In contrast, we rather optimize viewpoints "until" cuts between them satisfy film-editing rules.

Video summarizing is the automated generation of video previews, synopses, and trailers, to fasten the browsing of large video collections. Solutions involve analyzing visual, audio, or textual contents, the extraction of key frames or video clips, among which a subset is selected to form the targeted lightweight summary. In par-

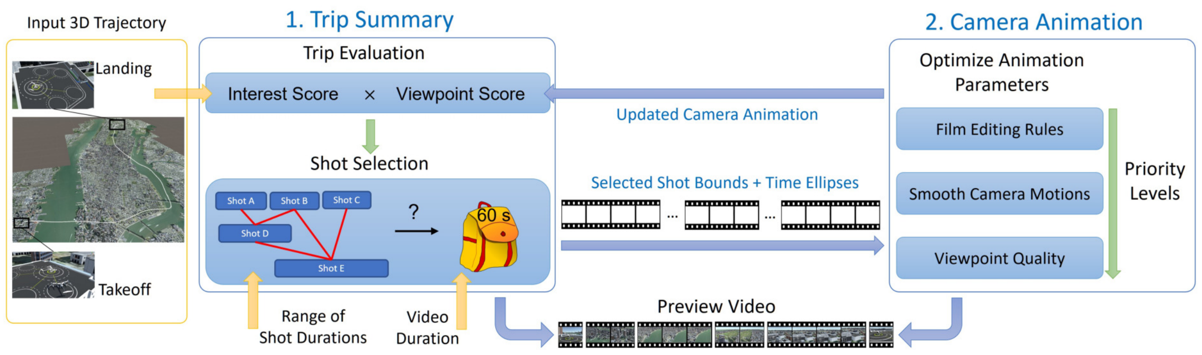


Figure 2: *Processing pipeline for generating a flight summary. From an input 3D vehicle trajectory, a target duration, and a cutting rhythm, we interleave two optimization processes: the first one selects salient trip segments, as a set of shot bounds, where shots are the not-cut time intervals, interlaced with time ellipses. The second one optimizes camera parameters used for each of these shots, accounting for film editing, smoothness of camera motion and viewpoint quality constraints. These two steps are iterated until convergence.*

ticular, Barbieri *et al.* [BDA04] derive a set of constraints based on cinematic rules to ensure storyline continuity and non-redundancy requirements. After segmenting an input video, they associate an importance score to each potential summary defined as a selected subset of segments. The best summary within a specific duration range is then found by solving a costly, combinatorial optimization problem. We share the challenges of defining good criteria for detecting the most informative segments, and for evaluating whether their concatenation provides a good summary under duration constraints. We therefore inspired from their methodology. Some recent techniques also attempted to learn interest and redundancy functions from examples of edited movies, provided side information such as substitles [HHFA18] or even the semantics of original shots [YMCZ17] is also given as input.

All video summarizing techniques share a common drawback: if the input video does not already conform to cinematic conventions, the output will not either. In constrast, we have the full control of camera viewpoints. In this work, we aim at properly capturing the trajectory from raw 3D data, by optimizing all camera parameters to adhere to the main cinematographic conventions, while simultaneously addressing a summarizing problem.

Virtual trip synopsis relate to the automated generation of virtual journey summaries, motivated by the increasing availability of tools to plan journeys in virtual, with renderings of increasing quality or based on real panoramic views. There have been few solutions that have been proposed, and these are not very recent. They target the generation of summaries of car journeys (therefore 2D routes). They apply a temporal speedup factor, to devote more time to interesting parts of a road-trip. [CNO*09] increase the vehicle speed when traversing long straight sections, and decrease it during short sections or turns. They also improve anticipation of turns by rotating the camera to look at a point placed at a fixed distance ahead on the path. Huang *et al.* [HLH*16] is our main concurrent method. They inspire from the speed-dependent zooming idea [IH00] introduced in the context of document browsing. They associate an interest score for buildings near the route, depending on their shape or size. Provided an initial camera path following a vehicle travelling at constant speed, they render images taken from sampled cameras along this path. They then compute a global interest for all parts of the trip, which they use to update the vehicle speed. The speed is inversely proportional to the viewpoint interest score. Viewpoints along the camera path are then updated by using a camera elevation and distance which increases with vehicle's speed. They interleave both steps (updating vehicle speed and camera motion) until convergence. Note that while the prescribed summary duration is guaranteed, the artificial speed assigned to the vehicle may distort the user's perception of the journey.

In summary, existing methods only tackled specific subsets of our problem. They either do not account for the optimization of camera paths (computing shots), for continuity editing (cutting and assembling shots given film editing rules), or for the generation of time-constrained summaries. In this paper, we propose an approach capable of solving all of these sub-problems at once.

3. Overview

We target generating a virtual flight preview, as an aesthetically-pleasing (*i.e.* more natural to watch) video of a prescribed duration. Our primary input is the 3D trajectory of the vehicle of interest, *i.e.* its continuous position and direction from take-off to landing. Secondary inputs are: (i) the scene geometry, *i.e.* a terrain populated with buildings, roads and rivers, and optionally (ii) the trajectories of a set of other vehicles – serving as secondary points of interest during the flight.

Existing trip synopsis techniques might be seen as watching a trip in a fast-forward mode, where the system controls the speed-up factor. In contrast, filmmakers adopt another strategy to convey stories in a concise manner: they omit parts of the story, by introducing time ellipses (*i.e.* gaps in the story) which are left for the viewers to fill. Following this strategy, we cast our virtual trip summarizing problem into two complementary objectives: (i) summarizing the trip, *i.e.* selecting the most interesting segments of the rendered camera animation, such that their total duration does not exceed the prescribed video duration, and (ii) generating an appropriate camera animation, *i.e.* optimizing the position and orientation of a camera following the trip, to best satisfy cinematographic principles (*i.e.* shooting and editing rules) [TB09a, TB09b] within and in-between camera shots. We formulate them as complementary optimization problems, detailed in sections 4 and 5 (notations we use are presented in table 1). We run them in parallel (Figure 2), and combine their results through an interleaved iterative process (the output of one step feeding the other step), until convergence.

$G(x,\sigma)$	Gaussian decay, equals to $e^{-x^2/(2\sigma^2)}$
$E(x,\lambda)$	Exponential decay, equals to $e^{-x/\lambda}$
$\mathbf{v}_p(t)$	Position of main vehicle (protagonist) at time t
$\mathbf{v}_j(t)$	Position of j-th vehicle at time t
$\mathbf{c}(t)$	Camera position
$\hat{\mathbf{v}}$	Normalized vector
$[x]^+$	scalar value of x clamped to $\mathbb{R}^+$, *i.e.* $max(0,x)$
$(\mathbf{u},\mathbf{v})$	angle between two vectors
$(\mathbf{u},\mathbf{v})_\beta$	relaxed angle of β degrees, *i.e.* $[(\mathbf{u},\mathbf{v})-\beta]^+$
$ratio(x,y)$	min-max ratio of x and y, *i.e.* $min(x,y)/max(x,y)$

Table 1: *Notations*

4. Trip Summarization

In this best-summary step, we select which parts of the trip should be included in a video of prescribed duration D. The VTOL trajectory is augmented with a camera animation tracking the vehicle along the trip. This camera animation, initialized as an offset from the vehicle's trajectory (this is detailed in section 5.4), is refined through iterations. The output is a list of camera shots (time intervals, whose total length is D) that best summarize the trip.The selected shots should be as informative and visually interesting as possible, *i.e.* highlight salient trip segments and properly convey them, while fulfilling cinematographic conventions.

We cast this optimal-selection process as a 0-1 knapsack problem (KP) where, given items i of weight w_i and profit p_i, we have to select a subset of items of total weight W (the knapsack capacity) and maximum total profit. The solution is a vector $\mathbf{x}$ computed as

$$\arg\max_{\mathbf{x}} \sum_i x_i.p_i \quad \text{under constraint} \quad \sum_i x_i.w_i \leq W$$

where x_i is a 0-1 variable (1 if the ith item is selected, and 0 if not).

