CSCI 1430 Final Project Report: HUSTL (Hyper-Ultra-Super-Time-Lapse)

Team HUSTL: Jiaju Ma, Michael Yicong Mao, James Li.
Brown University
9th May 2019

Abstract

We present a three-stage software pipeline that makes it easier for people to create quality hyperlapse videos. Our algorithm takes in either a series of photos shot individually or a video input. If the input is a video, our pipeline will extract optimal frames from the video through reducing a cost matrix (first stage). On the second stage, input images will be color corrected based strongest SIFT features of all inputs. This ensures that all images have a cohesive color (white balance and tone) and exposure. Finally, the pipeline takes in the processed images and conducts camera path stabilization on them. Our approach can assist users to produce hyperlapse videos of good quality (consistent color and stabilized shot) with minimal effort.

1. Introduction

Hyperlapse is a delicate art. Making professional hyperlapse videos requires precise and consistent step distance between each photo and skilled camera alignment with DSLRs on tripods. It is usually very cumbersome and time-consuming. A 20 second hyperlapse may take a photographer more than 2 hours to shoot.

Aside from effort and skills in shooting the photos, there are many technical challenges in post production. One such challenge is the compensation of insufficient or excessive exposure when moving between environments with different lighting conditions. Another obvious challenge is the compensation for unwanted camera movement. To deal with theses challenges, photographers often need to made the adjustment and compensation on a frame by frame basis which could become a disaster if there are more than hundreds of frames to edit.

To ease the process of hyperlapse creation, we propose and implement a software pipeline composed of three stages that use computer vision algorithms. The pipeline takes in either images or videos as input, and outputs a hyperlapse video of acceptable quality. If the input is a video clip, we would run frame selection on the video and send the selected frame to the next stage. If the input is images, the images are sent directly to the second stage. The second stage takes in a sequence of images and matches the color tone, white balance, and exposure of all inputs. The results are fed into the final stage, where the image sequence is stabilized through perspective warping and eventually recreated as a hyperlapse video.

2. Related Work

In our search for research on hyperlapse videos, we did not find many paper that directly matches our problem statement. In the end, we combined ideas from multiple related papers to form our unique pipeline.

Optimal Frames Selection

Frame selection is at the heart of of making hyperlapse video. Many popular video editing tools, such as Adobe Premiere, use a naive approach that selects frames uniformly at random. There are also hardware-based approach that relies on shutter and gyroscope information to select and warp frames that yields the best stabilization results [1]. However, in the space of software-based approach, there isn't much research into frame selection aside from the paper by Microsoft Research [6].

Color Consistency Across Frames

To ensure the coherence of the hyperlapse videos, we need to adjust the images so that they share a common color tone and similar white balance and exposure level (gamma value). HaCohen et al. [4] proposed an Non-Rigid Dense Correspondence (NRDC)-based method that optimizes color consistency in a collection of photos. However, their method is too computationally intensive and needs a large amount of data to train. Park et al. [5] proposed a more efficient method based on SIFT feature matching and observation matrix construction, which requires no training at all and is computationally much cheaper. We adapted Park et al. [5] for our purposes.

Video Stabilization

There is no paper that specifically refers to stabilization of hyperlapse created with individual photos. In a paper by Joshi et al. [6], they used normal video stabilization for their hyperlapse video generated from video. In their generated video, the stabilization result seems good enough, so we decided to use stabilization algorithms for normal video.

We found a 2013 paper by Liu et al. [7] that proposes a novel way to smooth camera motion and reduce distortion introduced by warping. A simplified version of this method was also used by the paper by Joshi et al. [6]. This led us to believe that such a method would be sufficient for hyperlapse video smoothing.

3. Method

Software Used

We used the following languages and libraries: Python, NumPy, Sci-Kit Image, Sci-Kit Video, OpenCV, Cyvlfeat, (Python wrapper of MATLAB's VLFeat library), MATLAB, Computer Vision Toolbox. In addition, MACE (MAximal Clique Enumerator) [9], a C program that finds the maximal clique within a graph, is used as part of the process to implement the method proposed by Park et al. [5]

Optimal Frames Selection

This stage aims at selecting the optimal frame path that renders the smoothest camera movement and the most consistent frame rate in the output video. The implementation is adapted from [6] and consists of three steps–frame matching, cost building, and frame selection.

