Photo Sequencing

Tali Dekel (Basha) · Yael Moses · Shai Avidan

Received: 18 June 2013 / Accepted: 3 March 2014

Abstract A group of people taking pictures of a dynamic event with their mobile phones is a popular sight. The set of still images obtained this way is rich in dynamic content but lacks accurate temporal information. We propose a method for *photo-sequencing*—temporally ordering a set of still images taken asynchronously by a set of uncalibrated cameras. Photo-sequencing is an essential tool in analyzing (or visualizing) a dynamic scene captured by still images. The first step of the method detects sets of corresponding static and dynamic feature points across images. The static features are used to determine the epipolar geometry between pairs of images, and each dynamic feature votes for the temporal order of the images in which it appears. The partial orders provided by the dynamic features are not necessarily consistent, and we use rank aggregation to combine them into a globally consistent temporal order of images. We demonstrate successful photo-sequencing on several challenging collections of images taken using a number of mobile phones.

Communicated by Carlo Colombo.

An earlier version of part of this work was published in ECCV 2012 Dekel (Basha) et al. (2012).

T. Dekel (Basha) (✉) · S. Avidan
School of Electrical Engineering, Tel Aviv University,
69978 Tel Aviv, Israel
e-mail: talib@eng.tau.ac.il

Y. Moses
Efi Arazi School of Computer Science, The Interdisciplinary Center,
46150 Herzliya, Israel

1 Introduction

Dynamic events such as family gatherings, concerts or sports events are often captured by a group of people, using what is probably the most popular means of photography today – smartphones. The set of photos, often taken within a short time interval of one of the event highlights, can be regarded as the output of a new type of extended camera, which we call a *crowd-based camera*, or *CrowdCam* for short (see Fig. 1). This setup differs substantially from traditional multi-camera systems that are usually calibrated and synchronized. CrowdCam is operated by multiple freely moving, uncooperative photographers, and there is no coordination in the capturing time. The question that motivates our study is whether it is possible to explore, visualize and analyze the *dynamic* content of the scene using the set of still images obtained by a CrowdCam.

Essential to this purpose is recovering the temporal order of the images, as demonstrated by many computer vision methods that use one or more video sequences. Clearly, temporal order is available when all images are taken from the

Fig. 1 CrowdCam: A group of people, holding cellphone cameras, capturing the highlights of a dynamic event

Published online: 03 April 2014

Fig. 2 The Photo-Sequencing Problem: recovering the temporal order of N still images of a dynamic event, taken from different locations at different time steps

same camera as in a video sequence. Alternatively, when two or more sequences are taken, temporal order can be recovered using video synchronization methods.

We propose a method for recovering the temporal order of a set of still images of an event taken at roughly the same time, and term this problem *photo-sequencing* (see Fig. 2). Our method takes as input a set of still images captured by a CrowdCam and returns a globally consistent temporal order for all images. Video synchronization methods are not applicable here because each camera may provide only one still image. In our case, the only temporal information available is given by the phones' clocks, which we show to be insufficiently accurate (see Sect. 4.1). More accurate temporal information can be obtained from other devices, e.g., GPS. However, their data may not be available in indoor scenes or may be blocked by the photographer for reasons of privacy issues. Moreover, a vision based solution is of scientific value in itself.

Photo-sequencing can be solved directly if the 3D structure of the dynamic scene is known. However, recovering the 3D structure of a dynamic scene often requires camera calibration, prior knowledge about the 3D structure or the motion of objects, and a very large number of images, which we do not assume to have.

Our goal is to compute photo-sequencing without recovering the 3D structure of the scene. To do so, we assume that at least two images are taken from roughly the same location by the same camera. This assumption is reasonable because people often take more than one image of an interesting moment in the event, without moving much. We further assume that within the short time interval, there are enough features that move approximately along straight lines. This assumption is needed to model the problem but in practice points can deviate considerably from the linear motion model.

1.1 Algorithm Outline

Consider a 3D scene point moving along a linear trajectory and captured by a set of cameras, at different time steps. Imagine sampling the 3D locations along the point trajectory, at the same time steps at which it was captured by the set of cameras. The spatial order of these 3D locations along the trajectory implicitly induces the *temporal* order of the images (see Fig. 3a). Our method avoids recovering the sampled 3D locations of a dynamic scene point by computing its projections on the reference image (see Fig. 3b).

All analysis is done in 2D. We use the static feature points to compute the fundamental matrix between every image and the reference image and map all dynamic features to their epipolar lines in the reference image. In addition, we use the two reference images (taken from the same position) to compute the 2D projection of the trajectory of each dynamic point. The intersection of the epipolar lines with the projection of the trajectory line gives the spatial order of the scene point and provides the partial time order for a subset of the images.

The resulting partial orders computed from each dynamic feature are not necessarily consistent because of matching errors, large deviation of some of the features from linear motion, and noise. One of the challenging problems is to compute a *full temporal order*, i.e., an order of all input images that is as consistent as possible with the computed partial orders. This problem is known as the *rank aggregation* problem, which is known to be NP-hard for the most general case. We first rely on geometric constraints to clean up the data, and construct a directed graph that represents the pairwise temporal orders defined by the dynamic features. If the graph contains no loops (i.e., it is a DAG), then finding a Hamiltonian path is reduced to computing a topological sort, for which a simple polynomial algorithm exists. However, if the graph contains cycles, finding a Hamiltonian path becomes NP-hard. Therefore, we adopt a rank aggregation Markov chain approximation to solve it.

Possible applications of photo-sequencing include visualizing and analyzing dynamic content from a set of still images. A temporally coherent presentation of a set of images taken from different viewpoints or time instances of a given event is one example. It may also be used to generalize various computer vision tasks, such as tracking or segmentation, to work with a set of still images instead of video. These applications, which are beyond the scope of this paper, arise naturally when people share their images and are willing to extract the most out of them.

1.2 Contributions

Our method offers a solution to the novel photo-sequencing problem—recovering the temporal order of a set of static images of a dynamic event. The method's robustness is based on a rank aggregation algorithm that aggregates noisy measurements to overcome inconsistencies.

In addition, we present a theoretical analysis of the minimal required set of images for recovering 3D trajectory from the 2D projection of a linear moving point. The analysis is performed under the assumptions used for photo-sequencing. We show the tradeoff between the number of required images and the time stamp information if available (see Sect. 3.4).

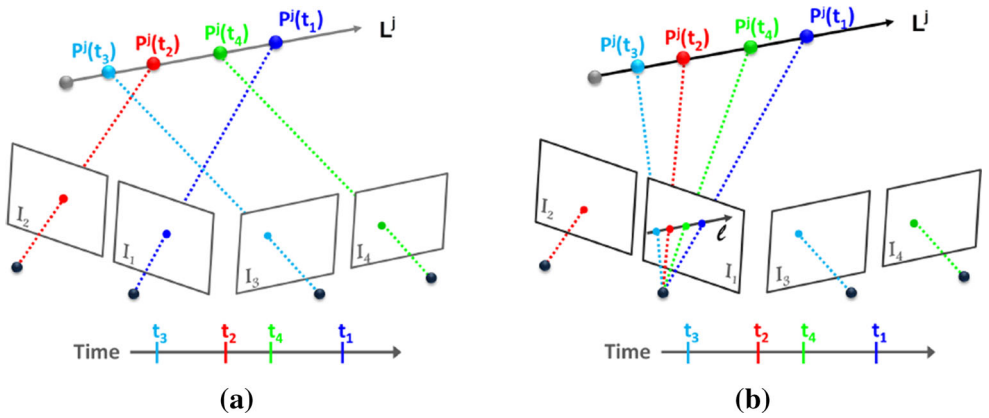

Fig. 3 (**a**) The order of the sampled location of a 3D point moving along a *straight line* implicitly induces the order of the corresponding images: $t(I_3) < t(I_1) < t(I_4) < t(I_2)$. (**b**) The same order is also defined along the projection of the *line* to a reference image

2 Related Work

2.1 Video Synchronization

Temporal alignment of visual data has been studied extensively in the context of video synchronization. Synchronization methods for aligning a pair of sequences include correlating motion signatures computed from a set of successive frames (e.g., Dexter et al. 2009), aligning tracked trajectories of features or objects visible in both videos (e.g., Caspi and Irani 2002;Tresadern and Reid 2003; Whitehead and Laganiere 2005), aligning all the frames (e.g., Caspi and Irani 2002; Sand and Teller 2004), assuming linear combination between object views under orthographic projection (e.g.,Wolf and Zomet 2002), assuming low rank for non-rigid moving objects (e.g., Zelnik-Manor and Irani 2003), and using tri-focal tensor-based relations when at least 3 videos are considered for spatial matching of points or lines (e.g.,Lei and Yang 2006).