Note that we cannot simply split the trip into small segments of same duration d (*e.g.* 1s), consider items (shots) of any duration d_i proportional to d (*i.e.* $d_i = w_i d$, where w_i is an integer), and set the knapsack (*i.e.* the summary video) to a capacity $W = D/d$, then consider non-selected items as time ellipses. Several problems would remain: (i) finding the right granularity for the segmentation is a problem in itself ; (ii) filmmakers avoid camera shots which are either too short or too long – they also avoid introducing cuts at a regular pace (*i.e.* using shots of same length), which could rapidly become visually disturbing for viewers. This naïve formulation must be augmented to overcome these limitations.

Cutting pace control: we allow the user to define a range of possible shot duration $[d_{min}, d_{max}]$, whose bounds must be multiples of some split duration d. We then generate items so that, all together, they represent every possibility of a shot with a start time $s_i = k.d$ and end time $e_i = s_i + d_i$, where k is a positive integer and $d_i \in [d_{min}, d_{max}]$. This enables to (i) control which moments to convey, and how long to stay in a camera shot (before cutting to a new one), as well as (ii) avoid cutting at a regular pace.

Yet, two overlapping items (shots) are in conflict. Selecting both would result in conveying the same moment twice. To enforce this behavior, we cast our optimization as a *Knapsack Problem with Conflict Graph* (KCG) – where a conflict graph $\mathcal{G}$ encodes time interval intersections – which incorporates a new constraint:

$$\forall (i,j) \in \mathcal{G}, x_i + x_j \leq 1 \qquad (1)$$

KCG is known to be strongly NP-hard. For this reason, we do not target an optimal solution, but rather use an easy-to-implement KCG solver [YKW02], to demonstrate the validity of our formulation. More precisely, we solve our KCG using the following greedy algorithm. We order items in decreasing ratio of profit over weight. In a first step, we progressively insert non-conflicting items.In a second step, we browse each item outside the knapsack to check (i) if we can directly insert it (*i.e.* no conflict exist) or (ii) if swapping it with conflicting items would improve the profit, until convergence.

Takeoff and landing phases should always be conveyed in the video. This constraint alone could be cast into another specialized knapsack, called *Knapsack with Forcing Graph* (KFG). However, (i) it is also strongly NP-hard, (ii) in our case it would be reduced to very few items (the take-off and landing phases), and (iii) we are not aware of any solution for a combination of a KCG and a KFG. For all these reasons, we instead integrate this constraint directly into our KCG solver. Before the first step, we insert the best item containing the landing phase, and do the same for the takeoff. In the second step, an item containing the landing or takeoff may only be swapped with an item conveying the same phase.

Redundancy between shots not limited to direct overlaps. Shots filmed at close intervals might also convey similar information. Further, the longer a shot, the more users have time to explore the images. Hence, the more it might seem redundant with an adjacent shot. We express such redundancy as new conflicts in $\mathcal{G}$, by (virtually) extending the time interval of each shot by an amount proportional to its duration (in our test we used a 30% extension).

With this solution in mind, we now need to define the interest of a summarized trip, *i.e.* assign a profit to each created item. To do so, we build an interest score by combining two important criteria: (i) the *trip interest*, evaluating the amount of information held by a trip segments, and (ii) the *viewpoint interest*, evaluating the visual appeal of the trip segments as filmed by the current camera.

4.1. Trip interests

Selected trip segments should be both informative and salient (*i.e.* non-redundant). In this work, we define their *importance* through a set of geometric criteria, and their *salience* through the amount of temporal variations of these criteria.

4.1.1. Importance score

After testing different geometric criteria, we selected the following subset which seems to capture well the essence of 3D trajectories of flying vehicles. Detailed formulas are given in appendix A. Note that, for other types of 3D trajectories, other criteria could be used without changing the general approach.

Context of the flight, and in particular interest points around the vehicle (*e.g.* tall buildings), give strong visual hints that ease the trajectory understanding (*e.g.* places where it unfolds). For the sake of simplicity, we use a definition similar to [HLH*16] for the importance of buildings located along the trip (a mix between its height, volume and uniqueness), but use geometrical computations rather than renderings. One benefit is the separation of trip segment importance from camera view quality, aiding later camera animation optimization. Using this criterion also eases comparisons with [HLH*16].

Altitude: Conveying when vehicles quickly go up or down is also crucial, as it is a factor that directly impacts the VTOL energy efficiency and noise. In practice, this will not only highlight trip segments close to takeoff and landing, but also segments with fast changes of altitude (*e.g.* to bypass tall buildings or hills). We rely on the trajectory slope to capture how fast the altitude evolves.

Flying direction: Moments when the vehicle sharply changes its flying direction along the trip also provide complementary hints about the shape of the trajectory and travelled aerial corridors.

Velocity: Showing a vehicle travelling at low speed may be boring, since the surrounding does not change much. Conversely, when travelling at higher speed (*e.g.* travelling large distances between critical waypoints), views tend to be more interesting and informative. Note that the changes of velocity (*i.e.* vehicle accelerations) could also be a relevant criterion, as it is another factor impacting energy efficiency and noise.

Traffic density: areas of higher traffic density along the trip provide hints on which aerial corridors have been travelled. We rely on the K-nearest neighbor vehicles to evaluate this density.

4.1.2. Saliency score

For sake of generality, we define a independent salience measure for each criteria. To do so, we rely on a Gaussian-weighted center-surround difference of the importance score $I_f(t)$ for a criteria f.

We first compute its Gaussian-weighted average around time t, *i.e.* within a time horizon of length H:

$$GA_f(t,H) = \frac{\sum\limits_{dt \in [-H,+H]} I_f(t+dt).G(dt,H)}{\sum\limits_{dt \in [-H,+H]} G(dt,H)} \qquad (2)$$

We then define the salience score $S_f(t)$ as the difference with the average computed on a larger surrounding ($2H$), *i.e.* :

$$S_f(t) = \left[GA_f(t,H) - GA_f(t,2H)\right]^2 \qquad (3)$$

which we max-normalize. The higher $S_f(t)$, the less the held information is redundant with what is conveyed before and after. In our tests, we used H equal to 0.6% of the total trip duration.

4.1.3. Interest score

When combining the importance and salience scores together, for each criteria, we favor important trip segments (*i.e.* conveying enough information) which are also salient (*i.e.* not conveying the same information for too long), by blending both scores. We also let users weight criteria, to specify the ones they care more about. The trip interest score is then formulated as:

$$T(t) = \sum_f w_f.\left[\alpha.I_f(t) + (1-\alpha).S_f(t)\right] \qquad (4)$$

In our test, using $\alpha = 0.66$ seemed to provide a good balance between importance and salience.

4.2. Viewpoint quality

Selecting informative trip segments lead to cover the important events. Though, if badly conveyed by the camera, they become less interesting for the viewer. Hence, it is crucial that camera viewpoints associated to selected trip segments properly convey information. Hence, we also compute a viewpoint score, evaluating the visual aesthetic of viewpoints along the trip. We rely on criteria mainly inspired from cinematographic conventions [TB09b].

Two classes of criteria impact the quality of viewpoints. First, some events such as following a vehicle are better conveyed by viewing them under a certain angle and distance [HLH*16]. This preferred view mainly depends on the current vehicle's speed and altitude. The vehicle should also always remain visible. Second, conveying the vehicle's motion, the scene context, and the traffic density, requires to properly arrange important objects (*e.g.* the vehicle, important buildings and other close vehicles) on the screen. We cast these constraints into a weighted mean of the satisfaction $Q_f(t)$ of each criterion f (where all weights w_f sum to 1):

$$Q(t) = \sum_f w_f.Q_f(t) \qquad (5)$$

We rely on four well-accepted visual criteria in the domain (computation formulas are provide in appendix B):

Preferred view: Filming moving vehicles is better conveyed by placing the camera behind and above them, at a distance closely related to its speed and altitude. We use a modified version of Huang *et al.* 's preferred view criteria [HLH*16], where we account for the specificity of flying vehicles (see the supplementary material).

