Supplementary File for
“Time-mapping Using Space-Time Saliency”

Feng Zhou
Carnegie Mellon University
http://www.f-zhou.com

Sing Bing Kang, and Michael F. Cohen
Microsoft Research
{sbkang, mcohen}@microsoft.com

Abstract

In this supplementary material, we provide details for the following: (1) multi-scale video segmentation and refinement, (2) STR feature, (3) dynamic programming solution for video re-timing, and (4) computational cost of the proposed system. In addition, we include a video for showing the results on the saliency benchmark dataset and the high-speed video dataset.

1. Multi-scale video segmentation and refinement

To capture region of interests in different spatial scales, we construct an \( l \)-level pyramid of segmentation by adopting different scale parameters that are used in [5]. Each level of the segmentation pyramid corresponds to the result of executing the algorithm [5] with different scale parameters (i.e., the \( \text{min} \) parameter in the command gbh_stream provided by the LIBSVX\(^1\) toolbox). In our work, we set \( l = 4 \) and \( \text{min} \in \{100, 1400, 2700, 4000\} \). The second column of Fig. 1 illustrates an example of segmenting one video in two scales.

The initial segmentation results generated by [5] contained many small and redundant spatio-temporal regions (STRs)\(^2\) that affects the smoothness of the computed saliency map. Therefore, we further refine the segmentation by merging adjacent STRs with similar colors. To do that, we compute the \( \chi^2 \) distance between each pair of adjacent STRs. Any pair with a smaller distance than \( 0.1 \) is merged to a single new STR. For instance, the second and the third columns of Fig. 1 compare the segmentation results without and with refinement, respectively.

\(^1\)http://www.cse.buffalo.edu/~jcorso/r/supervoxels/
\(^2\)The term “voxel” has been used to represent a segmented spatio-temporal region. Since “voxel” is a fundamental term in graphics that has a different meaning, we do not use it to avoid confusion.

2. STR feature

For each STR denoted as \( r_{c,t} \), we compute three feature vectors, one color histogram \( x_{c,t}^{\text{col}} \), and two flow-based descriptors \( x_{c,t}^{\text{mag}} \) and \( x_{c,t}^{\text{ori}} \).

To compute \( x_{c,t}^{\text{col}} \in \mathbb{R}^{d_{\text{col}}} \), we quantize the four color channels (L, A, B and hue) into 8, 16, 16, and 4 bins respectively. The dimension of the color histogram is \( d_{\text{col}} = 8192 \), where \( 8192 = 8 \cdot 16 \cdot 16 \cdot 4 \).

To compute \( x_{c,t}^{\text{mag}} \in \mathbb{R}^{d_{\text{mag}}} \), we uniformly quantize the flow magnitude into \( d_{\text{mag}} = 16 \) bins.

To compute \( x_{c,t}^{\text{ori}} \in \mathbb{R}^{d_{\text{ori}}} \), we follow the idea of the HoF descriptor [4] by quantizing the flow orientation into 8 bins and use magnitude for weighting. An additional zero bin is added to account for pixels whose optical flow magnitudes are lower than a threshold. The final dimension of the descriptor is \( d_{\text{ori}} = 9 \).

3. Dynamic programming solution for video re-timing

In the paper, we formalize the re-timing problem as minimizing the following sum of least-square errors:

\[
\min_{p} \sum_{i=1}^{m-1} (a_i - \bar{s})^2 + \lambda \Delta z.
\]

Due to its additive nature, Eq. 1 can be globally minimized through dynamic programming (DP) following the recursive Bellman’s equation [2],

\[
J_{i,j} = \min_{k=i}^{j} J_{i-1,k} + (a_i(s_{i,k,j}) - \bar{s})^2 + \lambda \Delta(s_{i,k,j}),
\]

where the cost-to-go value function, \( J_{i,j} \), represents the remaining cost ending at \( i \)-th step to be incurred for the subsequence, \( s_{1,j} \).

4. Timing statistics

Fig. 2 summarizes the average time cost of our system for processing the ten high-speed videos, whose video statistics are summarized in Fig. 2a.
Figure 1. An example of multi-scale video segmentation and refinement.

Figure 2. Time cost averaged over 10 high-speed videos. (a) Video statistics. Comparison on different components in (b) computing motion saliency, (c) video re-timing, and (d) temporal filtering.

Our system was implemented on a PC platform with 3.6GHz Intel Xeon and 16GB memory. The code for the system is primarily written in Matlab, except for the computation of optical flow [3] and video over-segmentation [5], where they were implemented in C++. As shown in Fig. 2b, our system takes on average ten seconds to compute the saliency map for each frame. In the re-timing step (Fig. 2c), both DP and smoothing methods take a few seconds per video, in contrast to the more significantly higher computational cost using BM [1]. To render each frame (Fig. 2d), the new BoxA filter takes less time than the normal box filter because it integrates fewer frames for the duration of the salient moment. The SalBlur filter takes much longer to compute. However, it can be largely sped up in a parallelized implementation because most steps are pixel-independent; this is future work.

References