Presentation of 3D Scenes through Video Example

Andrea Baldacci, Fabio Ganovelli, Massimiliano Corsini, and Roberto Scopigno

Abstract—Using synthetic videos to present a 3D scene is a common requirement for architects, designers, engineers or Cultural Heritage professionals however it is usually time consuming and, in order to obtain high quality results, the support of a film maker/computer animation expert is necessary.
We introduce an alternative approach that takes the 3D scene of interest and an example video as input, and automatically produces a video of the input scene that resembles the given video example. In other words, our algorithm allows the user to "replicate" an existing video, on a different 3D scene.
We build on the intuition that a video sequence of a static environment is strongly characterized by its optical flow, or, in other words, that two videos are similar if their optical flows are similar. We therefore recast the problem as producing a video of the input scene whose optical flow is similar to the optical flow of the input video. Our intuition is supported by a user-study specifically designed to verify this statement. We have successfully tested our approach on several scenes and input videos, some of which are reported in the accompanying material of this paper.

Index Terms—Multimedia Content Production, Video similarity, 2D vector field comparison, Computer Animation.

1 Introduction

3D models have become increasingly available, due to advances in 3D shapes acquisition (such as low-cost 3D scanners and image-based algorithms, e.g. SfM), to enhanced in applications for creating them, such as Google Sketchup. Having 3D content available is not enough, it needs to be seen and understood by the public. How this should be achieved depends both on the type of data and on the purposes behind it. For example, presenting a small 3D scanned model is different from presenting a large complex architectural environment, due to the very different geometry and visual properties. In this latter case, a video produced offline is often preferable because it fully exploits the 3D model, by tuning rendering parameters and camera trajectories, and avoids the need for a possibly complex graphical user interface. However, making a video can be costly, both in terms of time and money. Thus, automatic methods for defining camera paths have been proposed [1], [2], which generally try to build camera trajectories following cinematographic rules and/or other constraints.

In this work, we take an entirely different approach.Instead of creating a video from scratch, we start from an already available real or synthetic video and, given the 3D scene to present, automatically produce a new video of the 3D scene which is "similar" to the one provided as input. Broadly speaking, our approach makes it possible to create a video for a 3D scene in the same way our favourite film director might.

The notion of similarity is central to our approach, and since input and output video contents are generally unrelated, such a notion must as far as possible be content independent. For example, we may have an input video sequence showing a dolly-in on a green tree while our 3D scene contains neither trees nor green elements. We therefore focus on the optical flow as the visual attribute to be reproduced in the output video. The optical flow combines the camera movement and the 3D scene geometry and does not imply knowledge of the scene content. Hence, we state that two video sequences are similar if their optical flow is similar. This choice is supported by a user study, described in the Results section, which shows, among other things, that videos with very similar optical flows, are also perceived as similar despite their content. Note that we use the optical flow as an approximation of the motion field, which is a common strategy when the 3D scene is unknown. However, since we know the 3D scene for the production of the output video, what we really do is to match the optical flow of the input video with the motion field of the output video. In the following, we use the term *flow field* to indicate both the optical flow of the input video and the motion field of the output video.

Our algorithm works as follows: first we estimate the camera path from the input video sequence and compute a per-frame descriptor of the video, based on its optical flow. Such descriptors are matched against a non-redundant database of the same flow field descriptors extracted by pre-processing the 3D scene of interest. The results of this matching form the basis used to relocate the estimated camera path inside the scene of interest. A complete description of the algorithm and the details of the processing steps are provided in the following sections.

Beside the main result of this work, the automatic generation of high-quality video sequences to present a 3D scene, we contribute to the state of the art by outlining:

- A technique to create a database of non-redundant flow fields starting with a given 3D scene.
- A strategy to index and efficiently retrieve flow fields.

2 Related Work

Our approach borrows from several fields and a complete overview of the literature related to all such fields is beyond the scope of this paper. Below we concentrate on the works most closely related to the original contribution of this paper, i.e.: assisted or automatic camera paths in 3D scenes, the use of cinematography in computer graphics and the metrics for estimating video similarity.

2.1 Assisted or Automatic Camera Paths in 3D Scenes

Several approaches share the idea of building a graph of the empty space of the scene and deriving camera trajectories from paths on this graph. In Salomon et al. [3] the graph is built by random sampling the empty space and connecting the paths which are collision-free. Andujar et al. [2] compute the graph explicitly by positioning the nodes on the medial axis of the empty space, which is approximated by voxelizing the scene and computing a distance field. A similar result is obtained by Di Benedetto et al. [4] where a *k-means*-like approach is used to distribute the graph nodes so that each point of the surface of the scene is seen by at least one node. In Oskam et al. [5] a regular grid is used of as large spheres as possible that do not intersect the model, so that each segment connecting a pair of overlapping spheres defines a collision free arc. Paths in the graph are then refined and smoothed. In Secord et al. [6] a criterion for assessing how good a specific view is given by combining a number of known other criteria [7], [8], [9] and fitting the outcome of a user test case. They were thus able to provide a path along which the view is perceptually "optimal". The main limitation of this study is that it treats small objects by only using a world-in-hand paradigm: the path is defined over a bounding sphere and the view direction is assumed to always point toward the sphere center.

2.2 Using Cinematographic Criteria

Using cinematographic criteria for presenting 2D-3D content is not a new idea. The seminal paper by He et al. [1] formalized concepts of cinematographic language to produce automatic videos, and a tool for shooting dynamic scenes without user assistance is also proposed. Kardan and Casanova [10] build on this approach and propose a system for the cinematographic shooting of a scene with many actors. Note that these systems, like other similar ones, assume semantic knowledge of the scene, that is, a scripted animation. Cinematography is also used in presenting 2D scenes, with the ubiquitous Ken Burns effect, which consists of combining zooming and panning action on a single image to *animate* still photographs. The technique is also used in famous films such as "La jetée"(1962), where a small trembling effect is also added to convey a stronger feeling of watching a real take. More recently, Zheng et al. [11] extended this technique by considering a small portion of a light field and adding a parallax effect without the need for segmenting front and backward parts of the image.

To some extent, these approaches pursue the same goal as we do, that is, to provide a compelling view of 3D content. Most try to include cinematographic principles and language into the choice of a good point of view or planning the camera path. Our approach is radically different. We do not try to formalize the principles of film direction, instead we "clone" the optical flow of a specific video and transfer it to another scene in order to convey the same final impression/perception as the input video. In order to do this, we need a way to establish whether two videos are similar, a problem already raised in other fields, such as video retrieval.

2.3 Similarity Between Videos

Video similarity is a very general concept and various attempts have been made to reduce this concept to quantifiable criteria. Cherubini et al. [12] studied how several characteristics influence the human perception of the similarity of two videos: encoding parameters, overlay, audio commentary, photometric variations and semantic content are just a few discriminating factors. The problem mainly occurs in Content-based Video Retrieval, that is the problem of finding, in a large database of videos, the most related to a new video provided as input using only the visual information (i.e. audio information is not used). Typically, adding a video to a database requires processing it, segmenting it into sequences, to extract low-level features and creating a video signature from these features for fast retrieval (see [13] for a recent survey). Since in our case the visual content of the input video may be very different from the 3D scene of interest, we cannot use techniques that rely on color or texture features. Our method to compare videos is close to all those video retrieval approaches that use motion features for indexing and retrieval. Fablet et al. [14] use the causal Gibbs model to represent the spatio-temporal distribution of local motion-related measurements and use a statistical framework for the retrieval. Ma and Zhang [15] generate a motion texture for each frame, which shares some similarities with the histogram-based local descriptor we propose.

A subfield of video retrieval is Near Duplicate Video Retrieval (NDVR) (see [16] for a recent survey). NDVR is used for detecting copyright infringement and redundancy in large video databases such as YouTube, and the problem lies in finding when two videos are "essentially" the same. Motion estimation has proven to be a valuable feature also in this context. The method proposed by Hampapur et al. [17], splits the frame into sub-images, quantizes the motion direction of each sub-image, then uses the distribution of the vector on the possible directions as a per-frame signature. Instead of considering static cells, Hoad and Zobel [18] detect the motion of the centroids of the brightest and darkest areas of the frames and then use the distance between these centroids as a signature for the key frames.