Thirds rule and Motion space: Filmed moving objects should be well-arranged on the screen. Firstly, filmmakers usually divide the screen into three equally-sized areas, horizontally and vertically. Given the four power lines dividing the screen, important objects should be placed at key screen positions, *i.e.* along a line or at an intersection. Secondly, they also leave space in the apparent motion direction. From our study of movies picturing flying vehicles, we also observed that filmmakers usually center vehicles when at rest, and apply the thirds rule and motion space for vehicles in motion. We cast this as a desired screen position making the apparent motion centered on the screen. This is illustrated in Figure 3.

Visibility: The vehicle should remain visible from the camera along the summarized trip. To evaluate its visibility at time t, we simply rely on ray-casts from the camera to randomly sampled points in the bounding box of the vehicle. We then compute its satisfaction as the ratio of rays reaching the vehicle.

Context and traffic inclusion: To support the understanding of the trip, camera viewpoints should convey the surrounding context and other close vehicles in dense traffic areas. We rely on the previously computed K-nearest neighbors (buildings and vehicles), which should be included on the screen.

4.3. Computing the summary

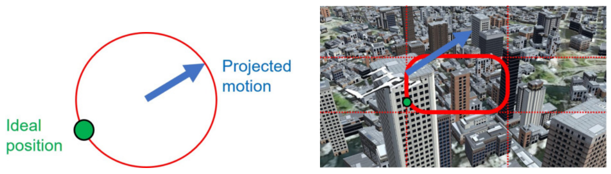
Selected trip segments should be both informative and well conveyed by the camera. If at least one score is very low, the segment should then be discarded. To reflect this behavior, we express the profit of a knapsack item (*i.e.* a trip segment) as a product of the interest and viewpoints scores, accumulated over time:

$$p_i = \int_{t=s_i}^{t=e_i} T(t).Q(t)\ dt \tag{6}$$

After computing the profits and solving the KCG, we are left with a sequence of items (representing time intervals), selected as camera shot candidates. At this point, these selected pieces could be rendered and assembled together into a video. Unfortunately, the quality of viewpoints, or camera motions, might not be optimal. Further, transitions between shots (cuts), should satisfy film editing conventions [TB09a], which are not accounted for in the KCG. Therefore, we might output a video of poor quality. This motivates interleaving trip summary steps with camera animation steps, presented next.

5. Camera Animation

In this camera control step our goal is, given a list of selected shots bounds (*i.e.* time intervals), to improve the quality of the camera animation. We account for three key aspects: (i) camera viewpoints quality should be as good as possible at any time, (ii) camera motions should be as smooth as possible and (iii) transitions (*i.e.* cuts) between consecutive shots should respect continuity-editing rules. We cast this problem as the continuous optimization of camera parameters along time, in a way that we enforce such cinematographic conventions within and in-between camera shots.



(a) *Normalized ideal screen location, regarding projected motion* **(b)** *Application to the screen subdivision into thirds*

Figure 3: *Combination of the rule of thirds and the motion space principles. (a) From the projected motion direction we compute an ideal screen position on a unit circle. (b) We warp this unit-circle to power lines (dotted) splitting the screen into thirds.*

5.1. Camera animation parameters

We define a camera configuration, at time t, as 6 parameters: the 3D position of the camera in world space, and the 3D position of a look-at point. *NB:* both together fully determine its orientation.

To follow the vehicle during the trip, we split our camera animation into a series of 6D spline curves C_i (*i.e.* on all 6 camera parameters). Each corresponds to a distinct time interval, either a selected *shot* or a *time ellipse*. For each interval we regularly sample keyframes (*i.e.* 6D camera configurations) along the curve. The first and last keys match the beginning (s_i) and end (e_i) of the time interval. To accelerate the optimization, we use a higher sampling rate for shots (in our tests, 1 for shots and 0.5 for time ellipses).

These curves are independent from each other. $\mathcal{C}^0$ continuity is not desired between two adjacent shot curves. Conversely, time ellipse curves simply serve as a smooth transition between two shots, hence $\mathcal{C}^0$ continuity is desired. To do so, we consider the first and last key frames of a time ellipse as duplicated keys – they must remain coherent with the last (resp. first) key of the previous (resp. next) shot, during the optimization.

Given this representation, we apply a set of constraints, arising from principles of cinematography and film editing. They are cast into as a set of costs to be minimized, each being defined on one or several keyframes. These costs are detailed in appendix C.

For viewpoints' quality, we designed costs to reflect the same criteria as in section 4.2. Other criteria are discussed below.

5.2. Camera motion smoothness

Providing smooth camera motion is a very important factor in films, as jerkiness in the image may cause discomfort to the viewer. Several requirements should be enforced, which we cast into a set of costs.

Viewing angle Generating nice-looking camera views require that the viewing angle on the vehicle changes as little as possible. We thus penalize such changes.

Camera path and rotation The camera should move smoothly, both in terms of world position and orientation. Along a camera animation curve, we smooth successive camera positions and orientation by penalizing changes in (i) the camera motion direction and velocity, and (ii) the camera rotation and rotational velocity.

Look-at path The camera rotation being indirectly controlled by the look-at point, its path should also remain smooth. Hence, we also penalize changes along the path of the look-at point.

5.3. Film editing conventions

Following film editing guidelines, we also enforce a visual continuity between shots. We express such constraints as costs accounting for the last key of a shot and the first key of the next one.

Jump Cuts: When cutting to a new viewpoint, enough change should be introduced in the vehicle's size (*i.e.* its distance to the camera) or its viewing angle (at least 30°). If not, the transition will be perceived as a fast camera motion, not as a cut.

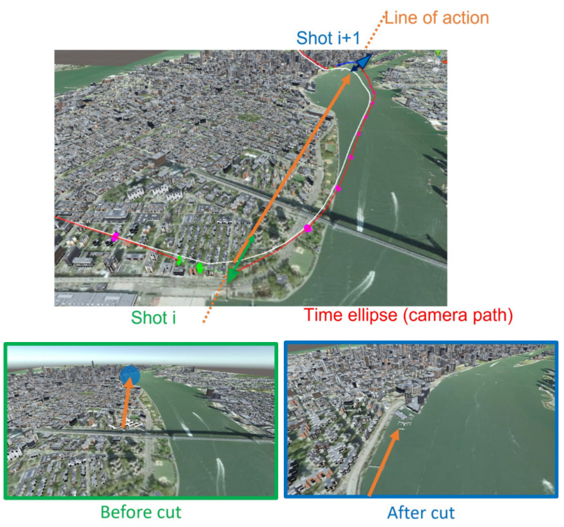


Figure 4: *Illustration of the Line of action and the Look-ahead principles. The apparent direction of motion (orange arrows) is enforced by placing the camera on the same side of a line of action (abstracting this motion) before and after a cut. To show the part of the trajectory cut by the time-ellipse, we also depict, on the "before cut" image, the vehicle location after cut as a blue dot.*

Line of action: The apparent motion should remain consistent across cuts, *i.e.* changes of motion direction in screen-space (to the left *vs.* to the right) should reflect changes in world-space. To this end, we define a plane passing through the vehicle's positions before and after the cut (illustrated in Figure 4), which the camera should not cross.

Screen Position Continuity To avoid jumps in the user's gaze position between shots, abrupt changes in the vehicle's position on the screen should also be avoided.

Look-ahead We experimentally observed that giving a look-ahead on the future position of the vehicle (after the cut) helps in filling the narrative gap created by a time ellipse (*e.g.* offering clues on the intermediate travelled areas). To this end, we constrain the last key of a shot to include this future position in screen-space (this is also illustrated in Figure 4).

5.4. Optimization algorithm

In our initial camera animation, all keyframes are set so that each camera position equals the preferred view and the look-at point is such that the vehicle of interest projects at its ideal screen position (*i.e.* we apply the rule of thirds and motion space). We then optimize key frames through a gradient descent over presented costs.

