Photo Sequencing

Tali Basha[1], Yael Moses[2], and Shai Avidan[1]

[1] Tel Aviv University, Tel Aviv 69978, Israel
[2] The Interdisciplinary Center, Herzliya 46150, Israel
{talib, avidan}@eng.tau.ac.il, yael@idc.ac.il

Abstract. Dynamic events such as family gatherings, concerts or sports events are often captured by a group of people. 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.

1 Introduction

In group photography a group of people takes pictures of some dynamic event such as concert, sports event or a family gathering. The photos are often taken within a short time interval of one of the event highlights, using what is probably the most popular means of photography today – cellphones. The question that motivates our study is whether it is possible to visualize and analyze the *dynamic* content of the scene using a set of still images collected by the group. An essential tool required for 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 same camera as in a video sequence. Alternatively, when two or more sequences are taken, temporal order can be recovered using video synchronization methods. In our case, however, the only temporal information available is given by the phones' imprecise clocks. 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.*

The input to our method is a set of still images captured by asynchronous and uncalibrated mobile phones (or other digital cameras) that are co-located in space and time. Such a set can be extracted using the inaccurate cellphone time-stamp associated with each image. It is important to note that video synchronization methods are not applicable here because each camera may provide only one still image.

The photo sequencing problem could be directly solved if the 3D structure of the dynamic scene were known. However, recovering the 3D structure of a dynamic scene often requires 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.

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 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. 1(a)). Our method avoids recovering the sampled 3D locations of a dynamic scene point by computing its projections to the reference image. To do so, we extract and match sets of static and dynamic features between each of the images and a reference image (see Fig. 1(b)). Using the fundamental matrices (computed by the static features), each set of corresponding dynamic features is projected onto the reference image. These projections preserve the spatial order along the 3D trajectory of the scene point, and hence provide the partial time order of the subset of 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 we have to solve is to compute a *full temporal order*, i.e., an order of all input images, which is as consistent as possible with the computed partial orders. This problem is equivalent to the *rank aggregation* problem, which is known to be NP-hard for the most general case [1]. 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 a simple polynomial algorithm to recover a directed Hamiltonian path can be used. However, the graph often contains cycles. In this case computing the Hamiltonian path is NP-hard, hence we use a Markov chain approximation to solve the problem [1]. This solution was shown to perform well on ranking search results of multiple search engines.

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

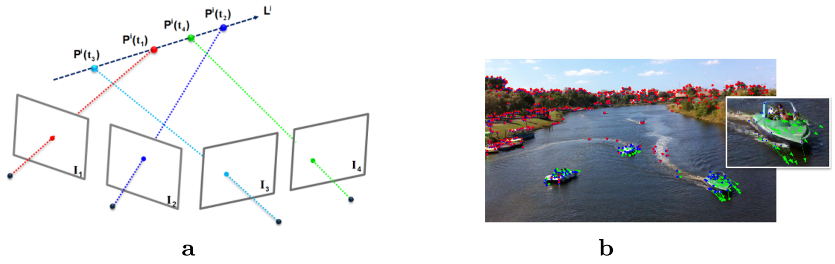


Fig. 1. (**a**) The order of the sampled location of a 3D point moving along a straight line implicitly induces the order of the corresponding images: $t(I_3) < t(I_1) < t(I_4) < t(I_2)$. (**b**) Static & dynamic features of the Boats dataset: the detected corresponding features of I_1 and I_2 (taken 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.

event. The method's robustness is based on a rank aggregation algorithm that aggregates noisy measurements to overcome inconsistencies.

Possible applications of photo sequencing include visualizing and analyzing dynamic content 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. In addition it may be used to generalize different computer vision tasks that uses videos to a set of still images such as object tracking, segmentation, or any other analysis of the dynamic content from a set of still images. These applications, which are beyond the scope of this paper, arise naturally when people share their images and are willing to extract the best out of the shared images.

