COMPGV15 Project 2 Report

Student Name: Yifei Gao Student ID:13028702

Key Idea

Construct a time-lapse video from from a long drawing footage. Show motion trail of drawing hands and pens. This idea comes from Bennett and McMillan's paper of Computational Time-Lapse Video[1].

Background reading and plans

Firstly Bennett and McMillan's paper [1] suggested a very clear and simple workflow of building a time-lapse video, which consist of two main components: Sampler and Virtual Shutter. The role of the sampler is to identify key frames that characterized or summarized a session of the video, where we usually downsampling thousands of frames into a few key frames. The simplest way to implement this sampler is to sample the video uniformly, where the sampling rate only depends on the ratio between the number input frames and output frames. But such uniform sampler will discard some important informations between the sampling interval, resulting in objects suddenly appear or disappear in the output sequence, which is described as *popping* in the paper. They suggest that by using a non-uniform sampler we can reduces the popping artifacts but the selected frames trends to cluster together in a close temporal. So there is a tradeoff between the quality of clustering and the amount of popping artifacts in sampling. Figure 1. Shows the sequences of three frames with uniform and non-uniform sampling.

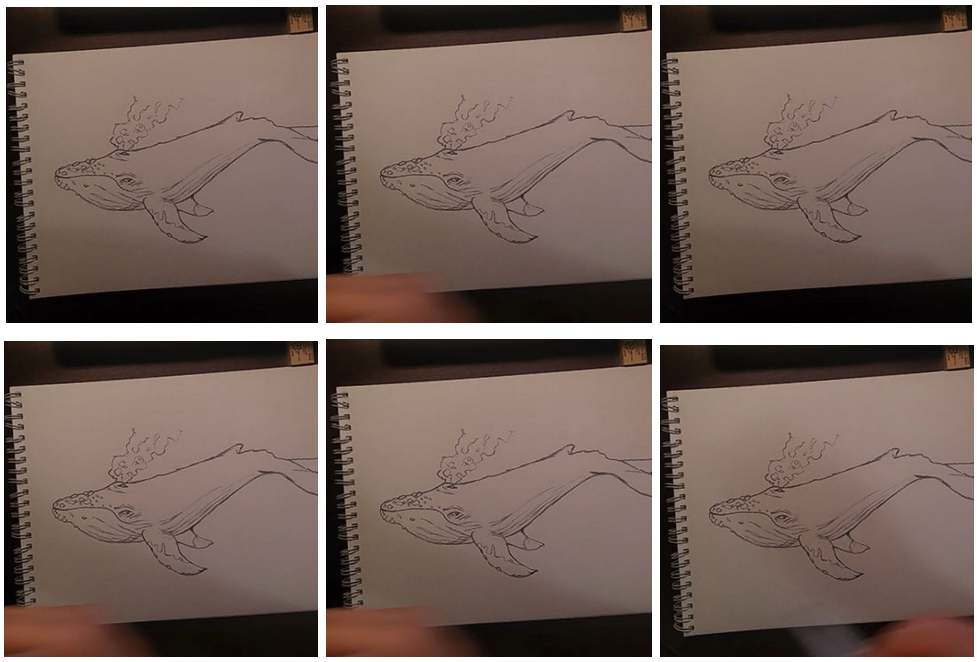


Figure 1. Sampling with uniform and non-uniform sampling.
(1st row: Uniform sampling; 2nd row: non-uniform sampling; Left to right: n-1,n,n+1 frames)

The role of virtual shutter is very self-explanatory, as the video-rate exposure from the input frames is not long enough to collect all the scene elements and motions, so we need to apply a post-process to create a exposures that simulates the effect of using a longer shutter. The paper suggest a sliding windows approach for generating different virtual shutter for different scenes, they are designed to capture the most representative pixels within the window, for example maximum virtual shutter to capture highlights in nighttime videos and minimum virtual shutter to remove bright foreground phenomena.