Not all constraints are equally important. Continuity-editing rules are more important as they will ensure consistency from shot to shot. Camera motions should then be smooth along each animation curve. The preferred view and the rule of thirds are guidelines. Lastly, though conveying the context is desired, this is an even softer constraint. We express such priorities through an inverse kinematics formulation able to solve primary and secondary tasks [BB04]. We split constraints into strict priority levels L_k considering only a subset of costs. The lower k, the greater the priority. Levels are then solved incrementally, by projecting the gradient of a level (*i.e.* the Jacobian J_k) onto the kernel of prior levels.

We further consider several independent gradient descents. We optimize all shots together – they are linked by editing constraints – but each time ellipse alone. We also optimize the camera position and the look-at position in separate steps. After optimizing shots, we copy duplicate keys into time ellipses. Duplicate keys are not optimized for time ellipses. The organization of priority levels and position/look-at optimization is detailed in appendix (Table 4).

(a) *Trip B*

(b) *Trip C*

(c) *Trip D*

Figure 5: *Screenshots from three of our preview videos.*

Name	Trip length (kms)	Trip duration	Trip split (sec)	Shot lengths (sec)
Trip A	23.9	11 min 58 sec	1	[4-12]
Trip B	24	11 min 44 sec	1	[4-12]
Trip C	17.3	6 min 32 sec	1	[3-9]
Trip D	14.9	6 min 58 sec	1	[3-9]

(a) *Test trips*

Name	Video (sec)	Iterations	Shots	Summary (sec)	Animation (sec)
Trip A	120	10	17	77	2084
Trip A	60	2	9	15	427
Trip B	60	4	8	29	627
Trip C	45	6	13	19	226
Trip D	45	3	9	9	172

(b) *Computation performances of our preview videos*

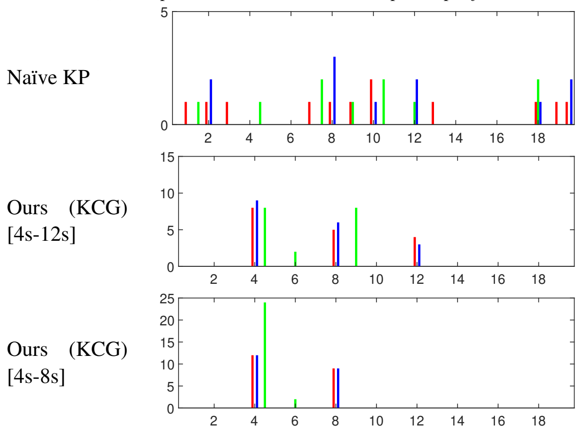
Table 2: *Test trips and preview videos statistics. All trips (a) have been simulated around the Manhattan urban area. For each, we provide the trip split we use in all preview videos. For all videos (b) we give the cumulative computation time for both stages.*

6. Two-step optimization

The camera animation optimization impacts the viewpoint score, due to film-editing constraints steering cameras before and after cuts, while path smoothing constraints will adapt intermediate camera positions and orientations. Hence, we iterate on both optimiza-

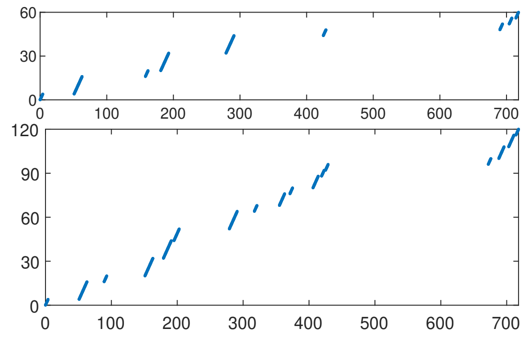
Inputs			Outputs	
Summary technique	Shot duration	Trip split (sec)	Solver time (sec)	Total profit
KP	-	1	0.034	34.97
		1.5	0.019	34.99
		2	0.014	34.93
Ours (KCG)	[4s-12s]	1	3.963	32.83
		1.5	0.622	32.26
		2	0.193	32.24
Ours (KCG)	[4s-8s]	1	0.737	32.34
		1.5	0.089	31.51
		2	0.063	31.27

(a) *Computation time and total knapsack profit*

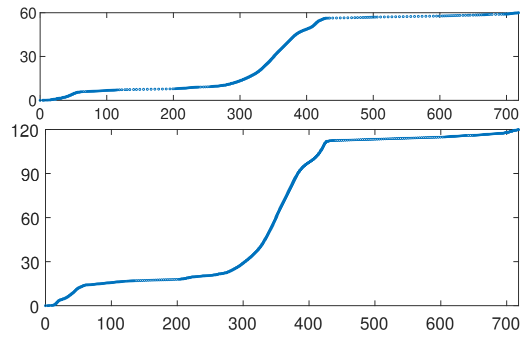


(b) *Output histograms of shot durations. Colored bars correspond to runs with trip splits: 1 sec (red), 1.5 sec (green), 2 sec (blue).*

Figure 6: *Summarization performances of our KCG solver, or a naïve knapsack (KP) solver, on trip A (120 seconds video). Given a trip split, the naïve KP outputs a summary not accounting for shot lengths, nor inclusion of take-off/landing phases. Our KCG-based solution, while solving these additional constraints, remains efficient and provides a total profit close to the optimal.*



(a) *Ours (KCG), shots lengths = [4-12s]*



(b) *Trip Synopsis [HLH*16]*

Figure 7: *Time mappings between the preview video (y-axis) and the conveyed trip portions (x-axis). (a) Using our technique, the slope is 1 for all shots (no speedup). (b) [HLH*16] instead devote more time to some trip portions, and less to boring ones. Our summary seem consistent with theirs, yet we additionally prune boring trip portions. As well, for a twice-longer preview video (bottom rows), we convey more interesting portions, while they seem to devote twice as much time on each portion (even boring ones).*

tion steps described earlier (Sections 4 and 5) to progressively improve the camera animation, while pruning trip segments where camera viewpoints cannot fulfill visual aesthetics criteria. We repeat this two-step process until the best-summary output converges.

7. Implementation and Results

We implemented our algorithms within the Unity3D 2019 game engine, and used the MapBox plugin to generate the 3D urban scenes. We also rely on the AlgLib and Eigen libraries to compute spline curves, integrate costs, and compute the gradient descent. All our results were computed on a desktop computer with a Intel Xeon CPU @3.9GHz and a NVidia Geforce RTX 2080 Super. We also re-implemented [HLH*16] rendering-based importance score computation and speedup optimization, for comparison purposes.

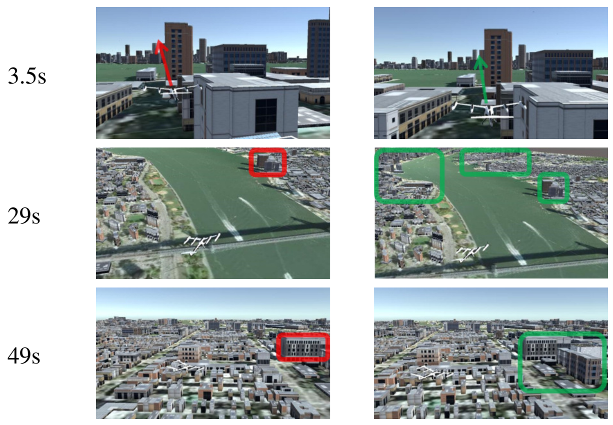
We used our tool to generate video previews of different duration, for four different trips. They take place between sky-ports placed at various locations in a 3D scene reproducing the Manhattan urban area. Screenshots of our preview videos are displayed in Figures 1 (trip A) and 5 (all other trips), while we include generated videos in the supplementary material. Statistics on our trips and video preview computations are also presented in Table 2. We observe that the most expensive step is the optimization of camera animation parameters. Though, we are confident that this step could be improved. In particular we currently do not leverage parallelization, while our costs could be computed in parallel.

