# Hyperlapse Creation via Consideration of Moving Objects

Gyutae Ha

Yonsei University

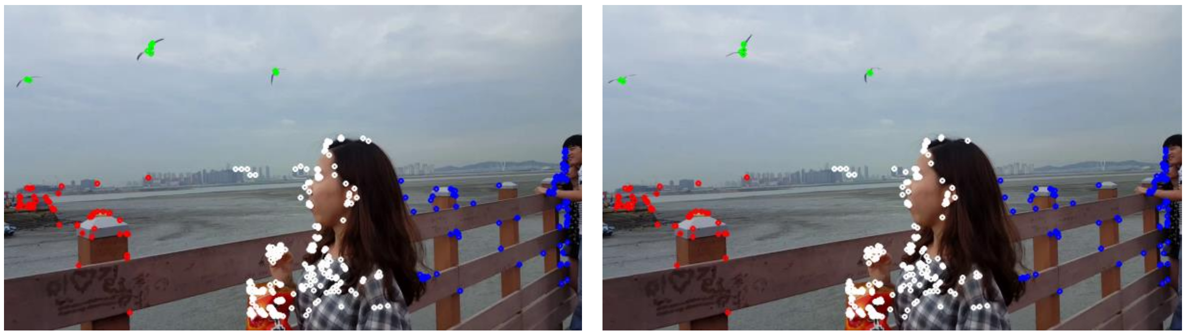

**Fig 1. K-means clusters of key points in frame i and clusters of key points in frame j which are matched to frame i key points. By comparing the position of clusters between frame i and j, the algorithm measures movement of main objects in scene and correspondence between frame i and j.**

## Introduction

*Hyperlapse* is a technique that makes the video play fast by dropping number of frames. In most *hyperlapse* videos, they contain dynamic camera moving. Since camera is not fixed, jitter of video is the main problem of *hyperlapse* [**1**].

A research [**1**] suggests the method of creating *hyperlapse* that selects optimal frame. In the paper, frame set that makes smooth transition between frames in result video and achieves target speed-up is called 'optimal frame set'. One of the algorithm the paper suggests, calculates the cost matrix between adjacent frames including *homography* and gap of the two different frames and find optimal path based on the cost matrix.

The main problem of optimal frame selection *hyperlapse* [**1**] is that it cannot catch the movement of main objects in the scene, because they only consider the overall correspondence between two frames in the algorithm. For the reason, the video created by the algorithm shows awkward movement of moving objects sometimes.

Solving this problem, this paper proposed a frame selection algorithm that calculate objects movement by k-means clustering of matched points and apply it to cost. [**Fig 1**]

## Frame Selection Algorithm

For finding optimal path, this algorithm is divided to few steps. [**Fig 4**]

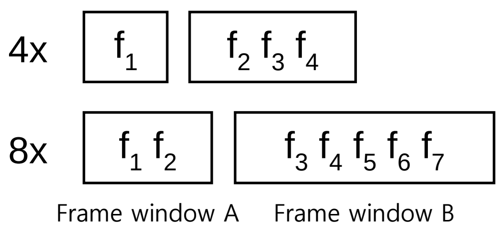

**Fig 2. Frame windows**

First, set two frame windows containing frames to be compared [**Fig 2**]. In 4x case, fill 'frame window A' with $f_1$(first frame) and fill 'frame window B' with $f_2$, $f_3$, and $f_4$ sequentially.

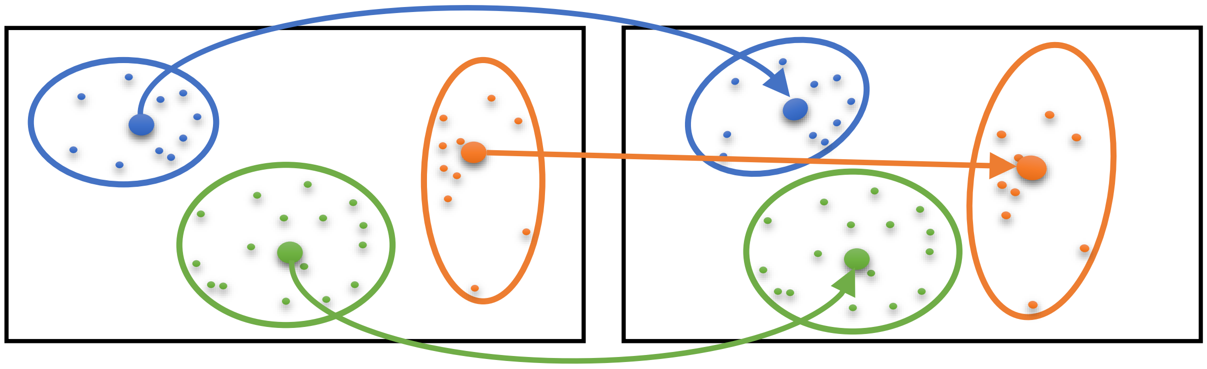

**Fig 3. Key points cluster from frame i and its movement to frame j.**

Second, after filling the windows, compare each pair, ($f_1$, $f_2$), ($f_1$, $f_3$), and ($f_1$, $f_4$), calculate cost of each pair and fill sparse matrix for accumulated cost. When comparing each pair and calculating cost, this algorithm exploits k-means clustering and *homography*.