3 Our Approach at a Glance

Our approach relies on a function $D(A, B)$, described in detail later in this section, which defines the distance between two videos of equal length. With this function we state our problem as: *given an input video A and a 3D scene S, produce a video B of S that minimizes $D(A, B)$.*

Since the video we are looking for will be produced by moving a camera along a camera path inside the scene of

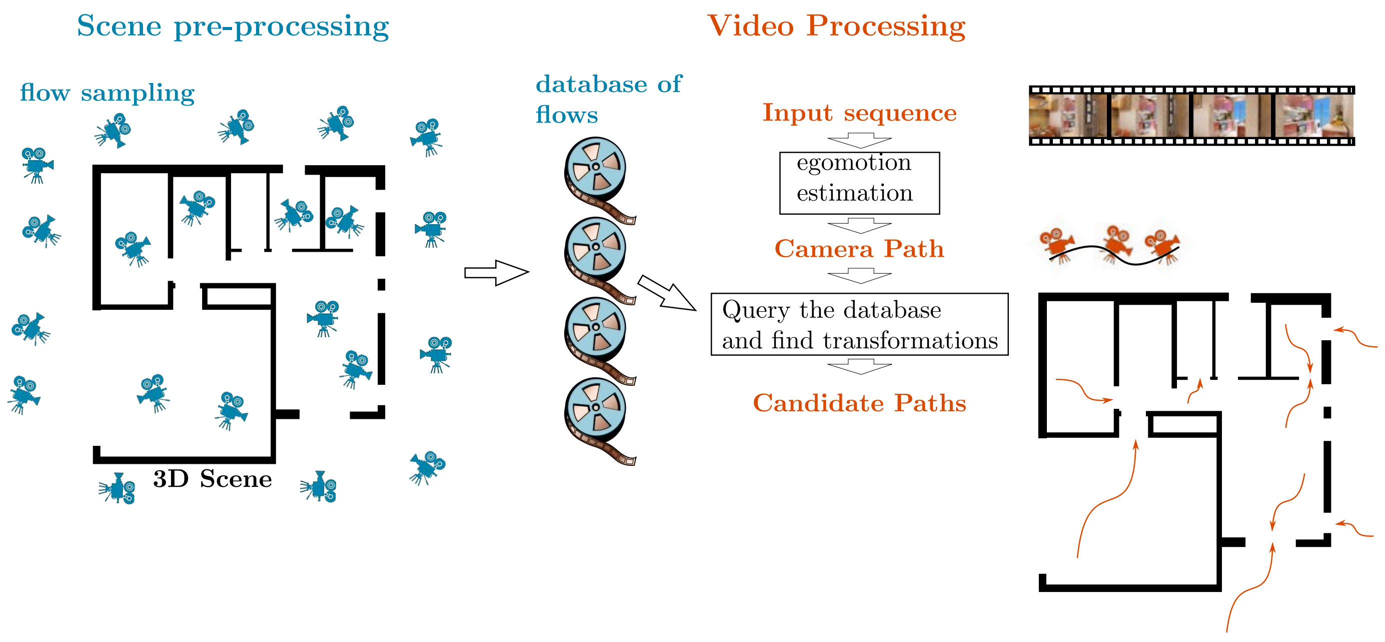


Fig. 1. The two main stages: scene pre-processing and video processing. The pre-processing phase consists of a sparse sampling of all the possible motion fields that can be obtained by moving a camera in the input scene and efficiently storing them in a database. This phase does not depend on the input video and thus is performed once for each 3D model. Video processing starts by extracting the optical flows from the input video and its estimated camera path. With this information we query the pre-calculated database and then transform the input path to best fit the most similar scene flows.

interest, we can re-formulate the problem in terms of camera paths:

$$\min_{x \in \mathcal{P}} D(A, v(x)) \tag{1}$$

where $\mathcal{P}$ is the set of all the possible camera paths and $v(x)$ is the video produced by the path x. More specifically, we define a camera path as a discrete sequence of cameras $x = \{C_0, C_1, \ldots, C_k\}$, where a camera C_i is defined by its extrinsic parameters, that is, its position and orientation.

The brute force solution to this problem would be simply to run over all the possible paths and pick up the one with the minimum D. Clearly, the combinatorial complexity of the dense sampling of camera positions and orientations would make the problem intractable. We thus designed a pipeline which, after a pre-processing of the 3D scene, makes the problem solvable for any input video within a few minutes.

Figure 1 shows a high level description of our algorithm. The pre-processing phase of the scene consists of a sparse sampling of all the motion fields that can be obtained by moving a camera within the scene (see Section 5.1). These flows are stored in a database that can be efficiently queried to return the set of similar flows, with respect to a certain distance function ϕ, to one provided as the input (see Section 6 for further details). Note that this database only depends on the input scene and it needs to be created only once per model. As will be shown later, creating a database of instant flows for a typical CAD model requires a few hours.

The video processing phase takes the input video sequence and estimates the optical flows and the camera path (up to a scale factor) that produced the sequence (see Section 7). The domain of the problem in Equation 1 is then restricted to those paths obtained as a similarity transformation of the camera path estimated from the input video (see Section 7.1). By exploiting the database of flows built in the pre-processing phase we find a set of candidate paths. A voting scheme inspired by the Hough Transform is used to select the best candidate paths (see Section 7.2). Since there are many camera paths with very similar motion fields, function D may have wide plateaus. We therefore use a clustering algorithm to group the selected paths into classes of equivalence (see Section 7.3). Finally, we perform a non-linear optimization for refining each camera path (see Section 7.4). The refined path with the smallest D value is chosen as the output.

4 Comparing Videos by Means of Motion Fields

The notion of similarity between two videos is central to our problem. Note that we cannot assume that the input video and the video to be generated are related in terms of color and texture information. The first can be any video sequence (e.g. downloaded from the internet) and the second is a synthetic video of the scene of interest. We thus compare two videos through the movement of the camera w.r.t. to the static scene, which is captured by the motion field, that is, the 2D vector field where each vector describes the displacement of the projection of a point of the scene between two consecutive (or nearby) frames. As stated in the Introduction, the motion field of the input video is approximated by its optical flow. Since we want to produce a video of the same length as the input video, input and output videos have the same number of frames. We define the distance between two videos A and B as:

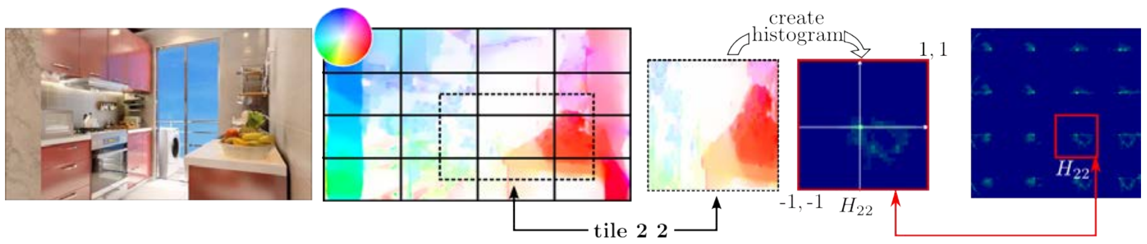


Fig. 2. Creation of the descriptor of vector field. The color wheel shows the color coding that maps the vectors from $[-1,-1] \times [1,1]$ to the RGB color. Tile $(2,2)$ is singled out to show how the relative distribution is stored with a histogram.

$$D(A,B) = \sum_{i=0\ldots k-1} \phi(F(A_i), F(B_i)) \tag{2}$$

where k is the number of frames, A_i and B_i indicates the i^{th} frame of the video A and B, respectively, $F(.)$ is a function returning the flow field and ϕ defines the distance between two vector fields as explained in Sections 4.1 and 4.2.