2 Related Work

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., [2]), aligning tracked trajectories of features or objects visible in both videos (e.g., [3–5]), aligning all the frames (e.g., [3, 6]), assuming linear combination between the objects' views under orthographic projection (e.g.,[7]), assuming a low-rank of non-rigid moving objects (e.g., [8]), using tri-focal tensor-based relations when at least 3 videos are considered for spatial matching of points or lines (e.g.,[9]). Other synchronization methods attempt to bypass the computation of spatial correspondence in these methods, by using spatio-temporal feature statistics (e.g., [10]), or temporal signals defined over corresponding epipolar lines [11]. Geometric constraints were used by other synchronization methods for aligning multiple video sequences (e.g., [12]) or for achieving sub frame accuracy (e.g., [13]). In both methods the intersections of the trajectory of dynamic features with the epipolar lines of corresponding features in the other images were used

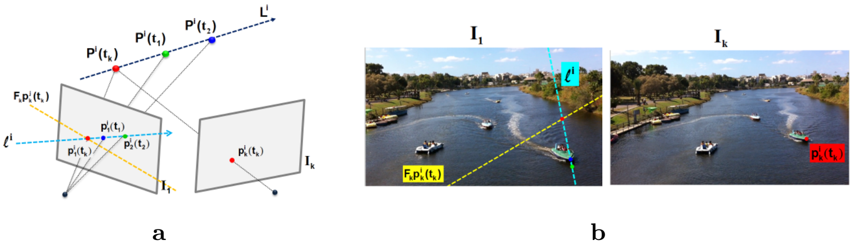


Fig. 2. **(a)** The projection of the trajectory, L^i, of the point P^i, forms the line ℓ^i on image I_1. The feature points $p_1^i(t_1)$, $p_2^i(t_2)$, in image I_1, and $p_k^i(t_k)$ in image I_k, are corresponding dynamic features. The line ℓ^i intersects the epipolar line (in yellow), which corresponds 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.

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.

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 [14]. 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 [15]. Trajectory triangulation was later extended using polynomial representations [16]. 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 is needed. Moreover, our method proposes a way to overcome the inconsistent ordering obtained by different features and allows to deviate from the linear motion assumption.

A different class of methods use factorization to deal with dynamic scenes [17]. 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 increasing the rank of the observation matrix, the non-rigid components of motion are captured by additional eigenvectors. This was later extended by [18, 19]. Indeed the solutions to these methods may result in photo sequencing. However, they are limited to restricted scenes, require features to be matched across a large number of images, which is not required by our method.

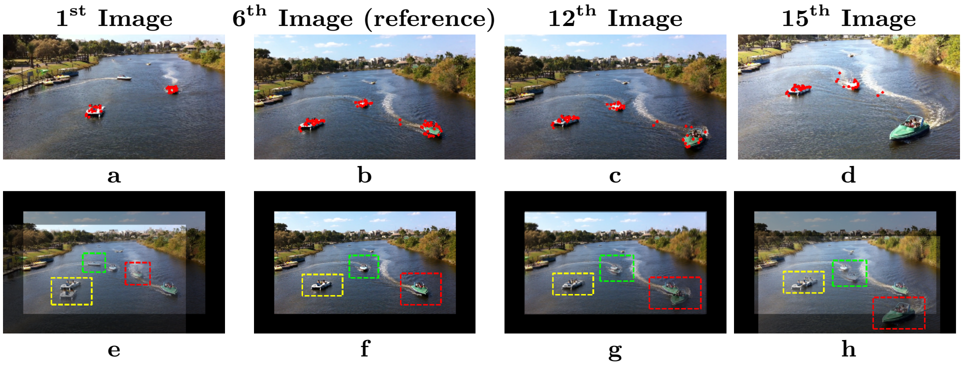


Fig. 3. Boats: First row **(a-d)**: four of the fifteen input images, ordered by our method; **(b)** is the reference; the detected dynamic features are marked in red points over the images. Second row: for evaluation purpose each boat is framed by hand and colored in the same color in all images. **(f)** the reference; each of the images **(a)**, **(c)** and **(d)** were aligned to the reference; **(e)**,**(g)**,**(h)** the aligned images of **(a)**, **(c)** and **(d)**, respectively, are shown over the reference **(f)**.

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.*[20]). 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 et al. [21]. 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 [22], but they use a collection of casually captured videos rather than still images as we do.

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, $\sigma : \{1, ..., N\} \rightarrow \{1, ..., N\}$, such that

$$t(I_{\sigma(1)}) \leq t(I_{\sigma(2)}) ... \leq t(I_{\sigma(N)}), \tag{1}$$

where $t(I_k)$ is the time at which image I_k was taken and $\sigma(i)$ indicates the temporal rank of image i. We assume that two of the images are taken from

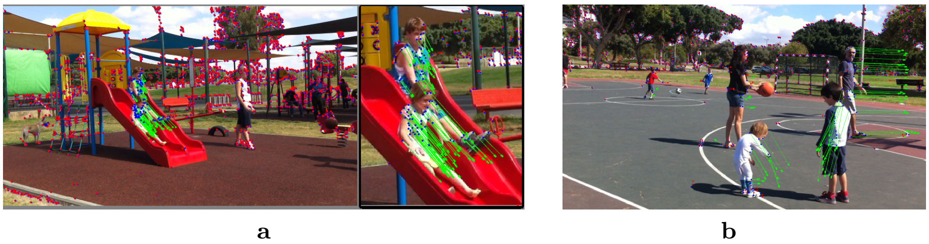


Fig. 4. (a) Slide dataset: The detected static and dynamic features corresponding features of I_1 and I_2 are marked over the reference image, I_1; blue are I_1 features; red and green are I_2 static and dynamic features, respectively; on the right is a close-up of the region where the children appear. **(b)** The detected static and dynamic features for the Basketball dataset.