Other synchronization methods attempt to bypass the computation of spatial correspondence in these methods, by using spatio-temporal feature statistics (e.g., Yan and Pollefeys 2004), or temporal signals defined over corresponding epipolar lines Pundik and Moses (2010). Geometric constraints were used for aligning multiple video sequences (e.g., Pádua et al. 2010) or for achieving sub-frame accuracy (e.g., Meyer et al. 2008). In both methods the intersections of the trajectory of dynamic features with the epipolar lines of corresponding features in the other images were used to define order.

None of these methods consider the inconsistent ordering that may be caused by different choices of features (e.g., matching trajectories). Moreover, all the above methods use successive frames in each of the videos in order to compute the synchronization. However, we assume here that the cameras might provide only a single image. Such synchronization methods are not applicable for temporal ordering of the images considered in this paper.

2.2 Structure Reconstruction from Still Images

Several methods address the problem of non-rigid shape and motion recovery from a set of still images when temporal order is not used directly. These methods assume that point correspondences are given and the motion of the objects is restricted. A method for reconstructing the 3D coordinates of a point moving along a straight line and captured by a moving camera was proposed in Avidan and Shashua (2000). They use trajectory triangulation and show that a linear solution exists if the camera parameters are known. If the camera parameters are unknown, it is still possible to reconstruct the 3D coordinates of points moving on planes Shashua and Wolf (2000). Trajectory triangulation was later extended using polynomial representations Kaminski and Teicher (2004). Park et al. (2010) proposed a method for reconstructing the 3D trajectory of a moving point from its correspondence in a collection of 2D perspective images. However, they assume that 3D spatial pose and time of capture are given. In our case, it is sufficient to have each feature matched in only 4, instead of 5 images, and a weaker calibration is required; that is, only the fundamental matrix between each image and the reference image are needed. Moreover, our method offers a way to overcome the inconsistent ordering obtained by different features and allows us to deviate from the linear motion assumption.

A different class of methods use factorization to deal with dynamic scenes Torresani et al (2008). These methods assume that the correspondence between features in a large number of images can be obtained. Furthermore, they assume that the deformation of a 3D shape can be represented by a linear combination of shape-bases, which often restricts the number of independently moving objects. Hence, by increas-

ing the rank of the observation matrix, the non-rigid components of motion are captured by additional eigenvectors. This was later extended by Bartol et al. (2008), Llado et al. (2005). Indeed the solutions to these methods may result in photo-sequencing. However, they are limited to restricted scenes and unlike our method, require features to be matched across a large number of images.

2.3 Image and Video Collections

Large numbers of images uploaded to the Internet are used for various applications such as 3D reconstruction, visualization, and recognition of static scenes (see review by Snavely et al. 2010). This is often referred to as *community photography* and assumes that images are co-located in space but not necessarily in time. Therefore, only the static regions of the scene are considered. In our case, the set of still images are co-located both in space and in time, and we focus on extracting the temporal information. We believe that future work will combine photo-sequencing with community photography leading to new ways to analyze the dynamic regions of the scenes.

A method for temporally aligning still images that span many years was suggested by Schindler and Dellaert (2010). Their solution is based on analyzing changes and occlusions of the viewed 3D static scene. Our method deals with a dynamic scene captured in a short time interval and is based on motion.

Recently, a method for navigating in a collection of videos of a dynamic scene, e.g., a music performance, was proposed by Ballan et al. (2010), but they use a collection of casually captured videos rather than still images as we do.

2.4 Rank Aggregation

Rank aggregation is the problem of obtaining a full ordering of elements (alternatives) given inconsistent partial orderings of the same elements from different sources. The rank aggregation problem was originally studied in the context of social choice theory and voting theory Young and Levenglick (1978); Young (1988). However, in recent years this problem has been of interest in various disciplines in the computer science community, for example, biological applications Shili (2010), and Web applications such as meta-search, page ranking, and spam detection Dwork et al. (2001), Schalekamp and van Zuylen (2009), Elena and Straccia (2003).

Due to the large and diverse body of work on the rank aggregation problem, we will focus here on the work of Dwork et al. (2001), who proposed a Markov chain approach for Web applications, which we adopt for photo-sequencing. They proposed several heuristic algorithms, and compared them to a number of traditional rank aggregation methods such as Footrule aggregation and Borda's method. While there is no guarantee on the quality of the output, the Markov chain approach is extremely efficient, and was shown to usually match or outperform the other methods. The basic idea is to construct a Markov chain, in which the states are the elements to be ordered. The stationary distribution of a state ranked high will have a larger probability than of a state ranked low. Hence, the stationary distribution will determine the aggregate rankings of the items (see Sect. 3.2 for further details). To the best of our knowledge, rank aggregation was used in computer vision only for Content-Based Image Retrieval Jegou et al. (2010), Pedronette and Torres (2011). We believe that rank aggregation is a powerful tool that can be adopted for other computer vision tasks.

Algorithm 1 Photo Sequencing Alg.

Input: N images, $\{I_k\}_{k=1}^{N}$ taken by the set of cameras. I_1 is the reference.
Output: A permutation, $\sigma : \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$.

```
[f_1, f_2] ← Match(I_1,I_2);
[f_D1, f_S1, f_D2, f_S2] = Classify_Dyn_Stat_Ref(f_1, f_2)
for each I_k and k = 3 to N do
    [f_1, f_k] ← Match(I_1,I_k);
    [f_Dk, f_Sk] = Classify_Dyn_Stat(f_S1, f_D1, f_k)
    F_k =ComputeFundamentalMat(f_S1, f_Sk).
end for
for each dynamic feature p_1^i ∈ f_1 do
    l̂^i = p̂_1^i × p̂_2^i        {l̂^i is the image line}
    for each p_k^i(t_k) ∈ S^i do
        l̂_k^i = F_k p̂_k^i(t_k)      {l̂^i is the image line}
        p̂_1^i(t_k) = l̂^i × l̂_k^i    {p_1^i(t_k) is the intersection point}
        α_k ← ComputeAlpha(p_1^i, p_2^i, p_1^i(t_k))
    end for
    σ_i ← sort({α_k | k ∈ n_i}).
end for
Full order: σ ← RankAggregation(σ_1, σ_2, ...):
    V ← VotingMatrix(σ_1, σ_2, ...).
    M ← TransitivityMatrix(V).
    σ ← MarkovChainRanking(M).
```

3 The Method

Consider a dynamic scene captured by N images $\{I_k\}_{k=1}^{N}$, taken at different time steps within a small time interval. Our goal is to determine the temporal order in which the images were taken. This is equivalent to finding a permutation on the image indices, $\sigma : \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}$, such that

$$t(I_{\sigma^{-1}(1)}) \leq t(I_{\sigma^{-1}(2)}) \ldots \leq t(I_{\sigma^{-1}(N)}), \tag{1}$$

where $t(I_k)$ is the time at which image I_k was taken, $\sigma(i)$ indicates the temporal rank of image i, and σ^{-1} is the inverse mapping. We assume that two of the images are taken from

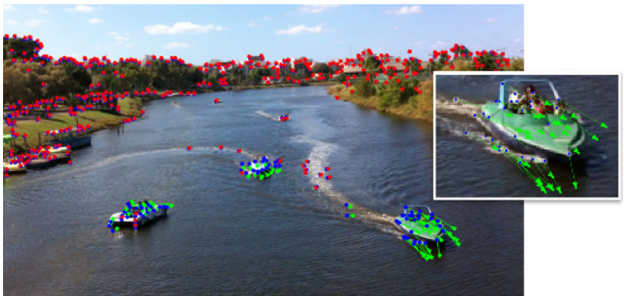

Fig. 4 Static & dynamic features: the detected corresponding features of I_1 and I_2 (taken from roughly the same position) are marked over the reference image, I_1; *blue* are I_1 features (static and dynamic); *red* are the corresponding static features of I_2, and *green* are the corresponding dynamic features of I_2

approximately the same position. Without loss of generality, let I_1 and I_2 be these images, and I_1 be the reference image.