My initial plan was to combine the uniform and non-uniform sampling to creates a new sampler that select a fixed number of frames non-uniformly within sessions that's been generated uniformly. This should increase the performance as a session will have a smaller input frames size, where the computation of pairwise error and searching is cheaper. To enhances the motion trails in the hand and pens, the virtual shutter needs to have a weighting function that can allow me to control the contribution of the pixel value at a different time (so it appears to fade out).

Data capturing

The source video is captured using my phone fixed on a tripods, the auto-focusing is turn off during the capturing. The video is capture at night time, so the background lighting only comes from the stationary white LED light and desk lamp in my room . I noticed there are some light intensity change from the desk lamp (due to Voltage fluctuations), which lead to some strip artifacts while capturing. But these strip artifacts are not found in the video and the video-rate frames, which I think might be corrected by the sampling algorithm in the phone. The video is capture at 30fps with 720p resolution, and its original duration is 1 hr 18 mins 53 secs. I've used Photoshop to downscale and downsample (for computation) the video into sequence of frames, 47330 frames are generated as input data frames in the end.

Implementation

Sampler

This part of the implementation is included in 'select_key_frames.m' function in my code folder. We firstly need to compute a distance matrix for all the frames, but as there are 47330 frames this means we need to compute a matrix of size 47330x47330 which is too large and computationally expensive. Therefore we manually divide them uniformly into 100 frames sessions, so we'are only computing the distance matrix of the frames in the session this could avoid the result sampling being cluster together in a close temporal. The distance matrix is calculated as the sum-of-squared difference between two frame across all channels, this distance is weighted normalized by a penalty weighting. I used a very simple linear weighting function,

$$W(i,j) = |\frac{j-i}{N_{in}}|$$

, where W is the weight, i and j are the frame index and N is the total number of input frames in the session. Then we can normalize the row-wise sum of the distance matrix by the row-wise sum of weighting to get a measure of the average cost of current frame to all other frames. At this stage we would have a 1xN array of cost that showed in the first graph in Figure 2.

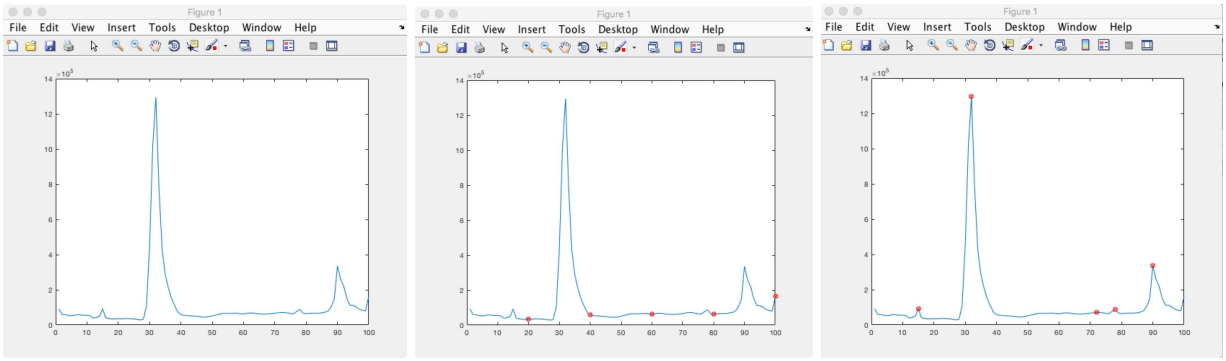


Figure 2. Row-wise distance graph
(Left to Right: Raw graph; Uniform sampling; My approach)

Red circle in the line graph identified the sampling points selected by different approach. My approach is to sample a fixed number of peaks values in the row-wise distance sequence, as they identifies the index of the frames that has the largest change over surrounding frames. You can see that uniform sampling missed a lots of peak frames in the sequences. Figure 3. shows the visualization of the sampling results of 3000 frames from the video.