There are many algorithms for computing the optical flow from an input video, the Middlebury Benchmark website [19] lists about 120 different solutions. For each pixel of one image, the problem is to find the best candidate to be the projection of the same 3D point on the other image. Computing the optical flow of a video is thus prone to errors, especially in the textureless parts of the image. We evaluate the optical flow using the algorithm by Brox et al. [20] which has a good trade-off between computation time and accuracy/robustness. On the other hand, we can compute the motion field of the virtual camera in the 3D scene by re-projecting each pixel back in the 3D scene and then onto the next camera frame.

4.1 Flow Descriptor

To design a distance function between two 2D vector fields we must consider their nature. Since ours are produced by a moving camera in a static environment, we can expect these fields to be quite spatially coherent. For example a *camera crane* (a straight upwards panning in CG parlance) would produce a vertical vector flow pointing downwards, while a *dolly in* would produce a vector field where all vectors are radially oriented away from the image center.

In order to build our field descriptor, we first normalize it by dividing all the vectors components by their maximum over all the images. Since for real videos there may be large outliers due to mismatching pixels, these maxima are computed after a simple outlier removal step, where all the vectors larger than a factor 1.5 of the standard deviation are discarded.

Then we divide the images in regular $T \times T$ tiles. Within each tile the vector field tends to exhibit a slow varying direction and magnitude. This flow can be easily represented by a 2D histogram of the distribution of vector values H_{hk}. In other words, each histogram H_{hk} stores the distribution of vectors in the tile (h,k) in $N \times N$ bins. Our local descriptor is the set of T^2 two-dimensional histograms, one for each tile from which the image is subdivided. Figure 2 shows how the histogram is built considering each tile plus 50% of the neighborhood tiles. Considering overlapping tiles mitigates the aliasing effects that can be produced. For example, a small translation of the video may correspond to a very large difference in the corresponding histograms. Note that, due to removing the outliers the amount of total units per histogram may not be always the same. This is why we normalize each histogram.

4.2 Distance Between two 2D Vector Fields

We define the distance between two flow fields as follows:

$$\phi(a,b) = \sum_{\substack{h=0\ldots T-1 \\ k=0\ldots T-1}} EMD(H'(a_{hk}), H'(b_{hk})) \tag{3}$$

where H' is the normalized version of the histogram H and *EMD* is the *Earth Mover's Distance* (see Rubner et al. [21]). The EMD defines the distance between two histograms as the minimum number of units to move in order to turn one histogram into the other. In other words, it is tightly related to the amount of "work" necessary to transform the first histogram into the second one. In our implementation, equation 3 is efficiently computed using the L1 version of the EMD, called EMD-L1 [22], which uses the Manhattan distance instead of the standard L2 ground distance.

4.3 Flow Comparison and Descriptor

Our definition of the distance between flows (see equation 3) depends on the number of tiles in the subdivision of the image (defined by parameter T) and on the number of bins (defined by parameter N) in the histograms. These parameters affect both the value of $\phi(a,b)$ and the time to compute this value. After a few trial and error tunings, we found that values of T under 4 produce very similar descriptors for dolly-in and dolly-out camera movements, therefore failing to properly discriminate between them, and values of N under 16 also led to noticeably different camera movements to produce similar descriptors. Finally, we found that $T = 4$ and $N = 32$ provides satisfactory results.

Since higher values of T and N require a longer processing time, we also investigated whether values higher than the selected ones could improve the descriptor. We thus created a reference set of 100 flow fields, computed by picking random consecutive frames for a video. We then chose a pivot field and sorted the other 99 in ascending order with respect to ϕ. We then repeated this sorting for different values of parameters T and N, and then compared the sequences with the original one. We used *Spearman's Rank-Order Correlation (ρ)* [23] to measure how much the ordered sequences differed. The comparison demonstrated that higher values of T and N basically produced the same results (e.g. $\rho = 0.96$ for $T = 8, N = 32$ and $\rho = 0.93$ for $T = 8, N = 64$). This confirms that $T = 4$ and $N = 32$ is a sensible choice for these parameters.

5 Pre-processing the Scene: a Database of Instant Flows

The basic building block of our pipeline consists of turning a 3D scene into a (large) database of motion fields. We define *instant camera* as a pair (C, Δ) where C is a camera and Δ is a small 3D displacement, and *instant flow* is the normalized motion field obtained considering the image produced by the camera C and its translated version $C + \Delta$. The length of Δ is chosen to be sufficiently small to guarantee that there is virtually no disocclusion between C and $C + \Delta$. This is achieved by rendering the scene with C and setting $\|\Delta\| = \frac{10 D_{\min}}{w\ f}$, where $D_{\min}$ is the minimum value of the depth map, w is the viewport size in pixels and f is the focal distance. This choice bounds the maximum length of a flow vector to be at most 10 pixels.

With these new definitions, the problem of creating the database of instant flows corresponds to sampling the space of our scene with a suitable set of instant cameras. Each instant camera is defined by eigth variables: three for the position, three for the orientation and two for the direction of the displacement vector Δ.

A simplistic way to build the database would be to perform a regular dense sampling of the scene for the camera position, and to consider for each position a set of predefined orientations and displacements. Clearly that the number of samples would rapidly grow to an unmanageable size. For example, even considering a $4 \times 4 \times 2\ m^3$ empty room with a sampling rate of a position every 10 cm, 128 orientations and 64 displacement directions would give over 262M instant cameras. The next section describes an ad-hoc strategy to obtain an efficient time and space sampling.

5.1 Adaptive Sampling of the Instant Flows

Let $\mathcal{C}^p$ be the set of all instant cameras with position p, that is, that differ only in terms of their orientation and displacement, and let $S(p)$ be a panoramic depth map taken from point p. If we create the instant flow of any instant camera in $\mathcal{C}^p$ by rendering the depth map $S(p)$ instead of the actual 3D scene, the result will differ significantly only where the disocclusions are, i.e. the part of the scene visible from $p + \Delta$ and not from p (and thus not in the depth map created from p). Since Δs are small (see below), we can reasonably assume that there are few disocclusions and that their effect is negligible. It follows that if the panoramic depth maps for two points are the same, the corresponding set of instant flows will also be the same. We can thus reduce the number of dimensions of the domain to be sampled from eigth to three, that is, the camera positions. Therefore, the original problem can be re-stated as finding a sparse sampling of 3D points (with associated sets of instant flows) according to panoramic depth map differences.

The sampling process is incremental and consists of sampling the volume with a layered series of the Poisson Disk distribution with a decreasing disk radius. The samples are generated with a technique inspired by the approach by White et al. [24] for the 2D domain. We start by initializing the sampling with a radius of $R = 1/20^{\text{th}}$ of the scene bounding box diagonal. Then, at each sampling step the radius is reduced by a factor equal to 0.8. Each candidate sample at the i^{th} step is inserted if there is no sample within a radius $(0.8)^i R$ whose associated panoramic map is too similar to that associated with the candidate sample. In our implementation, the panoramic depth maps are implemented as cube maps and the similarity is taken as the mean difference of depths. Note that we add a large sphere containing all the scene to prevent the constant far-plane depth values from biasing the depth map comparison, as is done in [4].

6 Indexing and Retrieval

For each sampling point p, we create the set of instant cameras $\mathcal{C}^p$ and the corresponding instant flows by rendering the scene from C_i and $C_i + \Delta_i$ for all $C_i \in \mathcal{C}^p$. These instant flows are then stored in a database.

Since we need to perform many *nearest neighbours* queries on this database, we want the query to be efficient. Unfortunately, the Earth Mover's Distance, which is at the core of the definition of our measure ϕ, is not amenable to organizing the data into a hierarchical data structure. Furthermore, computing the *EMD* entails solving a max-flow problem, which is computationally demanding. We thus define a Hamming Embedding, i.e. a binary hashing, to map a real-valued vector containing the data of interest, to a vector of boolean values. We then use the Fast Matching algorithm by Muia and Lowe [25] for the fast retrieval of nearest neighbours with respect to the Hamming Distance. This algorithm is implemented in the FLANN library [26]. This is a classic strategy for many applications, for example in the field of image retrieval. Storing the binary descriptor instead of the original onealso enables us to reduce the size of the database by about one order of magnitude.