3.1 Temporal Order Voting

We first extract and match features from all input images (see Sect. 3.3.1 for details). The detected features are classified into *static* and *dynamic* features (i.e., projections of static or dynamic scene points, respectively). To do so, we classify the matched features in I_1 and I_2. In these images, each pair of matched static features should be approximately in the same location. Thus, the static features are easily detected by thresholding the Euclidean distance between matched features, and the dynamic features are the remaining ones. An example of the classified features is shown in Fig. 4.

We then match features between the reference image, I_1, and each image, I_k. The static and dynamic features in I_k are taken to be those that are matched to the static and dynamic features of I_1, respectively. The static features are used to compute the fundamental matrix, F_k, between I_1 and I_k (we use the BEEM algorithm Goshen and Shimshoni 2006); the dynamic features are used to determine the temporal order of the images, as explained next.

3.1.1 Ordering by a Single Set of Dynamic Features

Let $p_1^i \in I_1$ be a dynamic feature in the reference image (homogeneous coordinates), and S^i be the set of its corresponding features in a subset of the images. Let $n_i \subseteq \{1, ..., N\}$ be the indices of this subset. The set S^i consists of the projections of a scene point P^i at different time steps. For simplicity, we assume that P^i moves along a line L^i, and its projection to the reference image is given by a 2D line ℓ^i (see Fig. 3a, b). The set of features S^i defines a set of epipolar lines in I_1 which intersect the line ℓ^i (see Fig. 5b). The spatial order of these intersection determines the temporal order, σ_i, of the corresponding images, since it is identical to the spatial order of the 3D positions of P^i along L^i.

Formally, let $P^i(t_k)$ denote the 3D position of P^i at t_k, the time image I_k was captured. The feature set S^i is given by: $S^i = \{p_k^i(t_k) \mid k \in n_i\}$. From the matching process, we have direct access to the projection of $P^i(t_k)$ to image I_k, namely $p_k^i(t_k)$.

Our goal now is to compute the projections of the set of points $P^i(t_k)$ onto the reference, I_1. That is, we compute $p_1^i(t_k)$ for each $k \in n_i$. Note that $p_1^i(t_2) \approx p_2^i(t_2)$ since both points were captured roughly from the same position. Hence the line ℓ^i is defined by the two corresponding points, $p_1^i(t_1)$ and $p_2^i(t_2)$. That is, $\ell^i = p_1^i(t_1) \times p_2^i(t_2)$. The point $p_1^i(t_k)$ is given by the intersection of the line ℓ^i and the corresponding epipolar line: $\ell_k^i = F_k p_k^i(t_k)$ (see Fig. 5b). Putting it all together, we have:

$$p_1^i(t_k) = \ell^i \times \ell_k{}^i = (p_1^i(t_1) \times p_2^i(t_2)) \times \left(F_k\ p_k^i(t_k)\right). \quad (2)$$

This equation degenerates if the epipolar line, ℓ_k^i, coincides with ℓ^i, i.e., the feature moves along the epipolar line. We

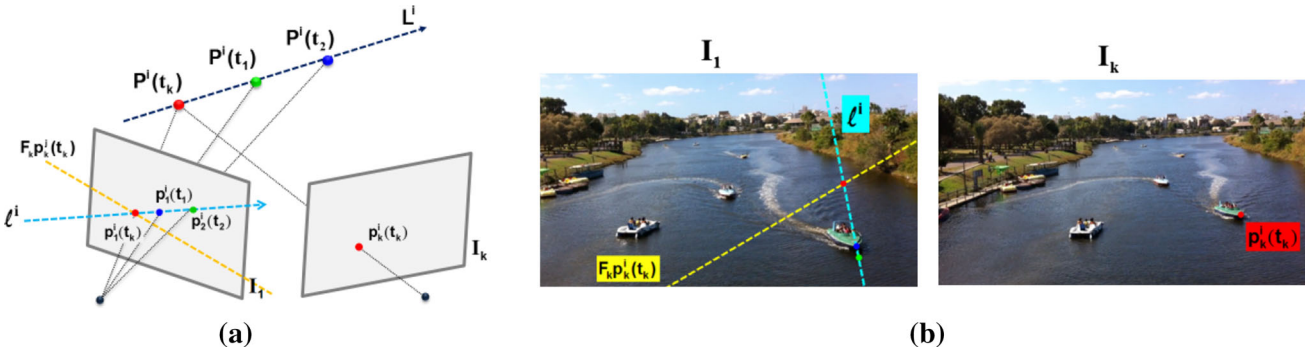

Fig. 5 (**a**) The projection of the trajectory, L^i, of the point P^i, forms the line ℓ^i on image I_1. The feature points $p_1^i(t_1)$, $p_2^i(t_2)$, in image I_1, and $p_k^i(t_k)$ in image I_k, are corresponding dynamic features. The line ℓ^i intersects the *epipolar line* (in *yellow*), which corresponds to p_k^i. The intersection point, $p_1^i(t_k)$, is the projection of P^i onto I_1 at time step t_k. The spatial order of $p_1^i(t_1)$, $p_2^i(t_2)$, and $p_1^i(t_k)$, along ℓ^i, defines the temporal order between I_1, I_2 and I_k. (**b**) The computation on real images: the projected trajectory, ℓ^i, in cyan; the *epipolar line* in *yellow*; the intersection in *red*

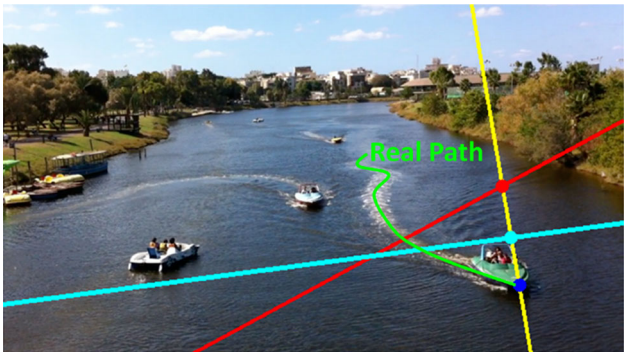

Fig. 6 Linear Motion Assumptions: In *green*, the real path of the green boat; in yellow, the approximated 2D image line. The epipolar lines intersect both the real path and the 2D image line. The spatial order of both sets of intersections is the same

detect this case by the angle between the two lines and remove such points from further processing.

Finally, the spatial order of the mapped features along the line ℓ^i is computed. Formally, the intersection point, $p_1^i(t_k)$, can be represented by:

$$p_1^i(t_k) = p_1^i(t_1) + \alpha_k(p_2^i(t_2) - p_1^i(t_1)). \tag{3}$$

The computed temporal order, represented by a permutation, σ_i, is obtained by sorting the computed values $\{\alpha_k \mid k \in n_i\}$.

It is worth noting that instead of recovering the 3D structure of the dynamic scene, we perform all calculations in the image plane of the reference image I_1. This allows us to match features in a smaller number of images than required for full 3D reconstruction, use weaker calibration (fundamental matrices with respect to the reference image instead of all pairs of images), and clearly avoid additional noise that may be introduced into the 3D reconstruction. As it turns out, the algorithm described above is still applicable when the linear motion assumption is violated. This is because all we care about is the spatial *order* of the intersections of the epipolar lines and the trajectory and not their actual locations. This is exemplified in Fig. 6, where the actual path of the green both was very far from the approximated linear line, yet the order of the intersection points did not change. Note that if the order is not preserved, then an incorrect order is produced by this feature. However, since the information from all features is aggregated, it is expected that a few such errors will not affect the result. This is indeed demonstrated by our experiments. Finally, our method will not work if all epipolar lines are parallel and the object moves along the epipolar line direction. In this case, the intersections of the epipolar lines with the projection of the real trajectory is expected to be noisy.