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

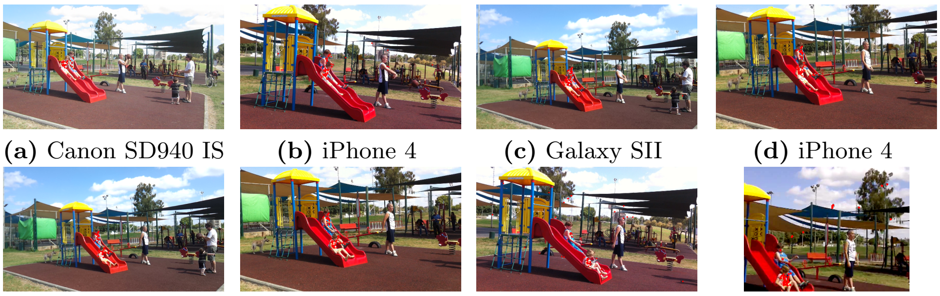
3.1 Temporal Order Voting

We first extract and match features from all input images (see Sec.3.3 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 first 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. 1(b).

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 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 [23]); the dynamic features are used to determine the temporal order of the images, as explained next.

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. 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 (see Fig. 1), and its projection to the reference image is given by ℓ^i. The set of features S^i defines a set of epipolar lines in I_1 which intersect the line ℓ^i. 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 (see Fig. 2).

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: $S^i = \{p_k^i(t_k) \mid k \in n_i\}$, where $n_i \subseteq \{1, ..., N\}$ is the subset of image indices. From the matching process, we have direct access to the projection of $P^i(t_k)$ to image I_k, namely $p_k^i(t_k)$.



(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. 5. 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.

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 $k \in n_i$. 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. 2(b)).

The line ℓ^i is defined by the two corresponding points, $p_1^i(t_1)$ and $p_2^i(t_2)$. That is, $\ell^i = p_1^i(t_1) \times p_2^i(t_2)$. Note that $p_1^i(t_2) \approx p_2^i(t_2)$ since both points were captured roughly from the same position. Putting it all together, we have that:

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

This equation degenerates if the epipolar line, ℓ_k^i, coincides with ℓ^i. We detect this case by measuring the angle between the two lines, and removing 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 presented 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 *order* of the intersections of the epipolar lines and the projection of the real trajectory. If this is not satisfied, then an incorrect order is produced by this feature. However, since the information from all features is aggregated, it is expected that 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.

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:

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

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 [1]. We adopt the Markov chain approximation of [1] that was shown to work well in Web ranking applications.

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 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. 9). 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, 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.

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, $\sigma_1, \sigma_2, ..., \sigma_{N_D}$ (see 3.3 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. In case 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 corresponds to the last captured image.

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 a initial uniform distribution $\mathbf{x}$, will converge to the eigenvector $\mathbf{y} = \mathbf{My}$. The eigenvector can be computed directly or using power iterations. In case 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 are 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 [1], 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 (it does not matter if there are cycles in the graph or not). 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

Detect and match features: Our method is insensitive to the way static and dynamic features are detected and matched, as long as enough features

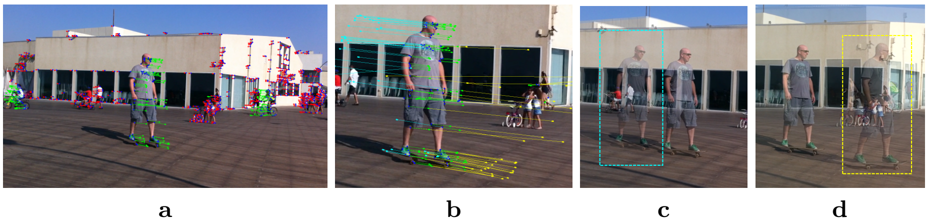


Fig. 6. **(a)** The reference image, I_1, overlayed with the detected corresponding features between I_1 and I_2 (taken by an approximately static camera): blue are I_1 features, and red and green are the I_2 static and dynamic features, respectively; **(b)** the image to the right is a close-up of the man region; **(c),(d)** overlay images: the first and last ordered images (see Fig.7.(a),(h)), are aligned to the reference image, and shown semi-transparently over it.

are available. We compute the fundamental matrices (using [23]) using SIFT features that were detected in the static regions. 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 [24]. 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.