Obviously, it is crucial that the embedding and the Hamming distance produce similar mapping from histograms to vector of bits. Following Joug et al. [27] we create the embedding as follows. Each $N \times N$ histogram H_{hk} is regarded as a N^2 vector of real values. A matrix Q of m random orthogonal vectors is generated and forms a basis of a m-dimensional subspace of R^{N^2}. Then, the embedding for a tile is defined as

$$\begin{aligned} H'_Q(a_{hk}) &= Q\ H'(a_{hk}) \\ emb(a_{hk})[i] &= \begin{cases} 1 & H'_Q(a_{hk})[i] > 0 \\ 0 & H'_Q(a_{hk})[i] \leq 0 \end{cases} \end{aligned} \tag{4}$$

The bigger m, the more accurate the mapping, and the larger the database. In our experiments we set $m = 128$ leading to a 128 bit long binary signature for each tile. The final instant flow signature is simply obtained by concatenating the signature of each tile thus obtaining a $128 \times 4 \times 4 = 2048$ bit signature for the descriptor of each instant flow. Hence, the database size is reduced by a factor of 32, that is 2 KB instead of 64 KB (4×4 tiles, 32×32 bins).

The execution of a query is a two-step process: we first retrieve the closest 2048 neighbors according to the embedding, and then re-rank the result of this query according to

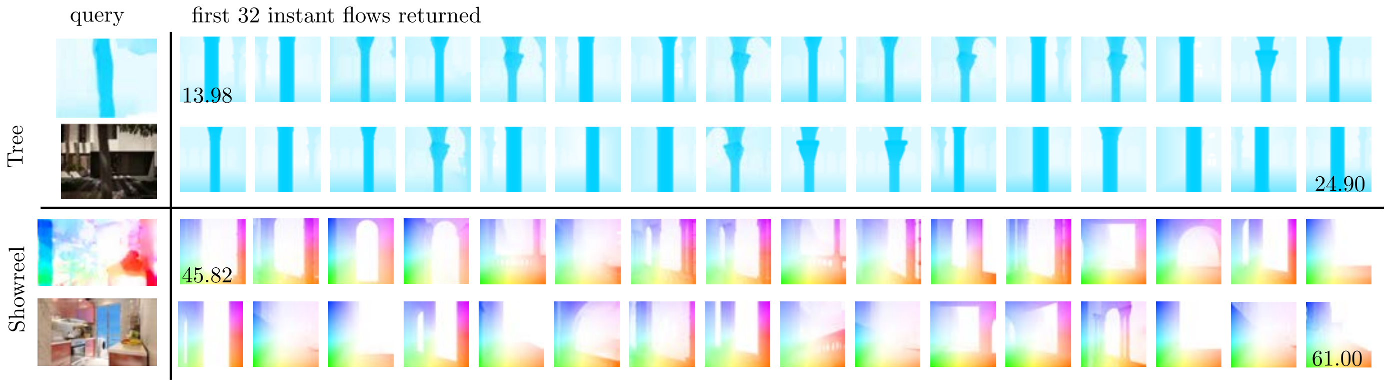


Fig. 3. Two queries from two separate videos on the databases: Sibenik and Museum (see Section 8). (Top) The input flow is produced by a camera track behind a tree. The result is a set of instant flows produced by a camera track behind various columns of the Sibenik 3D model (Bottom) The input flow is produced by entering a room. The result is a set of instant flows produced by passing through an arch of the Museum 3D model.

the ϕ distance (that is, with the EMD distance) in order to find the most similar instant flows.

Figure 3 shows two examples of queries and their results on the databases: Sibenik and Museum (see Section 8). We also report the value of ϕ for the first and the last returned instant flows. Notice that the error range of the first query ($[13.98, 24.90]$) is much smaller than the second one ($[45.82, 61.00]$), which presents a more cluttered flow.

6.1 Two-step Query Accuracy

As explained above, the query of the instant flows is approximated by a two-step approach. First, we query the database of the embedding vectors. Second, we sort the results according to the function ϕ. Since there is no guarantee that the closest 2048 w.r.t. to the Hamming distance will contain the closest 128 w.r.t. to the proposed ϕ distance we evaluated the accuracy of this approximation empirically.

We compared our query result with the result obtained by a linear search in the entire database using ϕ and taking the closest 128 elements. More precisely, we measured the accuracy $\mathcal{A}$ as:

$$\mathcal{A} = 100\,\frac{\#\{b\;|b \in \mathcal{N}(a)\;\wedge\;\phi(a,b) < \phi(a, EMD_{128})\}}{128} \quad (5)$$

where $\mathcal{N}(A_i)$ is the set of instant flows returned by our query and EMD_{128} is the last element of the result returned by the linear search. An accuracy of $x\%$ means that the $x\%$ of $\mathcal{A}$ is within the distance of the 128^{th} flows returned by the linear search. We evaluated the query accuracy $\mathcal{A}$ over 100 frames randomly selected among an input sequence. Eight 3D models were considered for this evaluation. For each model, a database of about 150,000 instant flow was generated. The number of instant flows was chosen so that the linear search could be done in tractable time and, at the same time is high as in the real scenario. On the average we achieved about the 52.5% of accuracy, which is more than enough for our purposes.

7 Processing the Input Video

Given a shot of a static scene we can estimate, up to a scale factor, the trajectory followed by the camera that produced it. This is process is known as *camera tracking*. In out setup this is done by using the Voodoo Camera Tracker software. Given the estimated camera path $P(A)$ and the corresponding input flow $F(A)$, we can define the search space of our problem as the domain of the siilarity transformations of $P(A)$. In other words, we can relocate and scale the path $P(A)$ in our scene by looking for the transformation that best replicates the flow of the input shot $F(A)$.

$$\arg\min_{\tau\in\Omega} D\left(A, v(\tau(P(A)))\right) \quad (6)$$

where Ω indicates the group of angle preserving affine transformations and $v(.)$ indicates the video corresponding to the given path.

7.1 Finding Candidate Paths

Our aim is to reduce the search space in (6) to those transformations that are most likely to provide a good result. We now show how this is done using the database of instant flows. Figure 4 illustrates an example with a path and a scene. Given a generic frame A_i, we can define the corresponding instant camera (C_{A_i}, Δ_{A_i}) where $C_{A_i} = P(A_i)$ and $\Delta_{A_i} = \nabla P(A_i)$. By querying the database for $F(A_i)$ we retrieve a set of instant cameras whose flow is similar to $F(A_i)$ that is, $\mathcal{N}(A_i)$ defined in Section 6.1. Each instant camera $C_h \in \mathcal{N}(A_i)$ defines a transformation RT such that $RT(P(A_i)) = C_h$, that is, the rigid transformation from the camera $P(A_i)$ to the camera C_h. This transformation is found by expressing the camera parameters as a set of 3D points and solving the point matching problem as in [28]. More precisely, each instant camera (C_h, Δ_h) is associated with one point for its origin, three along the axes, one for the direction of movement Δ_h.