3.2 A Full Temporal Order

Each dynamic feature defines a set S^i that give rise to a partial temporal order, σ_i, of a subset n_i of the images. These partial orders are often conflicting due to noise and matching errors. Such errors are unavoidable in practice. (For example, in one of our experiments only 23 out of 67 features agreed with the correct order, and none of them produced a full order of the set). Our goal is to compute a full order, σ, that is consistent, as much as possible, with the partial orders. This problem is known as the *rank aggregation problem*, and has been studied mostly in the areas of social choice and voting theory. Aggregation of partial temporal orders must be performed even if the 3D locations of each feature are fully recovered.

3.2.1 Objective

Formally, the widely accepted objective to minimize in rank aggregation is the number of pairwise disagreements between the full order, σ, and each of the input permutations, σ_i. Specifically, the distance between σ and an input permutation, σ_i, is measured by the *Kendall distance*.

$$K(\sigma, \sigma_i) = \\ |\{(l, m) \mid l, m \in n_i,\ \sigma(l) < \sigma(m),\ \sigma_i(l) > \sigma_i(m)\}|. \tag{4}$$

Therefore, our objective is to find σ^* such that:

$$\sigma^* = \operatorname*{argmin}_{\sigma} \sum_{i}^{N_D} K(\sigma, \sigma_i), \tag{5}$$

where N_D is the number of detected dynamic feature sets.

Minimizing this objective function, known as the *Kemeny optimal aggregation*, was proven to be NP-hard in the number of images, even when the number of input permutations (feature sets) is only four Dwork et al. (2001). We adopt, as we next describe, the Markov chain approximation of Dwork et al. (2001), which was shown to work well in Web ranking applications.

3.2.2 Graph Representation

Let $G = (V, E)$ be a weighted graph where the nodes in V correspond to the N images to be ordered, and the weight of an edge $(i, j) \in E$ corresponds to the probability that image I_i was captured before image I_j, $Pr(t(I_i) < t(I_j))$.

If all partial orders are consistent with each other then G is a DAG. In this case the problem reduces to a topological sort that can be found in polynomial time. (A topological sort is finding a linear ordering of the vertices such that, for every edge (i, j), i comes before j in the ordering.) In addition, if the set of partial orders defines a complete order, then the topological sort results in a Hamiltonian path. It can be easily shown that the order defined by this path is optimal with respect to the Kendall distance (Eq. 5).

In reality, the partial orders are not consistent, and G will contain cycles, (e.g., Fig. 10). Unfortunately, it is impossible

to compute a topological sort in this case. One alternative is to find and remove cycles (i.e., edges) from the graph before running the topological sort. However, efficiently choosing which edges to remove is non-trivial. A well-established alternative is to treat the graph G as a Markov chain system, that is, a memoryless random process that moves at each time step from one state to another.

3.2.3 Rank Aggregation by Markov Chain

A Markov chain system is defined by a set of states, and a (non-negative) transition matrix $\mathbf{M}$ that specifies the probabilities of moving from one state to another. In our case, the states correspond to graph nodes (i.e., images) and the transitions between states correspond to edge weights: $\mathbf{M}_{ij} = Pr(t(I_i) < t(I_j))$. We compute these probabilities using the computed partial orders, $\sigma_1, \sigma_2, \ldots, \sigma_{N_D}$ (see 3.3.2 for details).

Let us consider a random walk on this chain (graph), where the first state is chosen according to a uniform distribution across all states (nodes). If the chain describes a consistent full order between the images (G is a DAG), the walk will end in the steady state at an absorbing state (the graph sink), i.e., the state that corresponds to the image captured last. If there is no sink in G, then the random walk will end in a strongly connected component of G. (In a *strongly connected* subgraph, a directed path exists between each pair of nodes.) This connected component, which we name *sink-component*, contains the state corresponding to the last captured image.

The sink or the sink-component can be computed using eigenvectors in the following way. Formally, let $\mathbf{x}$ be a $N \times 1$ vector that describes the probabilities of being at each of the states (images). Then $\mathbf{Mx}$ will be the probability distribution over the states in the next time step, and after k steps the distribution will be given by $\mathbf{M}^k\mathbf{x}$. A random walk on this graph, with an initial uniform distribution $\mathbf{x}$, will converge to the eigenvector $\mathbf{y} = \mathbf{My}$. The eigenvector can be computed directly or using power iterations. If node i is a sink of G (and not part of a cycle), then the i entry of $\mathbf{y}$ equals 1, and the remaining entries equal zero. If G has a sink-component, then $\mathbf{y}$ will have non-zero entries only in the sink-component nodes.

Given this analysis, we run power iterations until a steady state is reached. Then, we take the state with the highest probability to be the last element in the current set (latest image). After removing it from the chain, the computation is repeated until all nodes are ordered. Removing the state with the highest probability is the heuristic part of the algorithm. Empirically, this formulation was shown to work for Web ranking applications Dwork et al. (2001), and we found it to work for photo-sequencing as well.

The connectivity (transitivity) of the chain allows us to infer relations between pairs of images that were not explicitly ordered in any of the partial orders. This lets us aggregate all available information. If the graph G contains more than one sink, the order between the sinks cannot be determined (independent of the existence of cycles). This situation is unlikely to occur in our case if every pair of images is ordered by at least one feature.

3.3 Implementation Details

3.3.1 Detect and Match Features

Our method is insensitive to the way static and dynamic features are detected and matched, as long as enough features are available. We compute the fundamental matrices (using Goshen and Shimshoni 2006) using the detected static SIFT features. However, we found that it is difficult to match SIFT descriptors of dynamic features because of the non-rigid transformations of moving objects. To overcome this challenge, we first find a dense correspondence between each image and the reference image, using the Non-Rigid Dense Correspondence (NRDC) algorithm HaCohen et al. (2011). Then, we use the Harris corner detector to detect feature locations and use only features with high confidence mapping, where the confidence is defined by NRDC.

3.3.2 Computing the transitivity matrix $\mathbf{M}$

The number of images to be ordered in our case is relatively small, so we explicitly compute the $N \times N$ matrix $\mathbf{M}$ from the estimated permutations, $\sigma_1, \sigma_2, ..., \sigma_n$. Each permutation, σ_i, votes for the order of pairwise images in its subset. The pairwise votes from all permutations are collected in an $N \times N$ auxiliary voting matrix, $\mathbf{V}$, where $\mathbf{V}_{ij}$ is the voting score for $t(I_i) < t(I_j)$. The weight of the votes of each permutation is proportional to its cardinality, and is given by: $|n_i|/N$. Thus, a permutation that includes all N images has the highest voting weight.

An incorrect order produced by a single feature can already result in a cycle in the graph defined by V. Hence, we remove edges by choosing a single direction between each pair of nodes by comparing $\mathbf{V}_{ij}$ and $\mathbf{V}_{ji}$. We set the weight of the votes to reflect the votes and their relative impact. That is, the weight is proportional to the value that support the edge direction and inverse proportional to the value that oppose the edge direction: $\mathbf{V}_{ij} > \mathbf{V}_{ji}$, we set $\mathbf{M}_{ij} = 1 - \mathbf{V}_{ji}/\mathbf{V}_{ij}$. We normalize the rows of $\mathbf{M}$ to 1 because $\mathbf{M}$ is a stochastic matrix.

3.4 Time and 3D Estimation

So far we have recovered the relative temporal **order** of the images, but can we recover the exact capturing **time** of each image? And when are time and space equivalent? In this

section we address these questions and study under which conditions time and space are equivalent.

We consider again a single dynamic scene point, P, moving along a straight line, L, and projected to a subset of images at different time steps. Without loss of generality, the set of images is given by $\{I_k\}_{k=1}^{n}$, where I_1 is the reference image and I_2 is taken from the same position, by the same camera. The question is whether the 3D location and the capturing time of each of the images are interchangeable.

To infer information regarding the time one must have prior knowledge regarding the kinematics of the 3D point since the motion of the point and the time are ambiguous. Thus, we assume that P is moving in a constant, but unknown, velocity v_P within the capturing time interval. Furthermore, the analysis should be performed in a space in which lengths are preserved. Therefore, full calibration between the cameras is required (at each location and each time step). Note that since I_1, and I_2 are taken by the same camera, their time difference is assumed to be known. Without loss of generality, we take $t_1 = 0$ and $t_2 = 1$. (We use t_k as short for $t(I_k)$). The capturing time instances of the rest of the cameras are measured with respect to $t_1 = 0$.