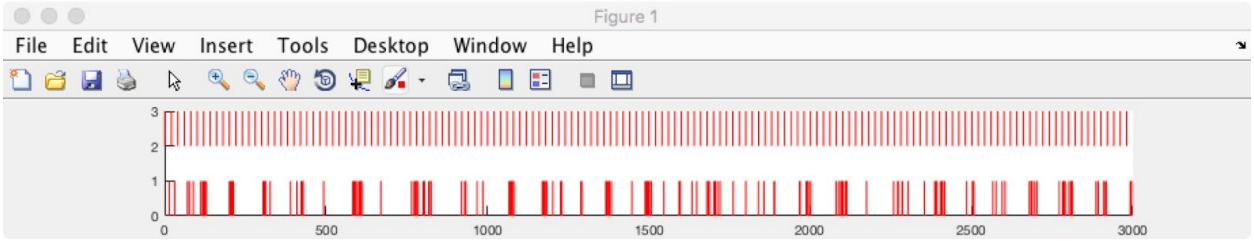


Figure 3. Resulting sampling frames
(Top: Uniform sampling; Bottom: My approach)

Virtual Shutter

I took a different approach of building the virtual shutter, usually the exposure duration is fixed and we're virtually rendering the virtual exposure image during the time window. But the length of the duration needs to be tuned to get a ideal result, a typical visual shutter model could be see in Figure 4.

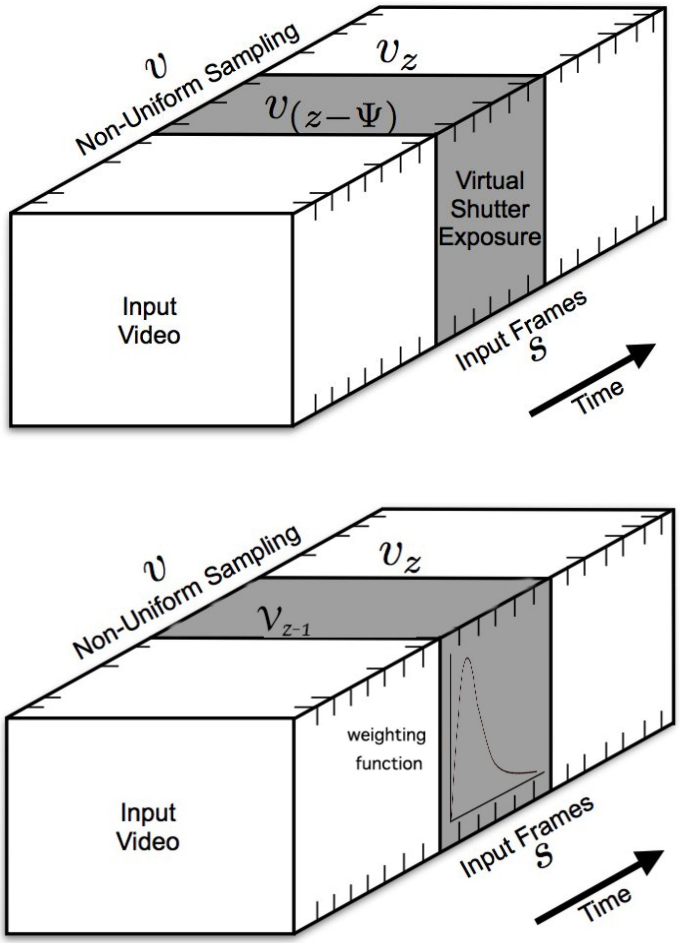


Figure 4. Illustration of the sliding window virtual shutter and My approach
(Top: Sliding window approach; Bottom: My approach
Top figure taken from Bennett and McMillan's paper [1], bottom figure is modified based on it.)