Note that choosing a single point on A does not uniquely define a transformation τ, because the scale factor is undefined. In other words, Figure 4 highlights that there are infinite transformed versions of $P(A)$ that go through (C_h, Δ_h) in $P(A_i)$. If, instead, we choose at least two points in $P(A)$, the scale will be determined and the resulting transformation will uniquely identify a *candidate path*. In summary, for each pair of instants (A_i, A_j), we can create a set of *candidate paths* $\Gamma(i,j) = \{\tau_{hk}|\,(C_h, C_k) \in \mathcal{N}(A_i) \times \mathcal{N}(A_j)$. We recall that the sets $\mathcal{N}(A_i)$ and $\mathcal{N}(A_j)$ correspond to those instant cameras whose flow is similar to $F(A_i)$ and

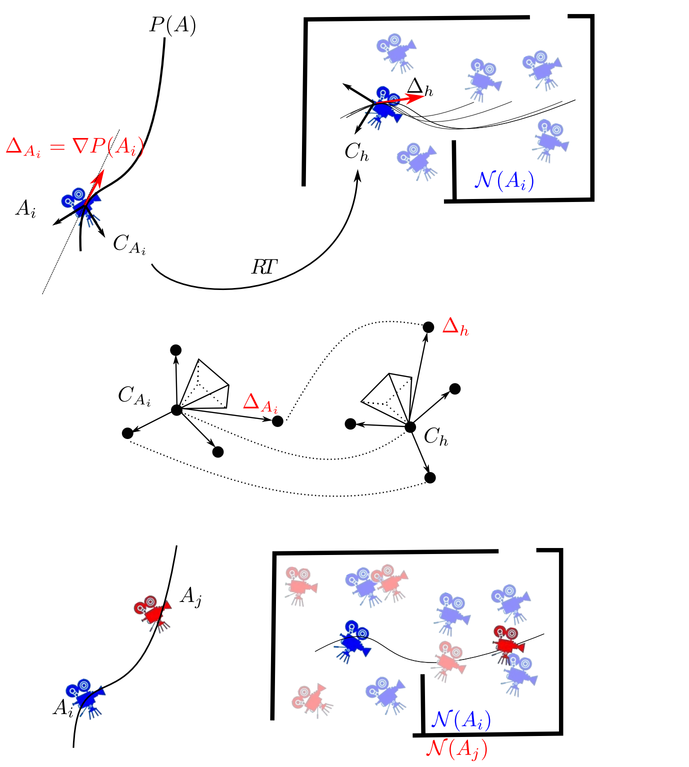


Fig. 4. (Top) The path $P(A)$ is estimated from the video A and an instant camera is created for frame A_i. For each instant camera $C_h \in \mathcal{N}(A_i)$ there is a rototranslation operation which brings the camera C_{A_i} on C_h, leaving the scale value undefined. (Middle) How a set of 5 points is built from an instant camera in order to find the transformation RT. (Bottom) Setting a correspondence for two instant cameras defines the scale factor.

$F(A_j)$. Therefore, the terms of the sum in equation 2 will tend to be small for frames i and j. What we need is a path for which as many as possible terms of that sum are small. The problem thus becomes a fitting problem where D is the error function and where each pair of frames (A_i, A_j) produces a set of candidate paths, that is, solutions, $\Gamma(i, j)$.

7.2 Best Paths Selection

In order to select the best paths we follow a Hough Transform-like approach [29]. Any pair of instant flows chosen along the path $P(A)$ produce a set $\Gamma(i, j)$ of candidate paths. It is likely that distinct pairs of instant flows may produce non-disjoint sets of candidate paths. Using a voting scheme we can select those candidate paths that appear in most sets. Votes are accumulated in a 7-dimensional matrix $\mathcal{M}$ where each cell corresponds to a value for the rototranslation and scaling parameters that define a candidate path. The voting scheme uses a subset of all possible pairs as follows. First the video is adaptively subsampled with respect to the change of optical flow: the higher the change the denser the sampling. We conservatively set the subsampling scheme to keep the 70% of the frames. Then we consider all the pairs (A_i, A_{i+1}), (A_i, A_{i+2}) and (A_i, A_{i+3}). This allows us to use only $3(0.7K - 1)$ frames out of the $K(K - 1)$ possible pair of frames without losing the local coherence of the path.

For each pair (A_i, A_j)

1) We retrieve the two sets $\mathcal{N}(A_i)$ and $\mathcal{N}(A_j)$ from the database
2) We compute the set of candidate paths $\Gamma(i, j)$
3) We quantize the elements in $\Gamma(i, j)$ thereby obtaining $\Gamma'(i, j) \subseteq \Gamma(i, j)$
4) for each element in $\Gamma'(i, j)$ we add 1 to the cell $\mathcal{M}[\Gamma'(i, j)]$ and to the 7-dimensional ball of radius 1 (in cells) centered in $\Gamma'(i, j)$

Thanks to the quantization in step 3, candidate paths that fall in the same cell in the parameters space do not lead to multiple votes. The extension of the vote to the adjacent cell in step 4 is simply to mitigate banding effects.

7.3 Clustering

At the end of the voting the peaks of the accumulation matrix will correspond to those paths that were voted by the greatest number of pairs of frames. As expected, there is not just one solution to our problem. The optical flow produced by a camera moving along a corridor may be replicated on each and every corridor of our 3D model. Thus usually there are many reason the peaks of matrix $\mathcal{M}$. Moreover, a set of peaks close to each other may essentially describe the same path. We thus apply a clustering algorithm to the peaks, that computes a few well defined and distinct set of output paths. In the current implementation clustering is carried out by a simple k-means on the position of the camera. We found that this is sufficient for our needs. Parameter K controls the number of distinct paths produced by the system. In our experiments we kept this parameter quite high ($K = 100$) in order to examine how the different paths explored the scene of interest. The number of distinct paths can also be estimated automatically by applying one of the methods for estimating the number of clusters of a dataset (see [30] for an overview).

7.4 Final Refinement

In order to create the database of instant flows, we quantized the set of all the possible orientations and displacements. To find the candidate paths, we did the same for the $7D$ space of transformations. Finally, we applied clustering to the final outcome in order to return a usable number of solutions. Although these approximations were necessary and are fine tuned, it is clear that they may lead to sub-optimal results, that is, paths which could be improved with small variations. We thus applied a final optimization procedure to each path returned by the clustering step (that is, each transformation τ in equation 6). We initialized equation 6 with the returned transformation, and ran the optimization with NEWUOA [31] (a derivative-free optimization algorithm for unconstrained minimization problems), on the 7 parameters of the similarity transformation.

Since we consider K clusters, we should solve the optimization problem K times. Instead, we first order the K path according to the distance D. Then, we only refined the first $\tilde{K} < K$ paths (we used $K = 100$ and $\tilde{K} = 5$ in our implementation). We refine only a subset to save computation time due to the fact that the refined path is quite close to the non-refined one and the order does not change significantly. The final output video was chosen as the path with the minimum distance D after the refinement. In the accompanying video we show an example of some of the generated solutions ranked according to the distance D.

8 Results

This section is organized in two parts. The first part reinforces the hypothesis we built our pipeline on, that is: the optical flow plays a fundamental role in perceiving video sequences as similar in static scenes. In order to do this, we show the results of a user study we conducted.

The second part shows that our algorithm is efficient in producing video sequences that are similar to the input one in terms of time and resource use. Some output videos produced with our system can be found in the accompanying material.

8.1 User Study

The idea behind this user study is to investigate the role of the optical flow in evaluating the similarity of two video sequences.

The test consists in showing a short reference video to the user plus other five video sequences and asking him/her to sort these five sequences from the most to the least similar to the reference video. It was not specified what *similar* means, that is, no suggestions were given concerning the criteria the user was supposed to use in sorting the sequences. We created five such tests with different models and camera movements (described below) and presented them to each subject through a web page. Figure 5 shows a snapshot of the web page for one test. The GUI was designed to be as straightforward as possible: when all the videos had been watched, the user could drag and drop the corresponding thumbnails to the right slot, rearranging and re-watching the videos without restrictions. All the users were presented with the same batch of five tests in a single scroll-down web page. The order in which the tests and the five videos of each test were laid out was always randomized to avoid possible bias. The total duration of the experiment was about 15-20 minutes. The user study was conducted with 47 subjects: 75% of whom were male and 25% female. Around 75% of the subjects had no specific skills in computer graphics or video-related topics. On the opening page, a short video tutorial showed users how to perform the test and a brief textual description of the study itself is provided. Personal data were collected using a form at the time of the results submission.

All the video sequences were obtained through rendering in order to have full control on their visual properties and on the camera movements.

We compared the ranking provided by the participants to that produced by our distance function ϕ. A high correlation between the two rankings indicated that the optical flow played a role in assessing video similarity. The user study web page is available at `http://similarvideos.isti.cnr.it`.

8.1.1 *Tests Description*

The describes how the five tests were designed.