Computing the transitivity matrix 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 a $N \times N$ 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 support and oppose voting of 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.

4 Results

We generated four challenging datasets of outdoor scenes. The images were captured from different viewpoints, without calibration or a controlled setup, by various cameras (including Apple iPhone 4, Samsung Galaxy SI/SII, Blackberry, and Canon PowerShot SD940 IS). Hence, matching features across images is very challenging. In particular, we found that SIFTs features cannot be correctly

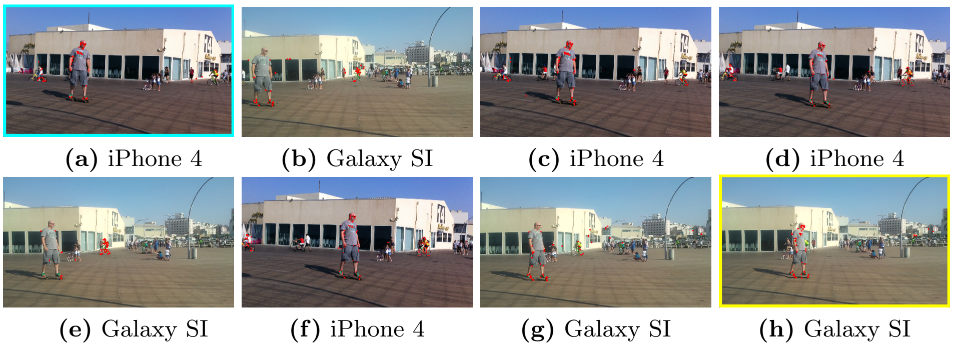


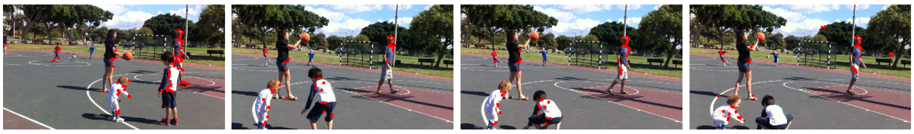
Fig. 7. Skateboard: Eight of the input images ordered by our method (left-to-right, top-to-bottom). The dynamic features are marked in red. The yellow and cyan frames are the first and last ordered images, respectively.

matched in the dynamic regions (see Sec. 3.3). Another challenge for sequencing each of the datasets is the large search space for possible solutions, that is $N!/2$, where N is the number of images. In these datasets N is between 9-15. We tested our method on each of the four 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). Hence, we captured video sequences instead of still images with each phone/camera. This allowed us to compute the ground truth of the data. The input to our method is a set of extracted still images from the sequences. However, 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 the order of the input images or the camera locations. The following datasets were considered:

Boats: This 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. 1(b). Four of the input images (1^{st}, 6^{th}, 12^{th} and 15^{th}), arranged in the order computed by our method, are shown in Fig. 3(a-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. 3 shows). For example, the features detected in Fig. 3(a) and Fig. 3(d) belong to different objects, (e.g., see the right-hand boat). As can be seen, it is hard to visually determine the correct temporal order. Therefore, to verify our results, we aligned the three images, Fig. 3(a,c,d), to the reference, Fig. 3(b), using the static features. Fig. 3(e-h) shows the aligned images, semi-transparently, over the reference image. The



(a) 2^{nd} Image (b) 5^{th} Image (c) 6^{th} image (d) 7^{th} image

Fig. 8. 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.

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, and only 22 of which fully agreed with the correct order. In addition, the obtained graph contain a cycle. A subgraph that contains the strongly connected component of the cycle (with 7 nodes marked in red) is presented in Fig. 9. 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.

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. 4(a). In Fig 5 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. 5(g)-(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.

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. 6(a). 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. Fig. 7 shows eight of the input images, arranged in the temporal order computed by our method. As Fig. 7 shows, the viewpoints of the images are very different. Thus, we align the first and last ordered images (see Fig. 3(a) and Fig. 3(h), respectively) to the reference image. Fig. 6(c-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. 6(d) appears after.