We next show that under these assumptions, recovering the capturing time of the images is equivalent to reconstructing the 3D locations of P at the time step at which it was captured.

We first note that when four images are available, the 3D locations of the points are uniquely defined independent of the temporal information. To see that, consider the proof by Avidan and Shashua (2000) for recovering L. They proved that five rays in the 3D space, each defined by an image point and the camera parameters, are required to uniquely recover L. In our case the two cameras are co-located in space; hence the plane containing L is defined by the two rays of the reference image's points. In addition, the projection of $P(t_k)$ onto the reference image is computed using the intersection with the corresponding epipolar line (as in Sect. 3.1). Thus, the 3D location $P(t_k)$ can be recovered using triangulation from the two corresponding points, p_k^1 and p_k^k in I_1 and I_k, respectively. Hence, the pair of 3D points $P(t_k)$ and $P(t_j)$ uniquely defines the 3D line L. The intersection of L with the rays of the other points uniquely defines their 3D locations.

We will next focus on the $n = 3$ case, where temporal information is required for recovering the 3D locations.

Claim #1 *R*ecovering the capturing time t_k is equivalent to recovering the 3D locations $P(t_1)$, $P(t_2)$,*a*nd $P(t_k)$.

Proof $\Leftarrow$ Without loss of generality, we assume that $t_1 < t_2 < t_k$. Since $t_1 = 0$ and $t_2 = 1$, it follows that given the 3D point locations, $P(t_1)$ and $P(t_2)$, the velocity, v_P can be computed:

$$v_P = \frac{\|P(t_2) - P(t_1)\|}{|t_2 - t_1|}. \tag{6}$$

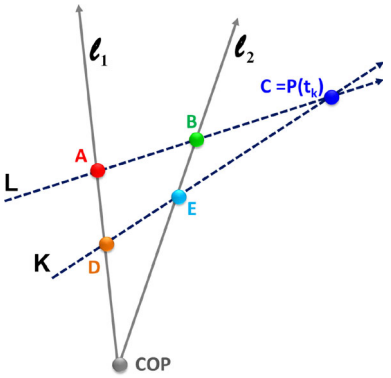

Fig. 7 Proof of the Lemma. Given two rays l_1, l_2 and the 3D point $P(t_k)$, there is only one line (L) that satisfies the time ratio constraints. See details in the text

Given $P(t_k)$, the time step t_k is given by:

$$t_k = \|P(t_k) - P(t_1)\| \,/\, \|P(t_2) - P(t_1)\|\,. \tag{7}$$

□

Note that if the time difference $t_1 - t_0$ is unknown, we can still recover t_k up to the scale, v_P.

$\Rightarrow$ The 3D location $P(t_k)$ can be recovered, as explained above, using triangulation from the two corresponding points, p_k^1 and p_k^k in I_1 and I_k, respectively. Thus, we have two rays, ℓ_1 and ℓ_2, that intersect at the center of projection of the reference camera, and a 3D point $P(t_k)$ that lies on the same plane (see Fig. 7). Ignoring temporal information for a moment, we can see that any line that passes through $P(t_k)$ and intersects the two rays is a possible solution for the unknown line L. However, only the true line would result in the known (or relative) time between the images, as we next prove.

Lemma *Assume perspective projection (Fig. 7). Then, of all the lines that pass through $P(t_k)$ and intersect the rays ℓ_1 and ℓ_2, only one line, L, satisfies the following:*

$$\|P(t_k) - P(t_1)\| \,/\, \|P(t_2) - P(t_1)\| = t_k/t_1, \tag{8}$$

where $P(t_1) = \ell_1 \times L$, and $P(t_2) = \ell_2 \times L$.

Proof **(by contradiction)** Assume that there are two lines, L and K, that satisfy Eq. 8, and let $C = P(t_k)$, $A = \ell_1 \times L$, $B = \ell_2 \times L$, $D = \ell_1 \times K$ and $E = \ell_2 \times K$ (see Fig. 7). It follows that:

$$|AC|/|AB| = |DC|/|DE| = t_k/t_1 \;\Rightarrow\; |AC|/|DC| = |BC|/|EC|. \tag{9}$$

□

Thus, $\Delta ACD \sim \Delta BCE$ (since they also share $\angle ACD$). Therefore, $\angle DAC = \angle EBC$, and $\angle ADC = \angle BEC$, which implies that $\ell_1 \| \ell_2$. This implies that the COP is at infinity (orthographic projection), and contradicts the perspective projection assumption.

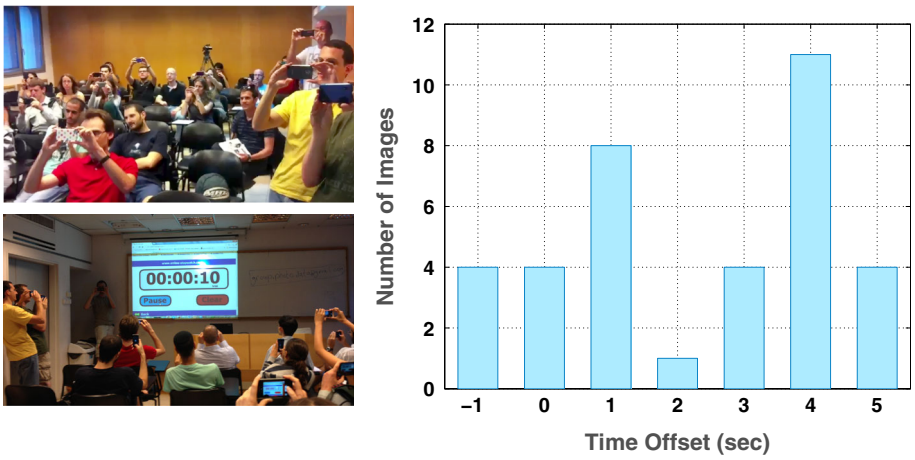

Fig. 8 Smartphone Synchronization: (**a**) the used setup; a group of 20 people (*top*) captured a screened running stopwatch (*bottom*); (**b**) the histogram of time offsets between the smartphones

4 Experimental Results

We conducted a number of experiments to test the proposed method. First, we measured the accuracy of smartphone synchronization, and then we tested our method on sets of real data images. Finally, we show, through a synthetic experiment, that it is possible to extend photo-sequencing to use more than a single reference camera.

4.1 Smartphones Synchronization

We tested to what extent different smartphones are synchronized using the time-stamp assigned to the photos' meta-data. To do so, we displayed a running stopwatch with millisecond accuracy, while a group of 20 people were asked to capture it when the clock reached 10 and 20 s (see Fig. 8a). We collected the 40 images and extracted the time-stamps associated with each image in the EXIF data.

Since not all the images were captured exactly at the same moment, we used the stopwatch time captured in each image to compensate for these differences. In particular, we computed the differences between the stopwatch time captured in each image and 10 (or 20) s, and corrected the EXIF time-stamps accordingly. Thus, the total offsets between the images are given by the differences between the corrected time-stamps.

As can be seen from the histogram of time offsets shown in Fig. 8b, the distribution is not unimodal, and there are two main peaks at around 1 and 4 s. It is important to note that two out of 20 phones had more than a one minute offset (they were not included in the histogram). Such outliers are phones for which the time is set by the phone's internal clock and not by the cellular network (as in most of the phones). This experiment indeed affirms that smartphones are not synchronized up to the frame capturing resolution. (Recall that video is captured at the rate of 30 frames per second.)

4.2 Real Data

We captured five challenging and diverse datasets of outdoor scenes. The images were captured from different viewpoints, without calibration or a controlled setup, by various cameras (including Apple iPhone 3 and 4, Samsung Galaxy SI/SII, Blackberry, and Canon PowerShot SD940 IS).

Matching features across images is very challenging and, in particular, we found that SIFT features cannot be correctly matched in the dynamic regions (see Sect. 3.3.1).

