[Supplementary]
Jointly Aligning and Segmenting Multiple Web Photo Streams for the Inference of Collective Photo Storylines

Gunhee Kim Eric P. Xing
School of Computer Science, Carnegie Mellon University
{gunhee,epxing}@cs.cmu.edu

1. Overview of Algorithm

We summarize the overview of our approach in Algorithm 1. The step **2–3** describe the alignment of multiple photo streams, and the step **4–6** outline the large-scale cosegmentation. We can iterate running these two major procedures until the output converges or maximum iterations reach.

Algorithm 1: Jointly aligning and segmenting multiple Web photo streams.

Input: (1) A set of photo streams $\mathcal{P} = \{\mathcal{P}\}_{l=1}^{L}$ with timestamps (also denoted by $\mathcal{I}$). (2) Number of foregrounds K in an unsupervised case or labeled foreground examples in a supervised case.

Output: (1) Photo stream (PS) alignment (*i.e.* L-partite graph $(\mathcal{I}, \mathcal{E}_B)$ where $\mathcal{E}_B$ include all matched pairs of images). (2) Image segmentation $\{\mathcal{F}_i\}$ for all $I_i \in \mathcal{I}$.

1: Perform feature extraction (sec. 3.1) and oversegmentation (sec. 4.2) for all $I_i \in \mathcal{I}$.

repeat

2: Define the image similarity σ as follows. At round 1, use the histogram intersection on the two-level pyramid histograms. From round 2 when images are segmented, use Eq.(1) instead.

3: For each PS, find K_P nearest ones by using NBNN method (sec. 3.4). The output is $\mathcal{E}_P$, the set of all pairs of nearest PS.

4: Run multiple photo stream alignment by solving Eq.(3) on $\mathcal{E}_P$ (sec. 3.4). The output is an L-partite graph $(\mathcal{I}, \mathcal{E}_B)$.

5: Build an image graph $\mathcal{G}_I$ from $\mathcal{E}_B$ (sec. 4.1).

6: Run large-scale cosegmentation by solving Eq.(4) (sec. 4.2). The output is the image segmentation $\{\mathcal{F}_i\}$ for all $I_i \in \mathcal{I}$.

until $\mathcal{E}_B$ *is not updated or maximum iterations reach*;

2. More Details of Experiments

In this section, we elaborate the application of our algorithms and baselines for the experiments. Then, we present more qualitative results of our experiments.

2.1. Application of Our algorithms and Baselines for Alignment Evaluation

Given randomly split training and test photo streams, the goal of each algorithm for the temporal localization task is to estimate the timestamps of all the images in the test photo streams.

Our algorithm (`BP`) **and** (`BPS`): Overall, our alignment algorithms are applied as described in the paper. However, the objective function of alignment in Eq.(2) assumes that the timestamps of photo streams are available, which is not the case for the test images in our experiments. Therefore, we use ordering information instead of the time information for Eq.(2).

Our algorithm (`BP`) and (`BPS`) differ from each other according to whether image segmentation is in a loop or not. Since the (`BP`) does not exploit the image segmentation output, we compute the image similarity from two-level spatial pyramid histograms on the whole images. On the other hand, for the (`BPS`), we run one complete loop of alignment and cosegmentation, and carry out the alignment again by using the segmentation-based image similarity metric of Eq.(1).

Baseline (`KNN`): For each test photo stream P^t, we first find K_T closest photo streams $\mathcal{P}^r$ from the training set by using the NBNN method in section 3.4. Then, for each test image $p \in P^t$, we search for K_p nearest images from $\mathcal{P}^r$. Finally, as an estimated timestamp of p, we compute the average of timestamps of K_p retrieved nearest images.

Baseline (`DTW`): For each test photo stream P^t, we find K_T closest photo streams $\mathcal{P}^r$, as done in the (`KNN`) baseline. Then, we perform the pairwise alignment between P^t and each photo stream in $\mathcal{P}^r$ by using the DTW (Dynamic time warping) algorithm. Finally, as an estimated timestamp of p, we compute the average of timestamps of the images that are matched to p.