We present below a series of comparisons between our optimization steps and alternative methods. In particular, we compare summaries computed using our KCG-based formulation against a naive knapsack formulation, or against the Trip Synopsis system [HLH*16] – the most recent system pursuing similar goals. We also perform an ablation study on our animation step to show how levels of priority impact the quality of generated preview videos.

Knapsack formulation: We compare our formulation with a conflict graph (referred to as *KCG*) and a naïve knapsack method (referred to as *Naïve KP*). For both, we compute summaries of lengths 120s for Trip A, and vary the split length of trip portions, from 1 second to 2 seconds. For our KCG method, we also vary the prescribed range of shot lengths. Results are presented in Figure 6.

	Preferred view	Rule of thirds	Context incl.	Traffic incl.
Before optimization	1 (0)	1 (0)	0.89 (0.07)	0.94 (0.09)
Film editing and Camera motion	0.92 (0.06)	0.99 (0.01)	0.88 (0.07)	0.94 (0.10)
All priority levels	0.96 (0.05)	0.99 (0.00)	0.90 (0.07)	0.95 (0.08)

Table 3: *Ablation study: viewpoint quality scores (section 4.2), according to accounted levels of priority in our camera animation step. For each, we provide the mean score (the higher the better) and standard deviation (in brackets) along all shots. While accounting for film editing and camera motion costs degrades the viewpoint quality, introducing viewpoint quality costs in the last level of priority enables to reinforce the desired visual layout.*



(a) *Film editing and camera motion* **(b)** *All priority levels*

Figure 8: *Ablation study: viewpoints quality. (a) Considering film editing and camera motion costs is not sufficient: (top) the motion space is not ensured, (middle and bottom) the context is badly conveyed. (b) Considering viewpoint quality costs enables to (top) better enforce the motion space and to (middle) better convey the context, among which (bottom) the landing skyport.*

For a given trip split length, the naive KP exhibits an optimal computation time and maximum profit when considering no cinematographic constraint as input (*e.g.* shot durations or inclusion of the takeoff and landing). As illustrated in Figure 6b, regardless of the trip split, it provides no guarantee on the duration of generated shots (some are very short, or very long). In contrast, our KCG can by design solve these extra constraints, while it allows variations in the cutting rhythm, and remains quite efficient (Figure 6a).

Comparison with time compression: We also compare our summaries against those generated with our main concurrent technique [HLH*16] (referred to as *trip synopsis*). Figure 7 presents a side-by-side plot of the time mapping between the preview video and the conveyed trip segments, on Trip A. From this figure, and the accompanying videos, it is clear that our summaries are coherent with those of [HLH*16]. Yet, they devote a large amount of time to few short trip segments, and convey the remaining (as well as the take-off/landing) very rapidly. In contrast, our technique allows a focus on a broader range of relevant trip segments,while our analysis of redundancy still enables a good coverage of the whole trip.

To evaluate our animation optimization step, we further performed an ablation study on our levels of priority (*i.e.* cinematic constraints). For all tests, we input the initial summary computed by our trip summarizing step, and the initial camera animation. We then optimize this camera animation while accounting for an increasing number of priority levels: (i) none of the levels (*i.e.* before optimization), (ii) film editing costs, (ii) film editing and camera motion costs, (iii) all costs (*i.e.* film editing, camera motion, and viewpoint costs). Hereafter, we comment on our results.

Before optimization: As expected, some initial camera shots violate film editing conventions (Figure 9a and 9b). Most obvious violations concern the position continuity rule and the 30 degree rule. Further, as we initiate camera positions in the local frame of the vehicle, the camera seem to be rigidly linked to the vehicle. This lack of camera motion smoothness is clearly visible in Figure 10a, where we observe large variations of the camera acceleration.

Editing costs only: film editing conventions are better satisfied between pairs of shots (Figure 9c and 9d). Yet, these costs only apply on the first and last key of each shot, which tends to distort camera trajectories. In turn, we observe even larger variations of the camera acceleration during shots (Figure 10b).

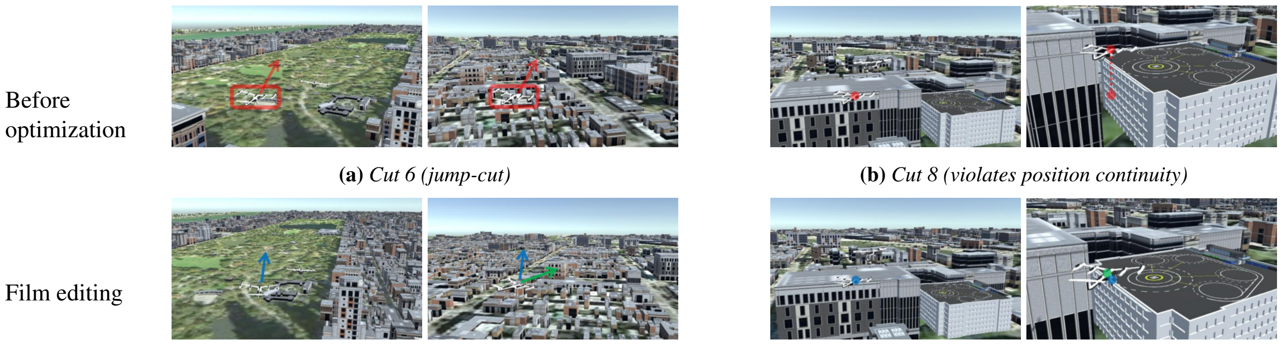
Film editing and camera motion costs only: camera motions become smoother during shots (Figure 10c). Yet, both film editing and camera motion costs steer keyframes in such a way that the camera position diverges from the preferred view, while the desired visual layout remain unsatisfied (Table 3 and Figure 8).

All costs: the satisfaction of the viewpoint scores is improved (Table 3). Figure 8 clearly illustrates how these costs allow to better satisfy the rule of thirds, motion space, as well as the context inclusion at different moments of the video. In figure 10d, we also observe that this improvement of the viewpoints quality comes without sacrificing the smoothness of camera motions.

8. Discussion and Conclusion

In this paper, we introduced a fully automated method to compute a video of a prescribed duration summarizing an input flight, with a user-specified cutting rhythm. To our best knowledge, this is the first approach addressing the three challenging problems of virtual camera control, film editing, and video summarizing, in parallel. Our best-summary solution uses time ellipses to prune the boring parts of a trajectory, enabling to rely on a knapsack problem with conflict graph for generating the result. This solution allows us to effectively enforce common shooting and film editing principles, leading to the generation of summary videos that conform to the main cinematographic principles.

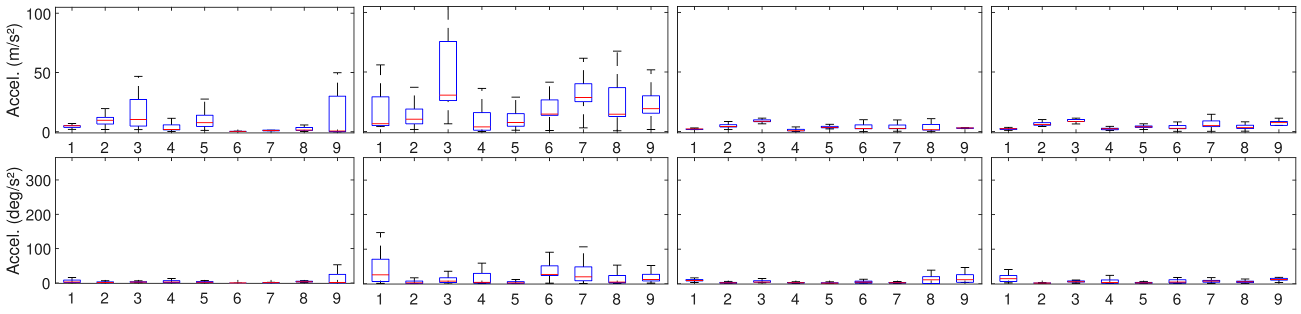
While this work focused on flight summaries, adapting the solution to other cases would only require the definition of adequate semantic features, such as considering crashes and collisions as interest points in a car chase case. Therefore, we believe that our general approach would easily generalize to a broad range of contexts.