Another challenge for sequencing each of the datasets is the large search space for possible solutions, $N!/2$, where N is the number of images. In these datasets N is between 9 and 15. We tested our method on each of the five datasets without assuming a priori knowledge about the scene.

To evaluate the results of photo-sequencing, the ground truth temporal order is required. However, manually ordering still images is difficult (even more than manual video synchronization). Therefore, we captured video sequences instead of still images with each phone / camera. This allowed us to compute the ground truth. The input to our method is a set of extracted still images from the video sequences. Our algorithm was not provided with this temporal information.

In each experiment, we choose two images, I_1 and I_2, taken approximately from the same location, with known relative order. This is the only assumption made regarding

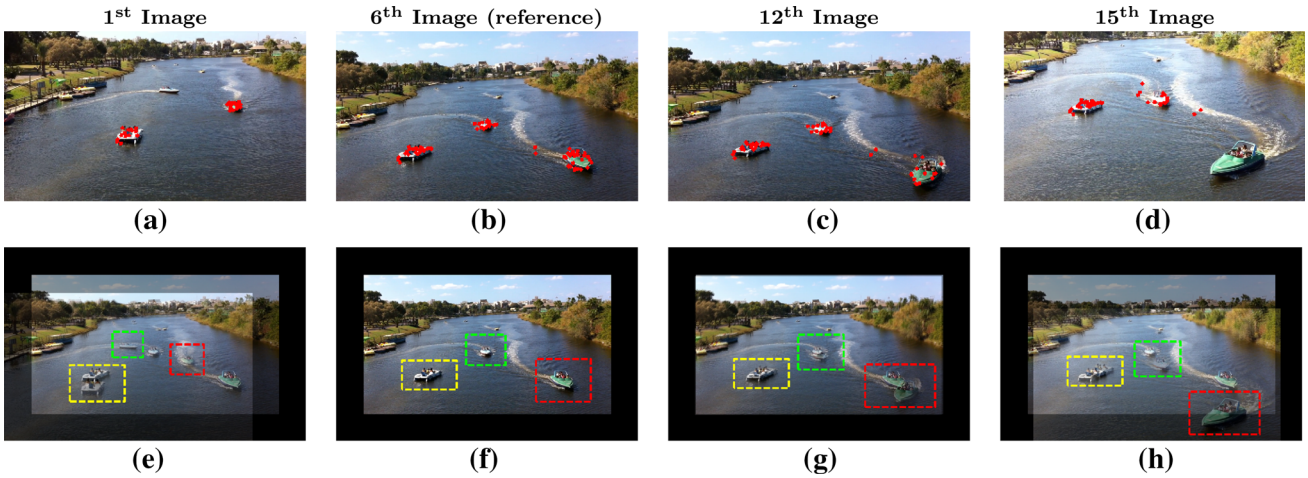

Fig. 9 Boats: First row **(a–d)**: four of the fifteen input images, ordered by our method; **(b)** is the reference; the detected dynamic features are marked in red points over the images. Second row: for evaluation purposes each boat is framed by hand and colored the same in all images. **(f)** The reference; each of the images **(a)**, **(c)** and **(d)** were aligned to it; **(e), (g), (h)** the aligned images of **(a), (c)** and **(d)**, respectively, are shown over the reference **(f)**

the order of the input images or the camera locations. The following datasets were considered:

4.2.1 Boats

The Boats dataset consists of 15 images extracted from sequences taken by hand-held mobile phones (iPhone 4). Ten images were taken by one phone, and the other 5 were taken from different locations by the other phone. The detected dynamic and static features of I_1 and I_2, are shown in Fig. 3c. Four of the input images (1st, 6th, 12th and 15th), arranged in the order computed by our method, are shown in Fig. 9a–d. The dynamic features are marked in red in each image. Note that the dynamic features may be different from image to image due to the difference in viewpoints and capturing time (as Fig. 9 shows). For example, the features detected in Fig. 9a, d belong to different objects, (e.g., see the right-hand boat). As can be seen, it is hard to visually determine

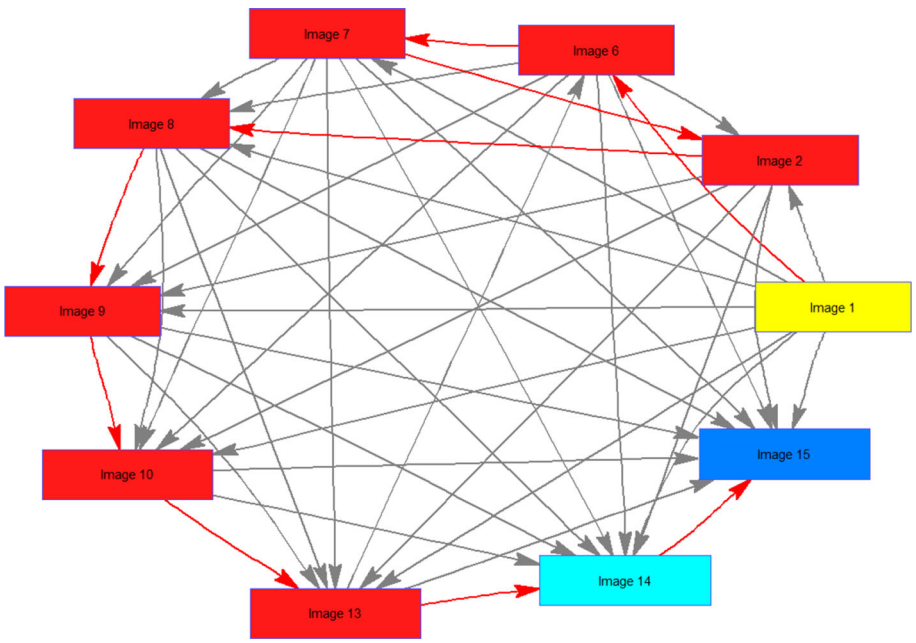

Fig. 10 The directed subgraph computed for the Boats dataset. The strongly connected nodes are marked in *red*. The correct detected order of the nodes is indicated by *red arrows*. Note that the incorrect edge that closes a cycle is (13, 6)

(a) **(b)**

Fig. 11 (**a**) The detected static and dynamic features of I_1 and I_2 are marked over the reference image, I_1; *blue* are I_1 features; *red* and *green* are I_2 static and dynamic features, respectively; on the right is a close-up of the region where the children appear. (**b**) The detected static and dynamic features for the Basketball dataset

the correct temporal order. Therefore, to verify our results, we aligned the three images, Fig. 9a, c, d, to the reference, Fig. 9b, using the static features. Fig. 9e–h shows the aligned images, semi-transparently, over the reference image. The resulting locations of three boats in the aligned images are shown in the colored frames. These locations agree with the recovered order.

The number of dynamic feature sets is 66, only 22 of which fully agreed with the correct order. In addition, the obtained graph contains a cycle. A subgraph that contains the strongly connected component of the cycle (with 7 nodes marked in red) is presented in Fig. 10. Therefore, this experiment demonstrates the necessity of using rank aggregation and the robustness of our photo-sequencing algorithm to aggregate multiple inconsistent partial orders into a globally consistent one.

4.2.2 Slide

The Slide dataset consists of ten images extracted from sequences captured by five different cameras: Samsung Galaxy SII, BlackBerry, iPhone 4 (two phones), and Canon PowerShot SD940 IS. Several cameras were mounted on a tripod, and the rest were hand-held. The detected features are shown in Fig. 11a. In Fig. 12 we present eight of the input images, arranged in the recovered order. The correct order of the images can be visually verified by the positions of the children along the slide. A closer look at Fig. 12g–h will reveal some incorrect dynamic features (e.g., the dynamic features detected in the background of (h)). However, since most of the detected features are correct, our method was able to recover the correct order.

4.2.3 Skateboard

The Skateboard dataset consists of nine images, extracted from sequences captured by a pair of hand-held mobile phones (iPhone 4 & Smasung Galaxy SI). The dynamic and static features of I_1 and I_2 are shown in Fig. 13a. A closer look shows that the corresponding static features (blue and red points) in I_1 and I_2 are not exactly at the same position (since the phone was hand-held). Since we threshold the distance between the features, we can handle slight movement between the two images. Figure 14 shows eight of the input images, arranged in the temporal order computed by our method. It can be seen that the viewpoints of the images are very different. Thus, we align the first and last ordered images (see Fig. 14a, h, respectively) to the reference image. Figure 13c–d shows the aligned images semi-transparently over the reference image. The resulting locations of the man in the aligned images are shown in the cyan and yellow frames. As can be seen, the man in the cyan frame indeed appears before the reference, whereas the yellow frame in Fig. 13d appears after.