In frame matching, we first extract SIFT features from each frame. Then, we find matching feature pairs between frames and calculate the homography matrix H between each frame in a given window size.

In cost building, we compute the alignment cost 1 and the overlap cost 2 between each frame. Together, they make up the motion cost 3 that measures the image similarity between each frame.

$$C_r(i,j) = \frac{1}{n}\sum_{p=1}^{n} ||(x_p, y_p)_i^T - H(i,j)(x_p, y_p)_j^T||_2 \quad (1)$$

$$C_o(i,j) = ||(x_c, y_c)^T - H(i,j)(x_c, y_c)^T||_2 \quad (2)$$

$$C_m(i,j) = \begin{cases} C_o(i,j) & C_r(i,j) < \tau_c \\ \gamma & C_r(i,j) \geq \tau_c \end{cases} \quad (3)$$

On top of motion cost , we also take the speed in which the optimal path travels into account. The speed cost is made up of velocity cost 4 and acceleration cost 5. By penalizing sudden jumps and incoherent frame rate, we yield a more consistent and smoother frame sequence.

$$C_v(i,j,\upsilon) = min(||(j-i) - \upsilon||_2^2, \tau_v \quad (4)$$

$$C_a(i,j,\alpha) = min(||(j-i) - (i-\alpha)||_2^2, \tau_a \quad (5)$$

$$C_s(i,j,\upsilon,\alpha) = C_v(i,j,\upsilon) + C_a(i,j,\alpha) \quad (6)$$

Finally, we employ a dynamic programming algorithm to find an optimal path that minimizes the transition cost 7 from frame to frame.

$$C(i,j,\upsilon,\alpha) = C_m(i,j) + C_s(i,j,\upsilon,\alpha) \quad (7)$$

Color Consistency Across Frames

This stage of the pipeline takes in a series of images, which can be photos taken with a camera, or frames extracted from a video by the first stage of the pipeline (optimal frames selection). Color adjustments (white balance, color tone, and gamma) are applied to all images through a global color correction model. The method we used in this stage is adapted from [5], based on their MATLAB implementation [8]. Python and Cyvfleat are used so that SIFT feature extractor is available to us (it is not available in OpenCV 3).

In our implementation, we firstly extract SIFT features from each input image. We randomly sample a certain amount of features (1000-2000) from all extracted to reduce computational cost. A bi-directional matching of feature points is performed for each input image pairs. The matches are then post-processed by removing non-unique pairs. An undirected match graph $G = (V, E)$, where each vertices in V is a SIFT feature and each edge in E represents a match, is constructed. We then use MACE [8] to find maximal cliques of size 2 and above.

Then, color patches are extracted from images based on SIFT features that are part of the correspondences found in the maximal cliques. These patches are put together to construct an observation matrix I such that

$$I = C + A + E \quad (8)$$

where C is the color coefficient matrix, A is the albedo matrix, and E is the residual matrix. To further process the observation matrix before applying the color adjustments to images, we used a technique called Factorization-Based Low-Rank Matrix Completion proposed by Cabral et al. [2]

Finally, we apply the post-process matrix I to all input images to achieve coherent color consistency.

Video Stabilization

We adapted the stabilization algorithm from this paper by Liu et al. [7]. The paper proposed a method of splitting the image into sub-sections and smooth the camera path of each sub-section. Then they used quadratic functions to calculate a smooth path for the stabilized footage. The images then goes through As-Similar-As-Possible warping [3] with shape preservation. This creates new frames that matches previous frames.

The camera path is estimated by the product homography between corresponding sub-sections in adjacent frames. This

provides a quick way to calculate path without calculating the fundamental matrix and the relative camera positions.

In our implementation, we used Python MATLAB engine provided by MATLAB, opencv, scipy, numpy and scikit-image. Due to the paper using As-Similar-As-Possible warping, which by itself is very difficult to re-implement, we decided to use part of the MATLAB code written by SuTan-Tank on GitHub.