(a) *Cut 6 (jump-cut)* **(b)** *Cut 8 (violates position continuity)*

(c) *Cut 6 (30° view angle change)* **(d)** *Cut 8 (position continuity)*

Figure 9: *Ablation study: cuts between shots. (a)(b) When not optimizing the initial camera animation, violation of film editing rules are clearly noticeable. (c)(d) When optimizing with film editing costs: for the same cuts, film editing rules are better enforced.*



(a) *Before optimization* **(b)** *Film editing* **(c)** *Film editing and Camera motion* **(d)** *All levels of priority*

Figure 10: *Ablation study: camera acceleration per film shot (top: camera position; bottom: camera orientation). (a)(b) As expected, not considering camera motion costs leads to strong camera accelerations during shots (c) Considering motion costs in our second level of priority, the camera acceleration is highly reduced during shots. (d) Further levels of priority have low impact on this smoothing.*

A remaining challenging task would then be to compare sets of features in term of the perceived quality of edited videos [LRGG14].

In the future, our best-summary step would benefit from further analysis of redundancy between shots, which is known as a challenging open problem. In addition, regardless of the method used to identify them, highly redundant shots could be expressed as conflicting in our KCG formulation, while keeping the rest of the pipeline unchanged. Lastly, we would like to extend our approach to summaries of multiple flights (*e.g.* a full day of vehicle flights). This would require analyzing complex spatio-temporal relationships between multiple trajectories, and devising higher-level methods to optimize the parameters of multiple interrelated camera animations, under film-editing constraints. From this perspective, it would be interesting to have a better insight of the perception of continuity between shots, for example in action films with chases, and to formalize empirically extracted film-editing rules.

Ackowledgement

We thank the *Integrated Urban Mobilty* chair between Uber and Ecole Polytechnique, for funding this work. We also thank V. Sungeelee for her contributions to a preliminary work on this chair.

References

[AAS*12] Ajmal M., Ashraf M. H., Shakir M., Abbas Y., Shah F. A.: Video summarization: techniques and classification. In *International Conference on Computer Vision and Graphics* (2012), Springer, pp. 1–13. 2

[BB04] Baerlocher P., Boulic R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The visual computer 20*, 6 (2004), 402–417. 7

[BDA04] Barbieri M., Dimitrova N., Agnihotri L.: Movie-in-a-minute: automatically generated video previews. In *Pacific-Rim Conference on Multimedia* (2004), Springer, pp. 9–18. 3

[CNO*09] Chen B., Neubert B., Ofek E., Deussen O., Cohen M. F.: Integrated videos and maps for driving directions. In *ACM symposium on User interface software and technology* (2009), pp. 223–232. 3

[CON08] Christie M., Olivier P., Normand J.-M.: Camera control in computer graphics. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 2197–2218. 2

[ETB15] Elkhattabi Z., Tabii Y., Benkaddour A.: Video summarization: techniques and applications. *International Journal of Computer and Information Engineering 9*, 4 (2015), 928–933. 2

[GCB*19] Gschwindt M., Camci E., Bonatti R., Wang W., Kayacan E., Scherer S.: Can a robot become a movie director? learning artistic principles for aerial cinematography. In *2019 IEEE/RSJ Inter-*

national Conference on Intelligent Robots and Systems (IROS) (2019), IEEE, pp. 1107–1114. 2

[GCLR15] GALVANE Q., CHRISTIE M., LINO C., RONFARD R.: Camera-on-rails: automated computation of constrained camera paths. In *ACM SIGGRAPH Conference on Motion in Games* (2015), pp. 151–157. 2

[GRLC15] GALVANE Q., RONFARD R., LINO C., CHRISTIE M.: Continuity editing for 3d animation. In *AAAI Conference on Artificial Intelligence* (2015). 2

[GW92] GLEICHER M., WITKIN A.: Through-the-lens camera control. In *Conference on Computer graphics and interactive techniques* (1992), pp. 331–340. 2

[HHFA18] HESHAM M., HANI B., FOUAD N., AMER E.: Smart trailer: Automatic generation of movie trailer using only subtitles. In *International Workshop on Deep and Representation Learning* (2018), IEEE, pp. 26–30. 3

[HLH*16] HUANG H., LISCHINSKI D., HAO Z., GONG M., CHRISTIE M., COHEN-OR D.: Trip synopsis: 60km in 60sec. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 107–116. 3, 5, 8, 9, 11, 12

[IH00] IGARASHI T., HINCKLEY K.: Speed-dependent automatic zooming for browsing large documents. In *ACM symposium on User interface software and technology* (2000), pp. 139–148. 3

[JRT*15] JOUBERT N., ROBERTS M., TRUONG A., BERTHOUZOZ F., HANRAHAN P.: An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics 34*, 6 (2015), 1–11. 2

[JWW*20] JIANG H., WANG B., WANG X., CHRISTIE M., CHEN B.: Example-driven virtual cinematography by learning camera behaviors. *ACM Transactions on Graphics 39*, 4 (2020), 45–1. 2

[LC08] LI T.-Y., CHENG C.-C.: Real-time camera planning for navigation in virtual environments. In *International Symposium on Smart Graphics* (2008), Springer, pp. 118–129. 2

[LC12] LINO C., CHRISTIE M.: Efficient composition for virtual camera control. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), Eurographics Association, pp. 65–70. 2

[LC15] LINO C., CHRISTIE M.: Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics 34*, 4 (2015). 2

[LCL*10] LINO C., CHRISTIE M., LAMARCHE F., SCHOFIELD G., OLIVIER P.: A real-time cinematography system for interactive 3d environments. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), Eurographics Association, pp. 139–148. 2

[Lin13] LINO C.: *Virtual camera control using dynamic spatial partitions*. PhD thesis, Université Rennes 1, 2013. 2

[LRGG14] LINO C., RONFARD R., GALVANE Q., GLEICHER M.: How do we evaluate the quality of computational editing systems? In *Workshops at AAAI Conference on Artificial Intelligence* (2014). 10

[NAMD*17] NÄGELI T., ALONSO-MORA J., DOMAHIDI A., RUS D., HILLIGES O.: Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters 2*, 3 (2017), 1696–1703. 2

[NMD*17] NÄGELI T., MEIER L., DOMAHIDI A., ALONSO-MORA J., HILLIGES O.: Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics 36*, 4 (2017), 1–10. 2

[OSTG09] OSKAM T., SUMNER R. W., THUEREY N., GROSS M.: Visibility transition planning for dynamic camera control. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), pp. 55–65. 2

[RH16] ROBERTS M., HANRAHAN P.: Generating dynamically feasible trajectories for quadrotor cameras. *ACM Transactions on Graphics 35*, 4 (2016), 1–11. 2

[Ron21] RONFARD R.: Film directing for computer games and animation. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 713–730. 2

[RU14] RANON R., URLI T.: Improving the efficiency of viewpoint composition. *IEEE Transactions on Visualization and Computer Graphics 20*, 5 (2014), 795–807. 2

[TB09a] THOMPSON R., BOWEN C. J.: *Grammar of the Edit*, vol. 13. Taylor & Francis, 2009. 2, 4, 6

[TB09b] THOMPSON R., BOWEN C. J.: *Grammar of the Shot*, vol. 13. Taylor & Francis, 2009. 2, 4, 5

[YKW02] YAMADA T., KATAOKA S., WATANABE K.: Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Information Processing Society of Japan Journal 43*, 9 (2002). 4

[YMCZ17] YUAN Y., MEI T., CUI P., ZHU W.: Video summarization by learning deep side semantic embedding. *IEEE Transactions on Circuits and Systems for Video Technology 29*, 1 (2017), 226–237. 3

	Level	Curves	Keys	Camera	Look-at
Line of action 29	L_1	S	1, n	1	
Jump cut 28	L_2	S	1, n	1	
Pos. continuity 30	L_1	S	1, n		8
Look-ahead 31	L_2	S	1, n		0.5
Motion direction 19	L_3	S, E	[2 ; n-1]	8	
Camera speed 20	L_3	S, E	[2 ; n-1]	0.5	
View angle 21	L_4	S, E	[2 ; n-1]	0.125	
Look-at path 24	L_3	S, E	[1 ; n]		1
Cam. rotation 22, 23	L_4	S, E	[2 ; n-1]		0.5
Preferred view 14	L_5	S, E	[1 : n]	0.5	
Thirds rule 15	L_5	S, E	[1 : n]		1
Context inclusion 17	L_6	S, E	[1 : n]		0.5
Traffic inclusion 17	L_6	S, E	[1 : n]		0.125
Look-ahead 32	L_6	S	[1 : n]	0.125	

Table 4: *Organisation of camera animation costs into priority levels. For each, we indicate if they apply to shots (S) or time ellipses (E), keys on which they apply to, and if they apply to the camera positions or look-at positions (with their weight in the Jacobian).*

Appendix A: Detailed importance scores

We here detail our trip importance scores. For each, we perform a max-normalization (*i.e.* we divide it by the maximum value along the trip), to ease further calculations.