Baseline (`HMM`): For each test photo stream P^t, we find K_T closest photo streams $\mathcal{P}^r$, as done in the (`KNN`) baseline. We first run K-means clustering to the descriptors of randomly chosen images from $\{P^t \cup \mathcal{P}^r\}$, in order to define observation alphabets. By assigning the closest alphabet to each image, we can represent each photo stream as a sequence of alphabets. Then, we apply Baum-Welch algo-

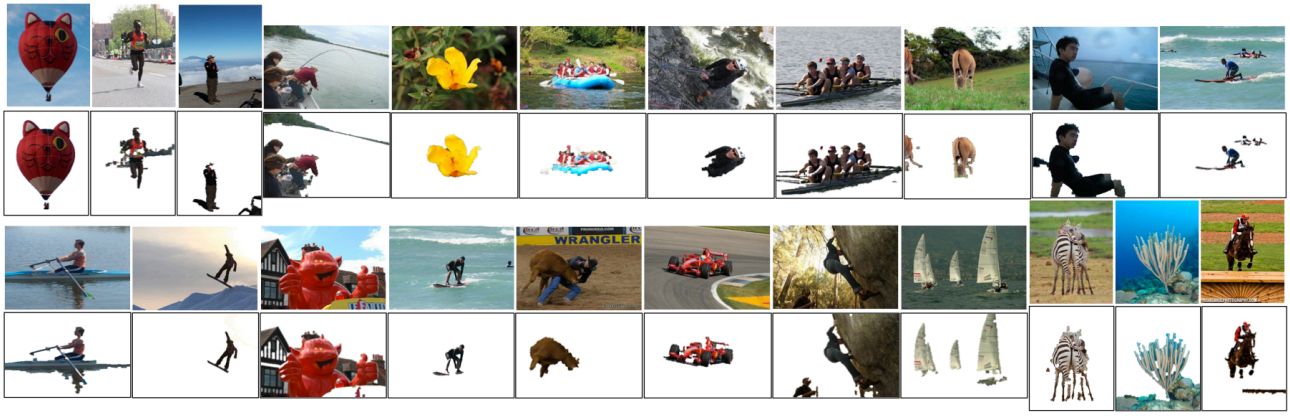


Figure 1. More cosegmentation examples of the Flickr outdoor recreational activity dataset.

rithm to estimate the most likely set of HMM parameters, including the state transition matrix, the observation probability matrix, and the initial probabilities. Given the learned parameter set, we carry out the Viterbi algorithm to find the single best state sequence for each photo stream. That is, all images in $\{P^t \cup \mathcal{P}^r\}$ are assigned to most probable state IDs. Finally, as an estimated timestamp of p, we compute the average timestamps of the training images that share the same state ID with p.

2.2. Application of Our algorithms and Baselines for Cosegmentation Evaluation

In all baselines, we use the source codes provided in the original authors' webpages. The (LDA) is applied to each photo stream separately. For (COS) and (MFC), we first split each photo stream into multiple image groups by K-means on visual features. Then, (COS) and (MFC) are applied to each image group separately. This decomposition improves not only segmentation accuracy but also computation speed.

2.3. More Cosegmentation Examples

Fig.1 shows more cosegmentation examples of 15 outdoor recreational activity classes of our Flickr dataset.

2.4. Preliminary Results of Photo Storylines

In this section, we present very preliminary results of photo storyline construction, some of which are shown in Fig.2. We create these examples using the similar method as described in [3]. As we discussed in section 4.1 of the main draft, we build an image graph $\mathcal{G}_I = (\mathcal{I}, \mathcal{E}_C)$ to facilitate large-scale cosegmentation from the output of photo stream alignment. We first apply the *affinity propagation* [1] to the image graph, in order to detect exemplars and clusters in the graph. Then, we find top five highest ranked clusters in every hour on the timeline. In order to compute the ranking values of clusters, we first obtain the stationary distribution of each node (*i.e.* image) by applying PageRank algorithm to the image graph $\mathcal{G}$. Then, we compute the ranking scores of clusters as the sum of stationary distribution of the nodes in each cluster, which means the portion of time that a random walker traversing the graph stays in the cluster. Finally, each picture in Fig.2 is drawn by averaging the 30 nearest neighbors of each exemplar.