① Run k-means clustering on key points (feature points driven by ORB algorithm) that are from first frame in 'frame window A'. (frame i)

② Find key points from first frame in 'frame window B' (frame j) that are matched (use ORB descriptor and BF Matcher) to key points from process ①.

③ Measure movement and standard deviation (points distribution based on center point) of the clusters and calculate the cluster cost ($C_c$) between the two frames based on the measured factors. [**Fig 3**]

④ Using *homography* between the two frames, calculate the *homography* cost($C_h$).

Third, after second process, pop $f_1$ from 'frame window A', pop $f_2$ from 'frame window B', push $f_2$ to 'frame window A', push $f_5$ to 'frame window B'.

Fourth, repeat these processes (second and third processes) to end of frame sequence. Since we should drop about 2~3 frames per 4 frames for creating four times fast video, we can find optimal path using the accumulated matrix and DTW algorithm [**1**].

```
Input : speed, frame sequence
Initialization :
  for $i$ = 1 to $fw_b$ size do
    for $j = i + 1$ to $i + fw_b$ size do
      $D_V(i, j) = C_c + C_h$
    end for
  end for
First pass : populate $D_V$
  for $i = fw_b$ size to $T - fw_b$ size do
    for $j = i + 1$ to $i + fw_b$ size do
      $c = C_c + C_h$
      $D_V(i, j) = c + min_{k=1}^{w}[D_V(i-k\ ,\ i)]$
      $T_V(i, j) = argmin_{k=1}^{w}[D_V(i-k\ ,\ i)]$
    end for
  end for
Second pass : trace back min cost path
  $(s, d) = argmin_{i=T-2*fw_b size,\ j=i+1}^{T-fw_b size,\ i+fw_b size}[D_V(i, j)]$
  $\boldsymbol{p} = \langle d \rangle$
  while $s > fw_b$ size do
    $\boldsymbol{p} = prepend(\boldsymbol{p}, s)$
    $b = T_V(s, d)$
    $d = s$, $s = s - b$
  end while
Return: p
```

**Fig 4. Algorithm pseudo code.**

**$fw_b$: Frame window B**
**$C_c$: Cluster cost**
**$C_h$: Homography cost**
**$D_V$: Accumulated cost matrix**
**$T_V$: Trace-back matrix**
**p: Result frame set**

## Results (Limitation & Problems)

- Transition between frames is not smooth. For this problem, this algorithm needs stabilization process [**2**].
- Setting small size window cannot make quite difference with naïve method (ex. method that simply drop a frame per four frame in 4x *hyperlapse*).
- When key points are not on main objects edge, this algorithm cannot work well. (Ex. In high illuminance frame) [**Fig 5**]

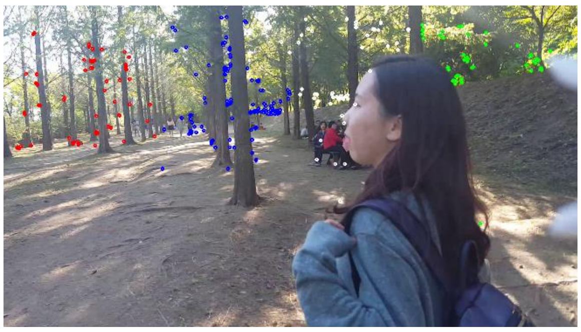
**Fig 5. Clusters of key points in high illuminance frame.**

## Future Work

- Time complexity optimization of the proposed algorithm.
- The proposed algorithm is required to consider the outlier of each cluster to remove the negative effectiveness.
- The weighted factor should be applied to evaluate the priority of the each clusters based on the position (central, sub-central, outer position) to enhance the relationship of frames.

## Conclusion

This paper proposed the frame selection algorithm to improve the smoothness of the object movement in *hyperlapse*. Proposed algorithm tried to measure the cost of frame transition by calculating *homography* and translation of k-means clusters of key features between two frames. However, this algorithm didn't show advanced performance comparing the naïve method and the method proposed in paper [1]. For improving algorithm's performance, video stabilization, weighted factor on each clusters, and time complexity optimization should be applied to the proposed algorithm.

## Reference

[1] Joshi, Neel, et al. "Real-time hyperlapse creation via optimal frame selection."*ACM Transactions on Graphics (TOG)* 34.4 (2015): 63.

[2] Hansson, Björn, and Kim Tengbom Zetterman. "Video Stabilization Algorithm from Low Frame Rate Video for Hyperlapse Applications." *Master's Theses in Mathematical Sciences* (2015).