Context: Showing a tall building located farther along the trip might provide a better hint (*e.g.* the skyline of a city) than a small building located close to the vehicle. Starting from the importance B_j of the jth building (computed using [HLH*16]), we calculate its warped distance to the vehicle at time t:

$$distance(j,t) = \frac{\|\mathbf{b}_j(t)\|}{B_j \left[\hat{\mathbf{b}}_j(t) \cdot \hat{\mathbf{v}}_P(t)\right]^+ f(\|\mathbf{b}_j(t)\|)} \quad (7)$$

where $\mathbf{b}_j(t)$ is the vector from the vehicle to the building top point. Our middle term gives more importance to buildings in the current direction of flight, to favor the provision of hints on where the vehicle is travelling to. Our right term $f(x)$, equal to $min(1, x/x_{min})$, gives less importance to buildings located closer than a threshold x_{min} (in our tests we used 150 meters). Indeed, such buildings would be badly captured by the camera (*i.e.* only very partially), and provide very few hints on where the vehicle is going.

As interest points, we use the K-nearest buildings, regarding this warped distance. They are sorted by decreasing importance B_k ($k \in [1,K]$). To provide a better hint on where the vehicle is travelling to, we also use the landing location as an additional interest point, and set its initial importance to $B_{K+1} = 1$. We then define a context score as a density on all interest points, computed as

$$I_{context}(t) = 1 - G\left(\frac{1}{K+1}\sum_k \frac{k}{distance(j_k,t)}, \sigma_c\right) \quad (8)$$

where j_k is the index of the k-th closest building. In our tests, we used $K = 10$ buildings, and $\sigma_c = 2.10^{-3}$.

Altitude: We set the slope score $I_{slope}(t)$ to the angle between the vehicle speed $\mathbf{v}_p(t)$ and its projection onto the horizontal plane.

Flying direction: We capture the vehicle's sharp turns by monitoring curvature along its trajectory, which we compute as:

$$\kappa(t) = \|\dot{\mathbf{v}}_p(s) \times \ddot{\mathbf{v}}_p(s)\|.\|\dot{\mathbf{v}}_p(s)\|^{-1} \quad (9)$$

using the arc-length parameterization of the vehicle's trajectory $\mathbf{v}_p(s)$. This is done by constructing a monotonic 1d spline curve where each travelled distance s is associated to its trip time t (*i.e.* $\mathbf{v}_p(s) = \mathbf{v}_p(t)$). We then define the curvature score as:

$$I_{curv}(t) = 1 - E(\kappa, 1) \quad (10)$$

The curvature tends to rapidly grow with the actual change in direction. The exponential decaying aims to avoid this effect, so as to highlight changes in a more linear way.

Velocity: we consider a velocity score $I_{vel}(t)$ computed as the norm of the vehicle's velocity vector $\|\dot{\mathbf{v}}_p(t)\|$.

Traffic density is captured in a way similar to context, by monitoring the K most important surrounding vehicles. We compute a warped importance per vehicle as:

$$importance(j,t) = \left[\hat{\mathbf{b}}_j(t) \cdot \hat{\mathbf{v}}_p(t)\right]^+ .f(\|\mathbf{b}_j(t)\|) \quad (11)$$

Here $\mathbf{b}_j(t)$ is the vector between both vehicles (*i.e.* $\mathbf{v}_j(t) - \mathbf{v}_p(t)$). As well, function $f(x)$ is a Gaussian decay $G(|x - x_{min}|, \sigma_x)$, aiming to favor vehicles around a preferred distance x_{min} (in our test we used 150 meters, and $\sigma_x = 300$). Then, we define the traffic importance score $I_{traffic}(t)$ using a density function similar to equation 8, where we use the original distance $\|\mathbf{b}_j(t)\|$ instead of a warped one. In our tests, we used $K = 3$ vehicles.

Appendix B: Detailed viewpoint quality scores

Preferred view: the preferred camera position is computed in spherical coordinate, around the vehicle. Its distance, horizontal and vertical angles are expressed through functions $\delta(r_v, r_a, r_s)$, $\theta(r_v)$ and $\phi(r_v, r_a, r_s)$ of the vehicle velocity (r_v), its altitude (r_a), and the trip summarizing ratio (r_s). Changes to the functions proposed by [HLH*16] are detailed in the supplementary material.

Given these functions, we then check how much this view preference is satisfied by the current camera position at distance $\delta(t)$, horizontal view angle $\theta(t)$ and vertical view angle $\phi(t)$. The satisfaction regarding camera distance is expressed as:

$$Q^{\delta}_{view}(t) = ratio(\delta(t), \delta(r_v, r_a, r_s)) \quad (12)$$

The satisfaction regarding horizontal and vertical camera angles are expressed as:

$$Q^{\theta}_{view}(t) = 1 - \frac{|\theta(t) - \theta(r_v)|}{180} \text{ and } Q^{\phi}_{view}(t) = 1 - \frac{|\phi(t) - \phi(r_v, r_a)|}{180} \quad (13)$$

We finally combine them into a global satisfaction:

$$Q_{view}(t) = \frac{6.Q^{\delta}_{view}(t) + 3.Q^{\phi}_{view}(t) + Q^{\theta}_{view}(t)}{10} \quad (14)$$

In other terms, we accept small variations around the preferred viewpoint, but penalize larger ones. We also place more importance on distance and vertical viewing angle, which provide more critical visual cues. Conversely, the horizontal angle has less impact on the quality of the views, hence small variations should remain allowed.

Thirds rule and motion space are expressed through a desired screen position $\hat{\Pi}$, accounting for both rules, and whose satisfaction is computed as:

$$Q_{thirds}(t) = 1 - \frac{\mathbf{v}_p(t) - \mathbf{c}(t)}{\|\mathbf{v}_p(t) - \mathbf{c}(t)\|} \cdot \mathbf{v}_{\hat{\Pi}} \quad (15)$$

where $\mathbf{v}_{\hat{\Pi}}$ is a unit-vector starting from the camera, corresponding to the screen position $\hat{\Pi}$.

To compute $\hat{\Pi}$, we rely on the screen projection of the vehicle's velocity vector, once normalized. To enforce motion space, the ideal screen position of the vehicle should be opposite to this projected motion vector (Figure 3a). We then warp this ideal position onto the center square formed by the four power lines (Figure 3b), by expressing the ideal screen coordinates as:

$$\hat{\Pi}_x = -0.33\left[\text{sign}(\Pi(\dot{\mathbf{v}}_p(t))_x).\sqrt[8]{\Pi(\dot{\mathbf{v}}_p(t))_x}\right].\sqrt[8]{r_v} \quad (16)$$

and similarly for the y component of the ideal screen position. The left and center parts represent the rule of thirds and the motion space, respectively. Using an L_8 distance ($\sqrt[8]{.}$) attracts the ideal screen position to the border of the center square, on the side opposite to the vehicle's apparent motion, when it is moving fast enough (Figure 3), and to the screen center when moving slow enough (or stopped), with a smooth transition between both cases.