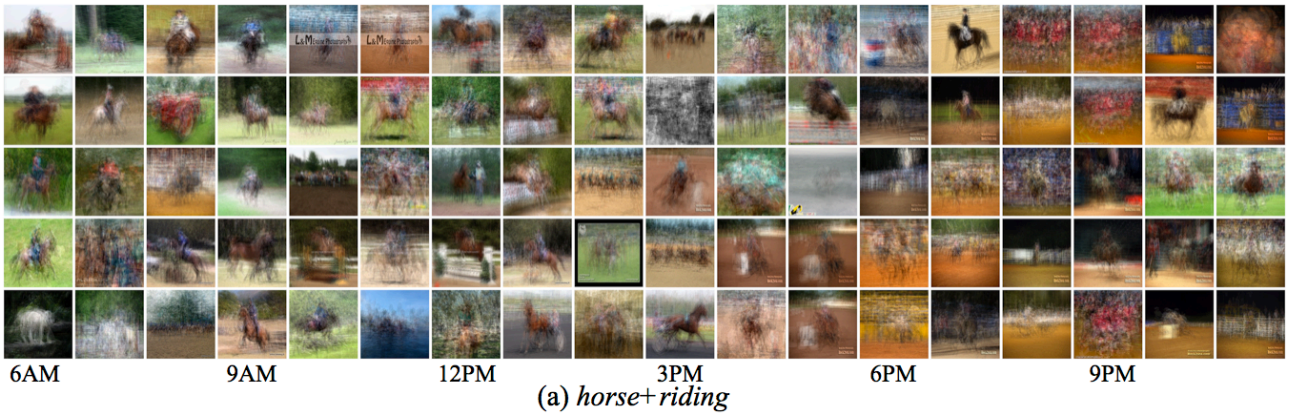
One of our important future research directions is to develop algorithms that can automatically build browsable story graphs from the photo streams of millions of users, and discover the relations between the reconstructed storylines and photo streams of individual users.

References

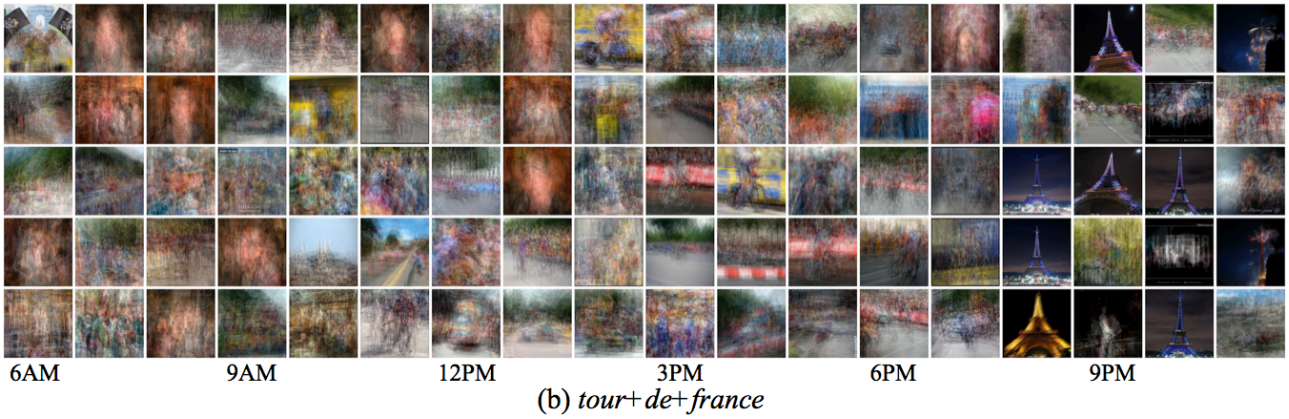
[1] B. J. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315:972–976, 2007. 2

[2] G. Kim, E. P. Xing, L. Fei-Fei, and T. Kanade. Distributed Cosegmentation via Submodular Optimization on Anisotropic Diffusion. In *ICCV*, 2011.