(a) Canon SD940 IS **(b)** iPhone 4 **(c)** Galaxy SII **(d)** iPhone 4

(e) Galaxy SII **(f)** iPhone 4 **(g)** iPhone 4 **(h)** BlackBerry Torch

Fig. 12 Slide: Eight of the images ordered by our method (*left-to-right*, *top-to-bottom*). The dynamic features are overplayed on the images in *red*. (d) and (f) refer to I_1 and I_2, respectively

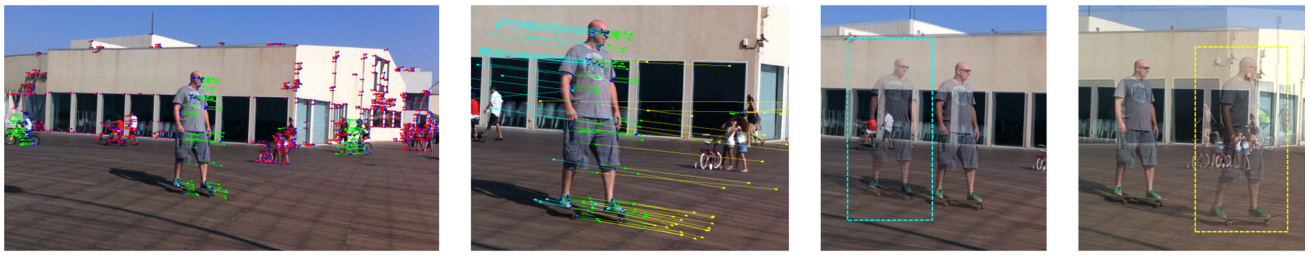

Fig. 13 (**a**) The reference image, I_1, overlayed with the detected corresponding features between I_1 and I_2 (taken by an approximately static camera): blue are I_1 features, and red and green are the I_2 static and dynamic features, respectively; (**b**) the image to the right is a close-up of the man region; (**c, d**) overlay images: the first and last ordered images (see Fig. 14(a, h)), are aligned to the reference image, and shown semi-transparently over it

4.2.4 Basketball

The Basketball dataset contains eight images extracted from sequences taken by a pair of mobile phones (iPhone 4), mounted on a tripod. The detected features are shown in Fig. 11b. As Fig. 15 shows, we can correctly order the photos even though the dynamic feature points move in different directions and follow a natural trajectory that is not necessarily linear. The correct order of the images can be visually verified by following the motion of the arm of the girl throwing the basketball.

4.2.5 Beach

The Beach dataset contains eight images extracted from sequences taken by a pair of hand-held mobile phones (iPhone 4 and iPhone 3). The detected features are shown in Fig. 16b. The scene consists of large low texture regions (the sky) in which static features cannot be found, and reflections. For this dataset the number of dynamic feature sets was only 32, while only 15 of them completely agreed with the correct order.

In all five datasets the number of dynamic feature sets were between 32 and 200, while only about 40 % of them fully agreed with the correct order. However, the pairwise voting was sufficient in three of the datasets to obtain a graph without cycles. Our method successfully obtained the correct order despite the difference in viewpoint, colors, resolution and aspect ratio between the images, as long as we managed to find correspondence (Fig. 16).

4.3 Analysis for Two Reference Images

So far we have tested our method using a single reference image (a single static image pair), but it can also integrate information from multiple reference images. This allows us to consider a wider range of camera configurations since not all the input images are required to be matched to one single reference image.

Fig. 14 Skateboard: Eight of the input images ordered by our method (*left-to-right*, *top-to-bottom*). The dynamic features are marked in *red*. The *yellow* and cyan frames are the first and last ordered images, respectively

(a) 2nd Image **(b)** 5th Image **(c)** 6th image **(d)** 7th image

Fig. 15 Basketball: Four of the eight input images, ordered by our method (*left-to-right*); the detected dynamic features are marked in *red* points over the images

In this experiment, we generated synthetic data to demonstrate the use of two reference images, i.e., two image pairs, each taken roughly from the same location. In particular, we simulated a 3D scene, captured by a set of 58 cameras. The cameras were randomly placed on a half-circle, facing the center (see Fig. 17a). Two of the cameras were chosen as the reference cameras; each provided two images from the same location (in total 60 images from all cameras). The 3D scene was generated by randomly choosing 100 lines inside a bounding box located in the center of the circle, where each line was projected to a random subset of the images. Gaussian noise with zero mean and $\sigma = 1$ (measured in pixel) was added to the 2D features.

In order to simulate a realistic scenario, we divided the cameras into two groups, each associated with one reference camera. We computed the partial orders (votes) for each group individually, and then fed the Rank Aggregation by the votes from both groups. Obviously, if there is no overlap between the groups, it is impossible to derive the full order of all the 60 images. The bigger the overlap is, the better the chance to obtain the correct order. Therefore, we tested our method considering 5 different levels of overlap between the groups. Due to the random nature of the simulation, we repeated each test 10 times, and computed the mean error and its standard deviation (see Fig. 17b). As expected, the mean error drops with the amount of overlap between the groups (from 4.89% error with 10.17% overlap to 1.08% with 63.17% overlap). Nevertheless, the maximal error is below 5% incorrect pairwise orders out of a total of 1770 image pairs. Table 1 shows the mean error and the average group size when considering only one of the reference images, and when integrating the information from both references. Note that mean error when considering a single reference image is computed over the subset of images associated with the group, which is smaller than the total number of images; hence, the error for a single reference may be lower than the error for the two reference case.

5 Limitations and Conclusions

Photo-sequencing orders a set of images in the correct temporal order. This is useful in real world scenarios when a group of people captures still photos of some dynamic event at approximately the same time.

We proposed a geometry-based method which aggregates partial orders of the images computed from a set of dynamic features. We make several assumptions to model the problem. These assumptions let us reduce temporal order to the order of line intersections in the image plane. In practice, our

(a) 1st Image **(b)** 4th Image **(c)** 6th image **(d)** 7th image **(e)** 8th image

Fig. 16 Beach: Five of the eight input images, ordered by our method (*left-to-right*); the detected dynamic features are marked in *green* points over the images

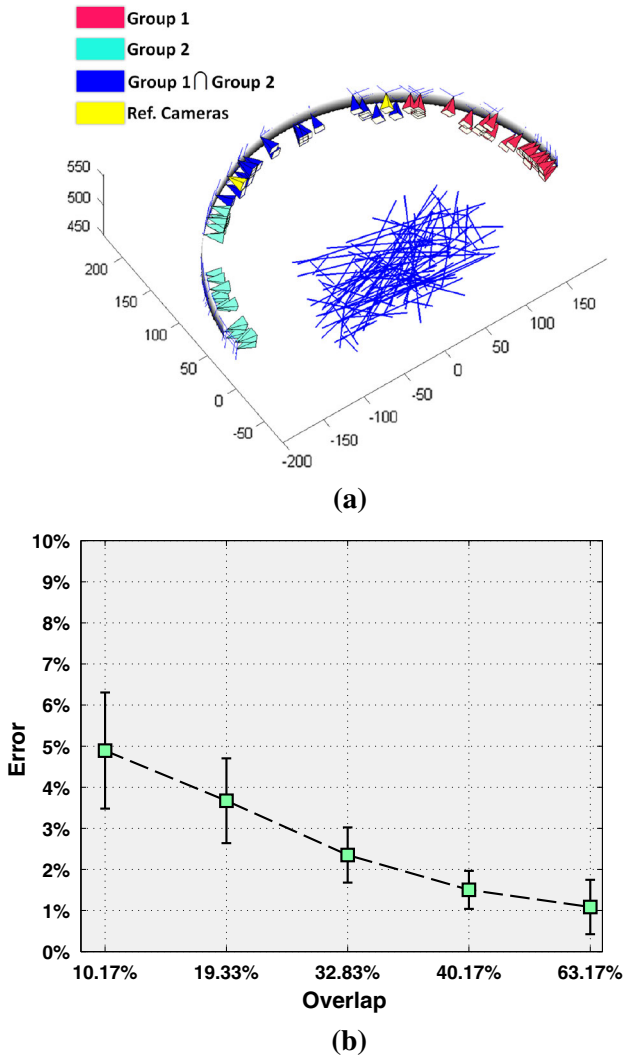

Fig. 17 Synthetic Experiment: (**a**) 3D visualization of the cameras setup; the two reference images are colored in *yellow*. The group of cameras associated with the left and right reference are colored in *green* and *red*, respectively; the cameras associated with both reference images are colored in *blue*. (**b**) The computed mean error and its variance when increasing the amount of overlap between Group I and Group II