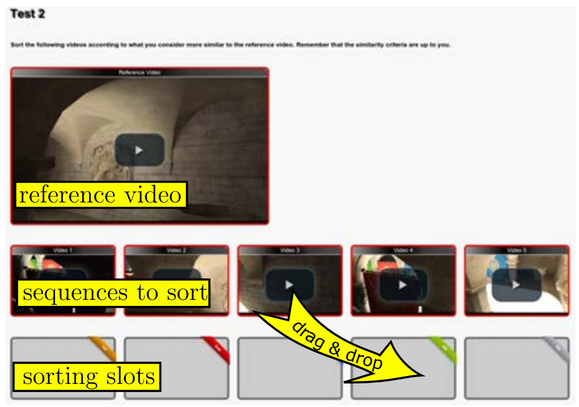
Test 1 - `Buildings1` In this test the reference video is an indoor sequence shot in a building, and all the other sequences are generated with similar camera paths inside a different building.

Test 2 - `Buildings2` This test is similar to the previous one, but different camera paths are used.

Test 3 - `Tree` This test consists of a video shot from a camera orbiting around a tree while the others video sequences were shot with similar camera movements but looking at different subjects. This test is designed to understand whether a user to considers, videos with similar optical flows but different content as similar.

Test 4 - `Lighting` In this test all the videos refer to the same 3D model while camera paths and lighting conditions vary. The 3D model used has a large opening on the top, so we manipulated the lighting conditions by simply using the sunlight as light source and setting a different daytime for each video. Of the five videos, one has the same daytime as the reference one, other two are generated with slightly different daytimes from the reference one, which causes a small visual difference in the projected shadows. Finally, the last two have a greater difference in the daytime settings so that not only the shadows but also the overall brightness of the scene are very different. We modify the path of these videos so that the ranking of the videos according to the lighting differences is the opposite of the ranking according to the optical flow measure ϕ.

Test 5 - `ColorGrading` In this test the videos are characterized by different camera paths but also with different color grading operation (a simple hue-shifting plus contrast changes are applied). The color of the video is modified so that the ranking according to the color difference is the opposite of the ranking according to the flow field distance (more details on this point are provided later).


Test 2

Sort the following videos according to what you consider more similar to the reference video. Remember that the similarity criteria are up to you.

Fig. 5. Using simple drag-and-drop operations, the subject should rank the video sequences by similarity with respect to the reference one. The position of each sequence above the sorting slots is always randomized to avoid possible bias.

In summary, the first two tests consider scenes with a similar semantic content but with different camera movements. The `Tree` test considers a scene with different camera movements and different semantic content, and tests 4 and 5 are used to understand the impact of optical flow w.r.t other visual attributes, i.e different lighting conditions and different color gradings.

Test Name	Kendall τ-c	Spearman ρ
`Buildings1`	0.4177 (0.0000)	0.9 (0.0833)
`Buildings2`	0.5422 (0.0000)	0.9 (0.0833)
`Tree`	0.1289 (0.0187)	0.3 (0.6833)
`Lighting`	0.4045 (0.0000)	0.8 (0.1333)
`ColorGrading`	0.4756 (0.0000)	1.0 (0.0167)

TABLE 1
Data analysis of the results. The numbers in parenthesis are the values of the corresponding Null Hypothesis (rounded to the fourth decimal place). Spearman's correlation coefficient is computed on the corresponding Normalized Rank (provided by the users).

8.1.2 Data Analysis

Our aim was to discover whether users sort the video sequences in the same way as our flow field-based distance function ϕ does. If this is the case, it means that our distance function is successful at capturing the perceived similarity between video sequences. To put this concept into numbers we calculated, for each test, a value typically used in non-parametric statistics for measuring the correlation between two rankings: the Kendall τ-c value. This value expresses the degree of correlation between a group of orderings (provided by the users) and a reference one (provided by our distance) and ranges between -1 and 1. A value of τ-c greater than 0.4 means that the reference order has a good correlation with the orderings provided by the users, while a value greater than 0.6 means that the reference order is strongly correlated with them. A value of 0.2 identifies a weak correlation.

Table 1 reports the Kendall τ-c values obtained. The value in parenthesis is the probability of the null hypothesis, that is, the probability that τ-c value is not significant. For further confirmation of the results obtained, we also calculated, for each test, the Spearman rank correlation coefficient between the *normalized rank* given by the subjects and the ranking provided by the distance ϕ. The normalized rank is the overall rank which takes into account all the user data for the specific test and is obtained by following the normalization procedure indicated by Guilford [23].

8.1.3 Optical flow and semantic

Our approach is built on the intuition that the optical flow of a video is related both to the camera motion and to the scene content. We would thus expect two video shots with a similar optical flow to be in some way perceived as visually similar despite their 3D content. Clearly this is particular true when we treat videos and scenes with similar content while other factors may come into play when the content is very different.

Our results revealed that our approach works very well for all the tests except for the **`Tree`** test. Note that for the **`Buildings1`** and the **`Buildings2`** tests, where the correlation coefficients are very high, the 3D content used is similar but not that similar. In fact the layout, the geometry and the furniture of the building in the reference video is different from the building in the comparison videos. In the **`Tree`** test, the correlation between subjective rankings and the optical flow distance ϕ is low as indicated by Kendall's τ (0.1289). Also the Spearman's ρ is low (and not significant, note the high p-value). This indicates that the optical flow alone is insufficient to provide perceptually similar results in this case, where very different scene content is used.

In conclusion, we can state that the results obtained support our approach although the lack of semantic information may prevent the output video from being perceived as similar to the input one in some cases. However, this does not automatically imply that the user is not happy with the result obtained, in fact our user study did not investigate the satisfaction of the user but only the perceived similarity. Adding semantic information to the 3D model (for example by annotation) and to the input video (for example by using object recognition algorithms) could be useful to enable our pipeline to produce an output video with *the same* semantic content. For example, if the input video contains a shop and we need to present the 3D model of a whole town, the candidate paths that produce videos with a shop may be preferred over others.

8.1.4 Optical Flow vs Other Visual Attributes

Although only two tests of this type were carried out, interesting aspect needs highlighting: the optical flow is a stronger stimulus than lighting or color grading in assessing video similarity.

In the **`ColorGrading`** test, the perceived visual difference caused by the color grading is evaluated by measuring, for each frame, the CIELab distance and taking the mean as a global value. Since the camera paths are set such that the objects observed are more or less always the same in all the frames this measure is a reasonable choice to objectively evaluating the impact of the color grading for each video. Taking this distance into account, we obtain a Kendall τ-c of -0.4045 (with high significance) and a Spearman's ρ of -0.8 w.r.t to the subjective evaluation. Instead, the optical flow predominates over color stimulus with a strong correlation with the human judgements (Kendall's $\tau{-}\mathrm{c} = 0.4045$, Spearman $\rho = 0.8$). These values are exactly the opposite because the videos are generated such that the color grading distance is inversely correlated with the ϕ distance.

We obtained very similar results for the **`Lighting`** test. Videos produced under different lighting conditions were perceived as similar to the reference one thanks to the low differences in the optical flow, that is low values of ϕ. Taking into account that the use of a different time for the daylight simulation leads to very different shadowing effects between the reference video and the videos rated as the most similar, this again demonstrates the importance of the optical flow in evaluating the perceived visual similarity.

8.2 Test Data and Performance

In order to test our approach, we downloaded several models from public repositories (Trimble 3D Warehouse, Archive3D). Table 2 shows a view of these models along with statistics regarding their size, time required to build the database of instant flows, and the corresponding database size. We performed the flow sampling using 16 orientations and 18 directions of displacement. The first layer of our sampling was done inside the bounding box of the scene inflated by a factor 0.3. Inflating the bounding box allows the cameras to be positioned outside the scene as well. As shown in Table3, the size of the database is not related

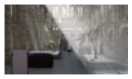
	 Museum	Sibenik	Town	Floorplan	House 1	House 2	House 3
Input							
#polygons	1,468,260	69,853	14,865	5,684,739	38,367	22,026	32,051
Database							
# Instant Flows	3,952,512	2,996,064	2,453,184	3,743,712	1,506,240	2,282,688	1,991,232
Size (GB)	1.37	1.03	0.87	1.29	0.53	0.81	0.70
Proc. Time							
Sampling	2.03	0.19	1.07	7.25	0.12	0.16	0.19
Instant Flows	5.15	3.41	3.01	5.4	1.52	2.45	2.27

TABLE 2
Statistics on the scene pre-processing for the 3D models used in our experiments (timing in hours). Note the compactness of the databases despite the large number of instants flows stored. Processing times are acceptable given the fact that datasets are built only once and can be reused for any video.

to the size of the scene as much to its complexity. This is not surprising, since the more complex the scene the more chances there are for different flows to be create by the sampling. With respect to the database sizes and calculation times, despite the high number of instant flows stored, the databases have a manageable size and their creation time is acceptable given the fact that they are created only once and can be reused for any video.