The key idea of my approach is to dynamically change the exposure time based on the interval between current and last non-uniform samples, so the elements over a large interval could be covered and we will not repeatedly computing the the exposure when the sampling is dense. There also an adaptive weighting function that controls the contribution of older and more recent samples, the weighting function I've used in the implementation is:

$$w(f) = \zeta^{\kappa},\ \kappa = \mu \cdot \frac{Vz - f}{Vz - S_{z-1,z}},\ S_{z-1,z} = \sum_{i=V_{z-1}}^{V_z} s_i$$

s_i is the value of the pixel at input frame index i, where μ and **ζ** are parameters that controls how the weighting function falloff as the sample gets more recent. The shape of the weighting

function should appears as it shown in Figure 5. In general, where it assign larger weight to older frames and gradually fallout as it get closer to current frame z.

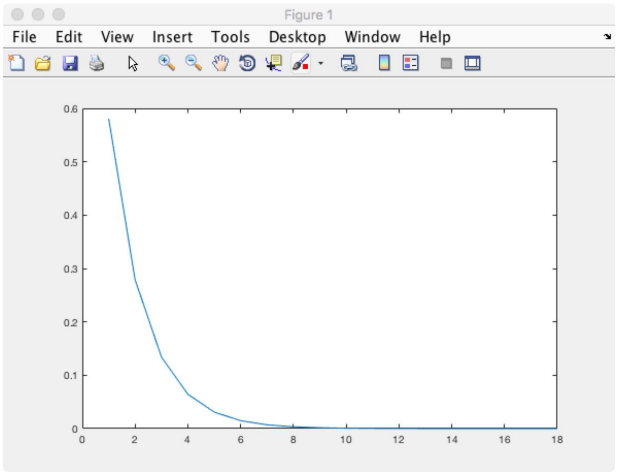


Figure 5. Shape of the weighting function

This function is designed to weight the oldest frame more heavily than others, so we can replace this oldest frame with other frames to add additional information or manipulate the virtual shutter characteristic. I've used it to maintain the motion trails of adjacent sessions, where the first frame in session n is replaced by the last frame of virtual exposed session n-1. The result of my virtual shutter implementation is shown in Figure 6.

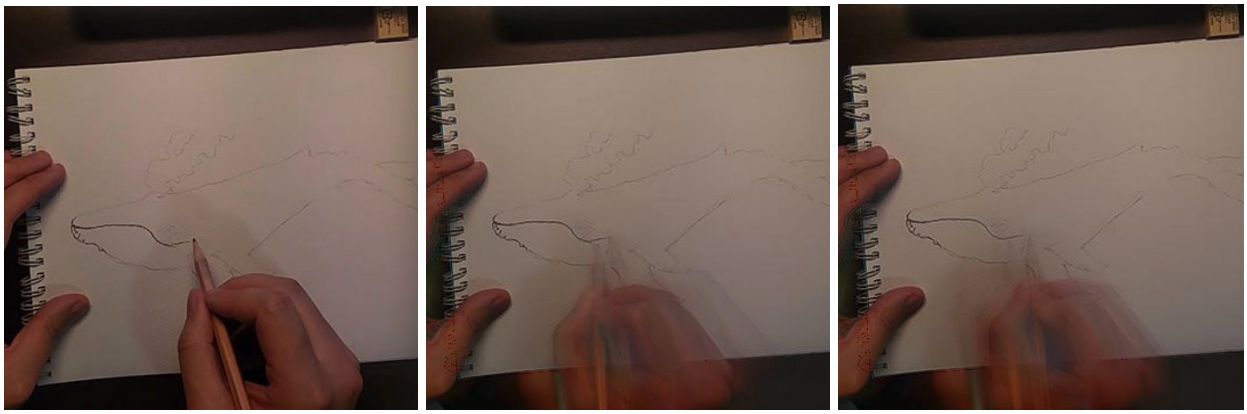
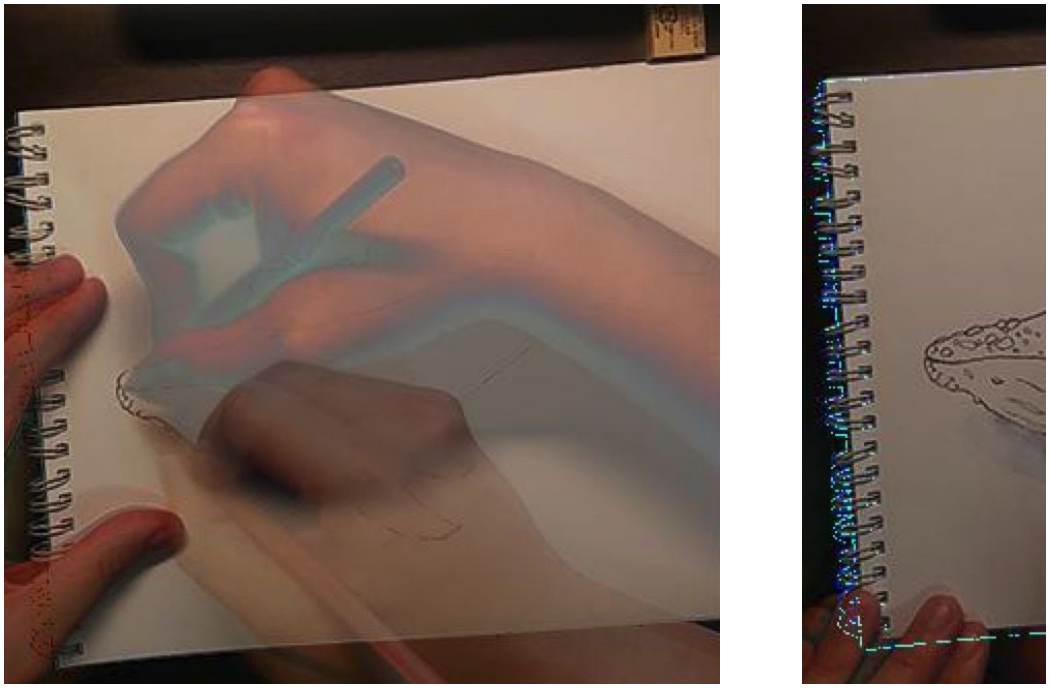