algorithm is quite insensitive to some of these assumptions, as the only thing matters is the *order* of the intersections and not their actual location. In particular, we found that our algorithm can handle cases in which the reference camera moves to some extent and the motion of dynamic features deviates considerably from a linear trajectory (as demonstrated in our experiments). This can be seen in Fig. 6, where the actual path of the green boat was very far from the approximated linear line, yet the order of the intersection points did not change.

Regarding limitations, our method cannot rely on features with repeated motions (back and forth) to derive the temporal order. Such features vote for incorrect ordering and are treated as noise. Moreover, our method requires that each of the input images be reliably matched to the reference image. That is, if one of the input images does not sufficiently overlap with the reference image, its temporal information cannot be recovered. Although we do not require that all features be found in all images, this assumption limits the range of spatial and temporal camera configurations that can be considered.

Table 1 Synthetic Experiment: For each amount of overlap between Group I and II, the table shows: the average number of cameras (group size) and the mean error over 10 tests; these values are computed when using only the group associated with reference (I), reference (II), and both refrences (I)+(II).

	Avg # cameras			Mean err. (%)		
Overlap (%)	I	II	I+II	I	II	I+II
10.17 (%)	32	34.1	60	2.03	1.92	4.89
19.33 (%)	35.5	36.1	60	1.38	1.96	3.67
32.83 (%)	41.9	38.4	60	2.19	1.45	2.35
63.17 (%)	48.7	50.1	60	1.29	1.58	1.08

In the future, we intend to relax the assumptions made by our method, and make it more scalable in both time and space. In addition, our method can be extended to longer time periods by first coarsely ordering the images using their EXIF timestamp, and then performing a finer ordering using our method in a sliding window fashion. We believe that photo-sequencing in a general setup will allow the development of novel computer vision and graphics applications for dynamic scenes captured by a CrowdCam.

Acknowledgments This work was supported in part by Israel Science Foundation Grant No. 1556/10 and 930/12, and European Community Grant PIRG05-GA-2009-248527.

References

Avidan, S., & Shashua, A. (2000). Trajectory triangulation: 3d reconstruction of moving points from a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(4), 348–357.

Ballan, L., Brostow, G. J., Puwein, J., & Pollefeys, M. (2010). Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM Transactions on Graphics (TOG)*, *29*(4), 115–122.

Bartoli, A., Gay-Bellile, V., Castellani, U., Peyras, J., Olsen, S., and Sayd, P. (2008). Coarse-to-fine low-rank structure-from-motion. In *Proceeding IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Caspi, Y., & Irani, M. (2002). Spatio-temporal alignment of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(11), 1409–1424.

Dekel (Basha), T., Moses, Y., and Avidan, S. (2012). Photo sequencing. In *Proceeding European Conference of Computer Vision (ECCV)*.

Dexter, E., Pe'rez, P., and Laptev, I. (2009). Multi-view synchronization of human actions and dynamic scenes. In *Proceeding British Machine Vision Conference (BMVC)*.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. (2001). Rank aggregation methods for the web. In *International Conference on World Wide Web*.

Elena, R. M. and Straccia, U. (2003). Web metasearch: rank vs. score based rank aggregation methods. In *Proceedings of the 2003 ACM Symposium on Applied, Computing*.

Goshen, L. and Shimshoni, I. (2006). Balanced exploration and exploitation model search for efficient epipolar geometry estimation. In *Proceeding European Conference of Computer Vision (ECCV)*.

HaCohen, Y., Shechtman, E., & Goldman, D. B., Lischinski. D. (2011).Non-rigid dense correspondence with applications for image enhancement. *ACM Transactions on Graphics (SIGGRAPH)*.

Jegou, H., Schmid, C., Harzallah, H., & Verbeek, J. (2010). Accurate image search using the contextual dissimilarity measure. *IEEE Transactions on: Pattern Analysis and Machine Intelligence*, *32*(1), 2–11.

Kaminski, J. Y., & Teicher, M. (2004). A general framework for trajectory triangulation. *Journal of Mathematical Imaging and Vision*, *21*(1), 27–41.

Lei, C., & Yang, Y. (2006). Tri-focal tensor-based multiple video synchronization with subframe optimization. *IEEE Transactions on Image Processing*, *15*(9), 2473–2480.

Llado, X., Del Bue, A., and Agapito, L. (2005). Non-rigid 3d factorization for projective reconstruction. In *Proceeding British Machine Vision Conference (BMVC)*.

Meyer, B., Stich, T., Magnor, M., and Pollefeys, M. (2008). Subframe temporal alignment of non-stationary cameras. In *Proceeding British Machine Vision Conference (BMVC)*, pages 103–112.

Pádua, F. L. C., Carceroni, R. L., Santos, G. A. M. R., & Kutulakos, K. N. (2010). Linear sequence-to-sequence alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(2), 304–320.

Park, H., Shiratori, T., Matthews, I., and Sheikh, Y. (2010). 3d reconstruction of a moving point from a series of 2d projections. In *Proceeding European Conference of Computer Vision (ECCV)*.

Pedronette, D.C.G., and Torres, R.S. (2011). Exploiting contextual information for rank aggregation. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*.

Pundik, D. and Moses, Y. (2010). Video synchronization using temporal signals from epipolar lines. In *Proceeding European Conference of Computer Vision (ECCV)*.

Sand, P., & Teller, S. (2004). Video matching. *In ACM Transactions on Graphics (TOG)*, *23*(3), 592–599.

Schalekamp, F. and van Zuylen, A. (2009). Rank aggregation: Together were strong.In *Proceeding of 11th ALENEX*.

Schindler, G., and Dellaert, F. (2010). Probabilistic temporal inference on reconstructed 3d scenes. In *Proceeding IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1410–1417. IEEE.

Shashua, A. and Wolf, L. (2000). Homography tensors: On algebraic entities that represent three views of static or moving planar points. In *Proceeding European Conference of Computer Vision (ECCV)*.

Shili, L. (2010). *Rank aggregation methods*. Wiley Interdisciplinary Reviews: Computational Statistics.

Snavely, N., Simon, I., Goesele, M., Szeliski, R., & Seitz, S. M. (2010). Scene reconstruction and visualization from community photo collections. *IEEE Special Issue on Internet Vision: Proc.*

Torresani, L., Hertzmann, A., & Bregler, C. (2008). Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(5), 878–892.

Tresadern, P., & Reid, I. (2003). Synchronizing image sequences of non-rigid objects. *Proc. British Machine Vision Conference (BMVC)*, *2*, 629–638.

Whitehead, A., Laganiere, R., and Bose, P. (2005).Temporal synchronization of video sequences in theory and in practice. In *WMVC*.

Wolf, L., & Zomet, A. (2002). Correspondence-free synchronization and reconstruction in a non-rigid scene. In *Proceeding Workshop Vision and Modeling of Dynamic Scenes*

Yan, J., and Pollefeys, M. (2004). Video synchronization via space-time interest point distribution. In ACIVS.

Young, H. P., & Levenglick, A. (1978). A consistent extension of condorcet's election principle. *SIAM Journal on Applied Mathematics*, *35*(2), 285–300.

Young, H. P. (1988). Condorcet's theory of voting. *The American Political Science Review*, *82*, 1231–1244.

Zelnik-Manor, L., and Irani, M. (2003). Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In *Proceeding IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.