The list of input sequences used in our tests is given in Table 3. The corresponding results are shown in the accompanying video. The first five entries of the table are various shots taken from YouTube. "The Great Beauty" is the initial sequence of shots of the famous Oscar winning movie by Paolo Sorrentino. Using this kind of movie is an entertaining use of our system. The most practical use is to automatically produce presentations for a portfolio of 3D models. To demonstrate this application, we took three shots of a simple CAD model and then replicated them automatically for three different models (referred to as "House1", "House2" and "House3" in Table 3).

Table 3 also reports the overall processing time for each video and the performances of the main stages of our pipeline. The "Track&Flow" column reports the time needed for the optical flow and the camera path estimations. The "Query - Step HM" and "Query - Step EMD" columns provide details on our two-step query approach (as described in Section 6). Finally, the "Candidates Paths", "Clustering" and "Final Refinement" columns report the processing times for the candidates paths selection, the clustering and the NEWUOA optimization, respectively. The total processing time is reasonable and is dominated by the EMD re-ranking and by the final optimization step. We used 100 clusters with the final refinement applied to the 5 best sequences. NEWUOA optimization is one of the most time-consuming step of our pipeline, however, in most of our tests, this optimization does not significantly improve the final quality. All tests were run on a desktop PC equipped with an Intel Core I7-4820K CPU and an nVidia GeForce GTX 780 GPU.

Finally, for the sake of completeness, we show two examples (see Figure 6 and Figure 7) of input videos and the corresponding output videos as a collection of frames. For a clear evaluation of the quality of the results obtained we refer to the accompanying video.

8.3 Discussion and Limitations

To the best of our knowledge, there are no other approaches that use a video guidance to produce automatic navigation of a scene. So, we give here a qualitative comparison w.r.t other methods instead of a quantitative comparison. Many approaches share broadly the same scheme, i.e.: computing a network of non colliding paths using distance field-like approaches and use them to constrain the camera movement. Their final goal is to control the camera to show the scene as in [2], [3], [4] or to follow a specific target inside a scene [5]. Avoiding collisions and, more generally, keeping the camera away from the scene surface are limitations our approach does not have. In fact, if the input video is a sequence going through a wall, the computed camera path will most likely do the same. By not precomputing paths, but only a dense sampling of instant flows, we have much more flexibility in the camera paths that can be created, and providing an example video is just a way to tell the system what type of path to create. On the other hand, our approach does not guarantee that every part of the scene will be shown in an output video, or that a specific landmark will be seen (although it could be easily constraint the search space to instant flows in the region around a specific region of interest).

The main limitation of our approach is the inability to work with dynamic scenes, where animated characters move and interact. In such cases, it is very difficult to try to obtain a similar flow field in the output sequence. One possible solution is segmenting the computed optical flow (as in [32]) in order to isolate and hence ignore the moving foreground parts, thus limiting the influence of the other moving objects. Another issue may arise from the processing of the input video; the optical flow may be difficult to estimate reliably in the case of large textureless areas. For the same reason, estimating the camera path may fail or provide poor results. Finally, the intrinsic camera parameters are defined at scene sampling time. While this limitation may be overridden by extending the sampling to cover a range of intrinsic parameters, the dynamic combination of camera movement and focal change would add further uncertainty in terms of camera tracking and finding candidate solutions.

Video Sequences								
Name	Length	Track&Flow	Query - Step HM	Query - Step EMD	Candidates Paths	Clustering	Final Refinement	Total
Tree	1.33	30.15	3.23	154.98	12.77	1.81	116.23	319.17
Showreel	3.84	84.25	14.25	392.54	55.98	4.55	272.17	823.74
Documentary	5.07	104.21	16.05	452.40	71.30	3.22	442.29	1,089.47
Ambulance	5.07	104.13	14.23	530.25	66.23	2.92	369.21	1,086.97
Villa	20.75	417.23	83.41	1,846.80	296.07	26.14	1,813.69	4,483.34
Great Beauty	17.07	363.87	72.02	1,252.68	224.38	13.25	1,459.44	3,385.64
House1	17.57	n/a	24.75	1,448.43	143.47	15.21	1,179.21	2,811.07
House2	17.57	n/a	28.35	1,448.43	121.55	18.80	1,165.81	2,782.86
House3	17.57	n/a	26.21	1,448.43	156.22	17.51	1,181.97	2,830.34

TABLE 3
Processing times for the video sequences (timing in seconds). The table also shows the performance details for the main stages of our algorithm. For the videos House1, House2 and House3 there is no camera track estimation nor optical flow calculation as the input is a known virtual camera path designed on a different CAD model.

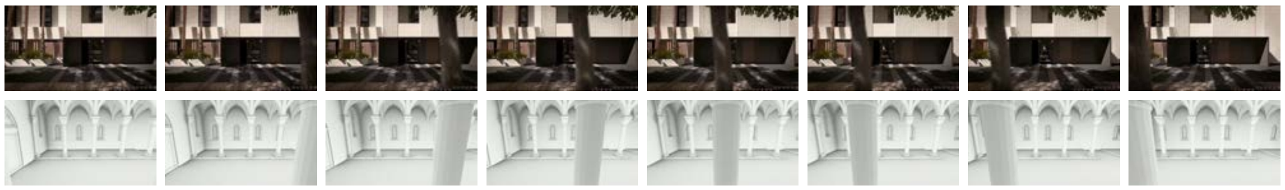


Fig. 6. Output video (2nd row) produced using the 3D model Sibenik and the input video shown in the first row. The algorithm is able to reproduce the moment of passage of the tree by replacing it with a column of the 3D model.

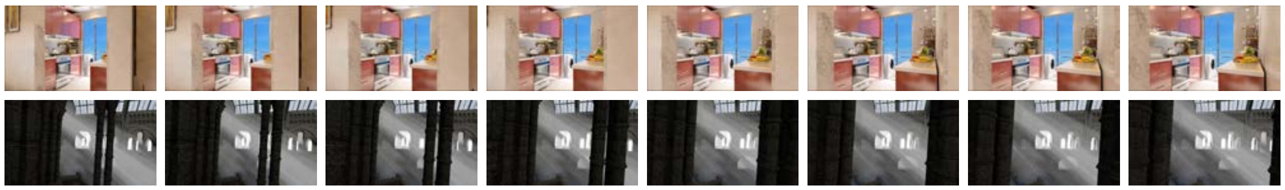


Fig. 7. Output video (2nd row) produced using the 3D model Museum and the input video shown in the first row. The algorithm maps a video of a camera entering a door to a path where the virtual camera enters an arch of the 3D model.

9 Conclusions

We have proposed a new approach to the problem of unassisted presentation of 3D scenes. Our algorithm produces a video which presents a static scene using another video as reference. We build on the idea of finding the camera path (or paths) in the 3D scene with the most similar optical flow as the input video. The processing steps of our pipeline have been finely tuned and the results produced are promising.

We presented a user study that enables us to assess the impact of the optical flow in perceiving two different video sequences as similar. This is still a early result and the results obtained are very interesting. We think that more exhaustive and complex subjective tests about video similarity and visual preferences in video production would be of great interest for the video editing/processing community.