Figure 6. Result frames with virtual shutter
(Left: Raw frame; Mid: μ =30, **ζ**=0.65; Right: Mid: μ =30, **ζ**=0.90)

But this virtual shutter also introduced some artifacts into the result frames, it appears when the the sampling within a session is selected very closely and the motion change between current and last session is very large. The colour channels becomes inconsistent with each other as they are being processed separately, resulting in the motion trails appears to contain higher blue channel intensity than expected. I've also find some blue outline appears on the edge of the paper, which I've found out that is caused by the image formatting error, this only appears

when I've tried to store the jpg image into png format. This is more of an error of the matlab framework than the error of the virtual shutter. The error examples is shown in Figure 7.



Figures 7. Error found in the result
(Left: inconsistent colour channel on the motion trail;
Right: Image formatting error lead to blue outline on the edges)

Result Video

The 1 hr 18 mins 53 secs original video is processed into a 1 min 18 secs time-lapse video in the end, which could be viewed in https://www.youtube.com/watch?v=UhVxr88waUc&index=1&list=UU1azpn3HGphuTMQjgFnd3VA .

What to improve?

I think the sampling I suggested still trends to clustering frames in a close interval, In the article [1], they suggested to enforce the sampling to uniform by adding penalty of closer frames. I think I could modify the algorithm so it stack this penalty values so when it reaches a threshold it drops current frame and choose a frame that is closer to principle uniform frame.

There could be a queue of frame that I could update as the reference frame of next session in the virtual shutter. My original intention was to assign a decay rate to each frames in this queue, by alternating this decay rate of older frames and recent frames of previous session, we could make the drawing starts to disappearing as time pass.

There could also be a tempel mapping of the tip of the drawing tool, so the pixel around that patch would have higher weight.

The quality of the result is poor as I had to reduce the size of the original image to ½ for it to be processed in a reasonable time.

Reference

1. Eric P. Bennett , Leonard Mcmillan, Computational Time-Lapse Video, University of North Carolina at Chapel Hill, Chapel Hill, NC, 2007.