Basketball: This 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. 4(b). As Fig. 8 shows, we can correctly order the

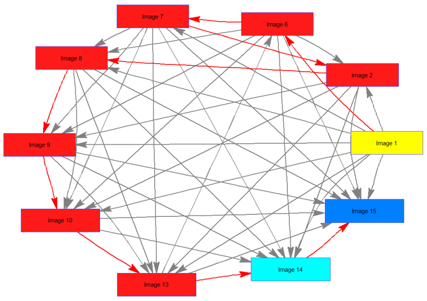


Fig. 9. The directed subgraph computed for the Boats dataset. The strongly connected nodes are marked in red. The correct detected order of the nodes is shown in red. Note that the incorrect edge that close a cycle is $(13, 6)$.

photos even though the dynamic feature points move in different directions (see Fig. 8), 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.

In all four datasets the number of dynamic feature sets were between 61 to 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.

5 Conclusions and Future Work

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 believe that photo sequencing in a general setup will allow the development of novel computer vision and graphics applications for dynamic scenes from a set of still images.

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 which let us reduce temporal order to the order of line intersections in the image plane. However, our 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 considerably deviate from a linear trajectory (as demonstrated in our experiments). Further relaxing the assumptions made by our method clearly remains for future work.

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

References

1. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: International Conference on World Wide Web. (2001)
2. Dexter, E., Pe'rez, P., Laptev, I.: Multi-view synchronization of human actions and dynamic scenes. In: BMVC. (2009)
3. Caspi, Y., Irani, M.: Spatio-temporal alignment of sequences. PAMI (2002)
4. Tresadern, P., Reid, I.: Synchronizing image sequences of non-rigid objects. In: BMVC. Volume 2. (2003) 629–638
5. Whitehead, A., Laganiere, R., Bose, P.: Temporal synchronization of video sequences in theory and in practice. In: WMVC. (2005)
6. Sand, P., Teller, S.: Video matching. In: ACM (TOG). (2004)
7. Wolf, L., Zomet, A.: Correspondence-free synchronization and reconstruction in a non-rigid scene. In: VMODS. (2002)
8. Zelnik-Manor, L., Irani, M.: Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In: CVPR. (2003)
9. Lei, C., Yang, Y.: Tri-focal tensor-based multiple video synchronization with subframe optimization. IEEEIP (2006)
10. Yan, J., Pollefeys, M.: Video synchronization via space-time interest point distribution. In: ACIVS. (2004)
11. Pundik, D., Moses, Y.: Video synchronization using temporal signals from epipolar lines. In: ECCV. (2010)
12. Pádua, F.L.C., Carceroni, R.L., Santos, G.A.M.R., Kutulakos, K.N.: Linear sequence-to-sequence alignment. PAMI (2010)
13. Meyer, B., Stich, T., Magnor, M., Pollefeys, M.: Subframe temporal alignment of non-stationary cameras. In: BMVC. (2008) 103–112
14. Avidan, S., Shashua, A.: Trajectory triangulation: 3d reconstruction of moving points from a monocular image sequence. PAMI (2000)
15. Shashua, A., Wolf, L.: Homography tensors: On algebraic entities that represent three views of static or moving planar points. ECCV (2000)
16. Kaminski, J.Y., Teicher, M.: A general framework for trajectory triangulation. JMIV (2004)
17. Torresani, L., A.Hertzmann, Bregler, C.: Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. PAMI (2008)
18. Bartoli, A., Gay-Bellile, V., Castellani, U., Peyras, J., Olsen, S., Sayd, P.: Coarse-to-fine low-rank structure-from-motion. In: CVPR. (2008)
19. Llado, X., Del Bue, A., Agapito, L.: Non-rigid 3d factorization for projective reconstruction. In: BMVC. (2005)
20. Snavely, N., Simon, I., Goesele, M., Szeliski, R., Seitz, S.: Scene reconstruction and visualization from community photo collections. Proc. IEEE Special Issue on Internet Vision (2010)
21. Schindler, G., Dellaert, F.: Probabilistic temporal inference on reconstructed 3d scenes. In: CVPR, IEEE (2010) 1410–1417
22. Ballan, L., Brostow, G., Puwein, J., Pollefeys, M.: Unstructured video-based rendering: Interactive exploration of casually captured videos. ACM (TOG) (2010)
23. Goshen, L., Shimshoni, I.: Balanced exploration and exploitation model search for efficient epipolar geometry estimation. In: ECCV. (2006)
24. HaCohen, Y., Shechtman, E., Goldman, D., Lischinski, D.: Non-rigid dense correspondence with applications for image enhancement. SIGGRAPH (2011)