Our algorithm could be further improved by transferring other visual attributes from the input to the output video, such as lighting and/or the colour grading in order to better reproduce the overall "mood" of the input video. Aside from the complete algorithm itself, our contribution includes an efficient strategy to index and retrieve optical flows. It would be very interesting to investigate the performance of the proposed optical flow descriptor and the related index and retrieve approach in different application contexts, for example for near duplicate video detection or for content-based video retrieval.

References

[1] L.-w. He, M. F. Cohen, and D. H. Salesin, "The virtual cinematographer: A paradigm for automatic real-time camera control and directing," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 217–224.

[2] C. Andújar, P. Vázquez, and M. Fairén, "Way-Finder: guided tours through complex walkthrough models," *Computer Graphics Forum*, vol. 23, no. 3, pp. 499–508, 2004.

[3] B. Salomon, M. Garber, M. C. Lin, and D. Manocha, "Interactive navigation in complex environments using path planning," in *Proceedings of the 2003 Symposium on Interactive 3D Graphics (I3D'03)*. New York, NY, USA: ACM, 2003, pp. 41–50.

[4] M. Di Benedetto, F. Ganovelli, M. B. Rodriguez, A. J. Villanueva, R. Scopigno, and E. Gobbetti, "ExploreMaps: Efficient Construction and Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments," *Computer Graphics Forum*, 2014.

[5] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross, "Visibility transition planning for dynamic camera control," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'09)*. New York, NY, USA: ACM, 2009, pp. 55–65.

[6] A. Secord, J. Lu, A. Finkelstein, M. Singh, and A. Nealen, "Perceptual models of viewpoint preference," *ACM Transactions on Graphics*, vol. 30, no. 5, Oct. 2011.

[7] P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich, "Viewpoint Selection using Viewpoint Entropy." in *Proceedings of the Vision Modeling and Visualization Conference (VMV2001)*, 2001.

[8] D. Plemenos, J. Grasset, B. Jaubert, and K. Tamine, "Intelligent visibility-based 3D scene processing techniques for computer games," in *Proc. of GraphiCon 2005 - International Conference on Computer Graphics and Vision*, 2005.

[9] C. Lee, A. Varshney, and D. Jacobs, "Mesh saliency," in *ACM SIGGRAPH 2005 Papers*, vol. 24, no. 3, Jul. 2005, pp. 659–666.

[10] K. Kardan and H. Casanova, "Virtual cinematography of group scenes using hierarchical lines of actions," in *Proceedings of the 2008 ACM SIGGRAPH Symposium on Video Games (Sandbox'08)*. New York, NY, USA: ACM, 2008, pp. 171–178.

[11] K. C. Zheng, A. Colburn, A. Agarwala, M. Agrawala, D. Salesin, B. Curless, and M. F. Cohen, "Parallax photography: Creating 3d cinematic effects from stills," in *Proceedings of Graphics Interface 2009 (GI'09)*. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2009, pp. 111–118.

[12] M. Cherubini, R. de Oliveira, and N. Oliver, "Understanding near-duplicate videos: A user-centric approach," in *Proceedings of the 17th ACM International Conference on Multimedia (MM'09)*. New York, NY, USA: ACM, 2009, pp. 35–44.

[13] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 6, pp. 797–819, Nov 2011.

[14] R. Fablet, P. Bouthemy, and P. Perez, "Nonparametric motion characterization using causal probabilistic models for video indexing and retrieval," *Image Processing, IEEE Transactions on*, vol. 11, no. 4, pp. 393–407, Apr 2002.

[15] Y.-F. Ma and H.-J. Zhang, "Motion texture: a new motion based video representation," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2, 2002, pp. 548–551.

[16] J. Liu, Z. Huang, H. Cai, H. T. Shen, C. W. Ngo, and W. Wang, "Near-duplicate video retrieval: Current research and future trends," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 44:1–44:23, Aug. 2013.

[17] A. Hampapur, K. Hyun, and R. M. Bolle, "Comparison of sequence matching techniques for video copy detection," in *Proc. SPIE 4676, Storage and Retrieval for Media Databases 2002*, 2001, pp. 194–201.

[18] T. C. Hoad and J. Zobel, "Fast video matching with signature alignment," in *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR'03)*. New York, NY, USA: ACM, 2003, pp. 262–269.

[19] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vision*, vol. 92, no. 1, pp. 1–31, Mar. 2011.

[20] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. of the 8th European Conference on Computer Vision (ECCV2004)*, vol. 3024, 2004, pp. 25–36.

[21] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth movers distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, p. 2000, 2000.

[22] H. Ling and K. Okada, "Emd-l 1: an efficient and robust algorithm for comparing histogram-based descriptors," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 330–343.

[23] J.P.Guilford, *Psychometric Methods*. McGraw-Hill, 1954.

[24] K. White, D. Cline, and P. Egbert, "Poisson disk point sets by hierarchical dart throwing," *Symposium on Interactive Ray Tracing*, pp. 129–132, 2007.

[25] M. Muja and D. G. Lowe, "Fast matching of binary features," in *Proceedings of the Ninth Conference on Computer and Robot Vision (CRV'12)*. Washington, DC, USA: IEEE Computer Society, 2012, pp. 404–410.

[26] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, 2014.

[27] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *European Conference on Computer Vision*, ser. LNCS, A. Z. David Forsyth, Philip Torr, Ed., vol. I. Springer, oct 2008, pp. 304–317.

[28] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[29] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[30] M. Yan, "Methods of determining the number of clusters in a data set and a new clustering criterion," Ph.D. dissertation, Virginia Tech, 2005.

[31] A. Zaslavski and M. Powell, "The NEWUOA software for unconstrained optimization without derivatives," in *Large-Scale Nonlinear Optimization*, ser. Nonconvex Optimization and Its Applications, P. Pardalos, G. Pillo, and M. Roma, Eds. Boston: Springer US, 2006, vol. 83, ch. 16, pp. 255–297. [Online]. Available: http://dx.doi.org/10.1007/0-387-30065-1_16

[32] J. Lezama, K. Alahari, J. Sivic, and I. Laptev, "Track to the future: Spatio-temporal video segmentation with long-range motion cues," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2011)*, June 2011, pp. 3369–3376.

Andrea Baldacci received a degree in Information Engineering from University of Pisa in 2012. In 2013 he joined the Visual Computing Laboratory, ISTI-CNR, Pisa, as a PhD student of the University of Pisa. His research interest are in the field of Computer Graphics and Computer Vision and includes image-based reconstruction algorithm, real-time rendering, computational geometry and stereo image processing.

Fabio Ganovelli is a Researcher at the Visual Computing Laboratory of the Istituto Scienza e Tecnologie dell'Informazione (ISTI-CNR) in Pisa. His research interests are in the field of Computer Graphics and Computer Vision. He worked in several topics including deformable objects, multiresolution rendering techniques, photorealistic rendering and geometry processing and education.

Massimiliano Corsini received a PhD degree in Information and Telecommunication Engineering from the University of Florence. Currently, he is a Researcher at the Visual Computing Laboratory of the ISTI-CNR in Pisa, Italy. His research interests are in the fields of Computer Graphics, Computer Vision and Image Processing and include 3D watermarking, perceptual metrics, visual appearance acquisition and modelling, registration algorithms and image-based relighting. He published more than 50 papers in peer-reviewed international conferences and journals. He also collaborated in several National and International projects and served on numerous program committees.

Roberto Scopigno is a Research Director with CNR-ISTI and leads the Visual Computing Lab. He graduated in Computer Science at the University of Pisa in 1984. He is engaged in research projects concerned with 3D scanning, surface reconstruction, multiresolution, scientific visualization, and cultural heritage. He published more than hundred eighty papers in international refereed journals/conferences, with H-index 34. Roberto has been responsible person for CNR-ISTI in several EU projects and co-chaired several international conferences. He is member of the Eurographics Association, was awarded the EG "Outstanding Technical Contribution Award" in 2008, served as Chair of the Eurographics Association (2009-2010), Co-Editor in Chief of the Computer Graphics Forum journal (2001-2010) and is currently member of the Editorial Board of the ACM J. on Computing and Cultural Heritage.