Context and traffic inclusion account for the K-nearest neighbors. For the sake of generality we denote as $\mathbf{v}_k(t)$ the position of the k-th neighbor at time t. Neighbors should be included on the screen:

$$Q_{context}(t) = \frac{1}{\sum_k imp(k,t)} \sum_{k=1}^{K} imp(k,t).incl(k,t) \quad (17)$$

$imp(k,t)$ is the warped importance of the k-th neighbor, defined using equation 11 (traffic density), and the inverse of equation 7 (context). $incl(k,t)$ is the inclusion of k-th neighbor at time t:

$$incl(k,t) = 1 - \cos\left(\left(\mathbf{v}_k(t) - \mathbf{c}(t), \mathbf{d}(t)\right)_{\varphi/3}\right) \quad (18)$$

with $\mathbf{c}(t)$ and $\mathbf{d}(t)$ the camera position and view direction, respectively. φ is the camera's field of view. In other terms, we aim to position important context and traffic elements as close as possible to the center third of the screen.

Appendix C: Detailed camera animation costs

Viewpoint quality

Viewpoint costs for any keyframe k along a curve, defined at time t_k, are computed as $C_f(t_k) = 1 - Q_f(t_k)$ where f is a feature among $\{view, thirds, traffic, context\}$.

Camera motion smoothness

For the sake of simplicity we here use the sub-scripted notation $\mathbf{c}_k$, $\mathbf{l}_k$, $\mathbf{d}_k$ and $\mathbf{q}_k$ to refer to the camera position, look-at point, view direction and orientation at a keyframe k.

Camera path: We first constrain the camera motion direction to change as little as possible, through the cost:

$$C^{direction}_{path}(k-1,k,k+1) = 1 - \frac{\mathbf{c}_k - \mathbf{c}_{k-1}}{||\mathbf{c}_k - \mathbf{c}_{k-1}||} \cdot \frac{\mathbf{c}_{k+1} - \mathbf{c}_k}{||\mathbf{c}_{k+1} - \mathbf{c}_k||}. \quad (19)$$

and its speed to change as little as possible, through the cost

$$C^{speed}_{path}(k-1,k,k+1) = 1 - ratio\left(||\mathbf{c}_k - \mathbf{c}_{k-1}||, ||\mathbf{c}_{k+1} - \mathbf{c}_k||\right) \quad (20)$$

Viewing angle: We penalize changes in the viewing angle on the vehicle, with the cost:

$$C^{direction}_{view}(k,k+1) = 1 - \hat{\mathbf{c}}^{local}_k \cdot \hat{\mathbf{c}}^{local}_{k+1} \quad (21)$$

where $\hat{\mathbf{c}}^{local}_k$ is the camera position at time t_k, transformed in the local frame of the vehicle then normalized.

Camera rotation: We avoid abrupt linear rotations of the camera:

$$C^{direction}_{rotation}(k,k+1) = 1 - \mathbf{d}_k \cdot \mathbf{d}_{k+1} \quad (22)$$

and rotational velocity changes as well:

$$C^{speed}_{rotation}(k-1,k,k+1) = 1 - \left(\mathbf{q}_k \cdot \mathbf{q}^{-1}_{k-1}\right) \cdot \left(\mathbf{q}_{k+1} \cdot \mathbf{q}^{-1}_k\right) \quad (23)$$

where each component of the dot product represents the rotation change between two keys, in quaternion space.

Look-at point: its path is smoothed through the cost:

$$C^{position}_{lookat}(k-1,k,k+1) = 1 - \frac{\mathbf{l}_k - \mathbf{l}_{k-1}}{||\mathbf{l}_k - \mathbf{l}_{k-1}||} \cdot \frac{\mathbf{l}_{k+1} - \mathbf{l}_k}{||\mathbf{l}_{k+1} - \mathbf{l}_k||} \quad (24)$$

Film-editing conventions

We consider cuts between two consecutive shots i and j. For the sake of clarity, we now include the shot index in our sub-scripted notations (*e.g.* $\mathbf{c}_{i,n}$ is the camera position for the last key of shot i).

Jump Cuts (aka 30°-rule): avoidance is expressed as the cost:

$$C^{30°}_{jump}(i,j) = \left[\frac{30 - \left(\mathbf{c}^{local}_{i,n}, \mathbf{c}^{local}_{j,1}\right)}{30}\right]^+ \quad (25)$$

To provide enough size change, we penalize changes on the camera distance to the vehicle. We compute a ratio of distance change:

$$r_{actual} = 1 - ratio\left(||\mathbf{c}^{local}_{i,n}||, ||\mathbf{c}^{local}_{j,1}||\right) \quad (26)$$

then consider that this ratio is sufficient if it is greater or equal to $r_{desired}$, leading the cost:

$$C^{size}_{jump}(i,j) = \left[\frac{r_{desired} - r_{actual}}{r_{desired}}\right]^+ \quad (27)$$

In our tests, using $r_{desired}$ equal to 0.5 (*i.e.* a logarithmic change of camera distance) seem to work well.

To avoid jump cuts, one should satisfy at least one rule:

$$C_{jump}(i,j) = \min\left(C^{30}_{jump}(i,j), C^{size}_{jump}(i,j)\right). \quad (28)$$

Line of action: We consider the rule of not crossing the line of action of a moving object. We here generalize to its line of action during a time ellipse, by accounting for the vehicle's positions before and after a cut (see Figure 4).

We firstly compute the reference frame to this line of action (NB: it is in fact a plane). Its forward vector (defining the line) is given by $\mathbf{v}_p(s_j) - \mathbf{v}_p(e_i)$, once normalized (with e_i and s_j the end time and start time of shots i and j). Its right vector (defining the plane normal) $\mathbf{r}(i,j)$ is given by the cross product of the forward vector with the upward vector. We secondly compute the dot product between the vehicle-camera vector (taken before or after the cut) and $\mathbf{r}(i,j)$. We refer to these two products as as $dot(i)$ and $dot(j)$. To not cross the line-of-action, their signs should be equal.

To define the side where both cameras should lie, we use the sign of the dot product of maximum absolute value (we refer to it as $sign(i,j)$). We finally express our line-of-action constraint as:

$$C^{motion}_{continuity}(i) = \frac{[dot_{desired} - dot(i) * sign(i,j)]^+}{1 + dot_{desired}} \quad (29)$$

for the last key of shot i, and similarly for the first key of shot j. Here, we force the dot product to be sufficient to reinforce the coherence between apparent motion directions. In our tests, we used $dot_{desired} = 0.25$ (*i.e.* force a 15° minimum angle with the line of action). We then average both costs into a final cost $C^{motion}_{continuity}(i,j)$.

Screen position continuity: We penalize abrupt changes in the vehicle's position in screen-space:

$$C^{position}_{continuity}(i,j) = 1 - \frac{\mathbf{v}^{cam}_p(e_i)}{||\mathbf{v}^{cam}_p(e_i)||} \cdot \frac{\mathbf{v}^{cam}_p(s_j)}{||\mathbf{v}^{cam}_p(s_j)||} \quad (30)$$

where $\mathbf{v}^{cam}_p$ is the vehicle's position transformed in the camera space (hence strongly linked to its screen position).

Look-ahead: We express the inclusion of the future position of the vehicle (after a cut) on the screen, as a cost on the camera direction:

$$C^{look}_{ahead}(i,k) = 1 - \cos\left(\ (\mathbf{v}_p(s_j) - \mathbf{c}(e_i), \mathbf{d}(e_i))_{\varphi/3}\ \right) \quad (31)$$

To unable the camera to look toward this look-ahead, we also improve the camera positioning (*i.e.* viewing angle) through the cost:

$$C^{look}_{ahead}(i,k) = 1 - \cos\left(\ (\mathbf{v}_p(s_j) - \mathbf{c}(e_i), \mathbf{v}_p(e_i) - \mathbf{c}(e_i))_{\varphi/3}\ \right). \quad (32)$$