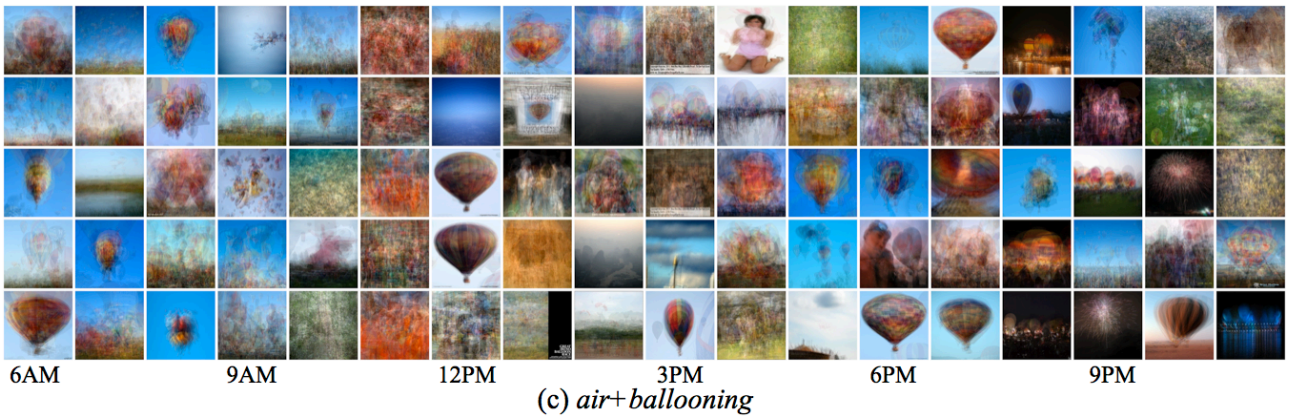
[3] G. Kim, E. P. Xing, and A. Torralba. Modeling and Analysis of Dynamic Behaviors of Web Image Collections. In *ECCV*, 2010. 2



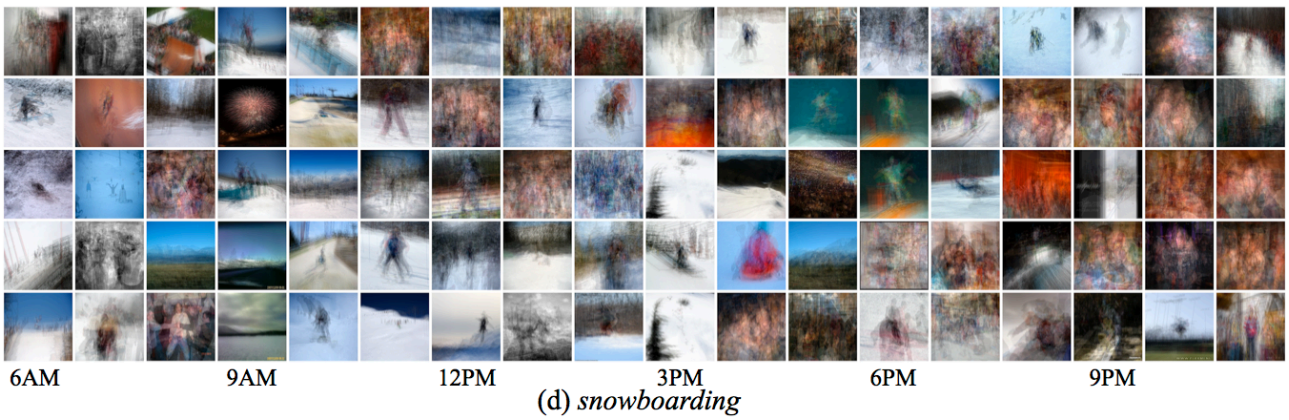
(a) *horse+riding*



(b) *tour+de+france*



(c) *air+ballooning*



(d) *snowboarding*

Figure 2. Examples of preliminary photo storyline reconstruction for three selected activity classes. Top five highest ranked image clusters are shown at every hour on the timeline. Each picture is the average of top 30 highest ranked images in each cluster.