The algorithm would split the image in to a grid mesh of i cells. For each cell, every point is represented as a bilinear interpolation of the edges of the cell. Then, the same cell is matched on the next image, and a homography matrix $H_i(t)$ is calculated for cell i at time step t. Then, the path of each cell over time is calculated as

$$P_i(t) = \prod_{m=0}^{t} H_i(m) \tag{9}$$

$H_i^{(0)}$ is the original camera pose, which would be interpreted as a matrix of 1s.

This path is smoothed over by optimizing the following term

$$\mathcal{O}(\{P(t)\}) = \sum_t (||P(t) - C(t)||^2 + \lambda_t \sum_{r \in \Omega_t} w_{t,r}(C) \cdot ||P(t) - P(r)||^2) \tag{10}$$

Ω_t is the neighborhood of t, and $w_{t,r}$ is the weight to preserve weight discontinuities in panning and transitions.

Then, using the smoothed path, we get optimized homographies $\{\hat{H}_i(t)\}$ that we can use to perform As Similar As Possible warping [3] with paddings on the outside. Then, the padded image is cropped to get stable footage.

In our implementation, we used SURF as features, rather than minimum eigenvalue corner points as in the MATLAB code. We extensively experimented on the hyper-parameters and found a generally well-behaved set of hyper-parameters on multiple sets of our data.

4. Results

Result Videos

1. Main Green Video
    - Baseline video
    - Result video
    - Video showing Warping
2. Arch Video
    - Baseline video
    - Result video
    - Video showing Warping

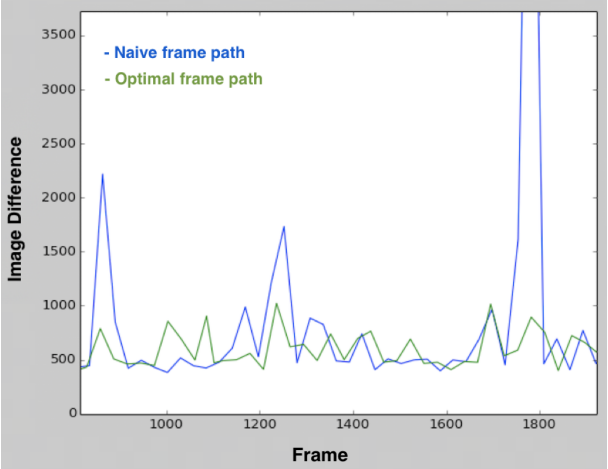

Figure 1. Image Difference

Optimal Frames Selection

To assess our results, we also output a sequence of frame images selected uniformly at random as the benchmark. Then we compare the frame sequence from naive approach and optimal approach to adjust our hyperparameters for the best performance. Then, we measure the image difference 2 between consecutive frames in both naive and optimal approach. As illustrated in figure 1, the optimal path yields higher overlaps between consecutive frames than the naive path for most of the times with some outliers here and there. We believe these outliers are contributed by moving objects in the input video, such as walking pedestrians in front of the camera, that results in necessary image difference and cannot be avoided. Overall, the selection process is able to identify the most cost-effective frame sequence that is consistent with the target frame rate for the final output.

Color Consistency Across Frames

We fed two datasets into this stage of the pipeline - a series of photos depicting a movement from the interior of Brown's Salomon Hall to the Main Green and frames extracted from a video taken by a camera travelling through the Main Green.

The photo series has a sudden change in exposure (indoor-to-outdoor transition) and color tone. Our method in this stage is able to both brighten the underexposed images and keep a cohesive color tone across images. Selected input images from this photo series and corresponding results are shown in Figure 2

The extracted Main Green frames have a consistent color tone and exposure level, except for those taken under the Faunce Arch, where low light caused the images to appear darker than rest of the frames. Our implementation is able to significantly brighten the underexposed images to the same level as rest of the extracted frames (Figure 3).

Video Stabilization

We tested the pipeline on a series of images of walking towards the engineering building, a series of images walking towards the arch between Metcalf and Caswell, and a series of images extracted from a video walking through main green. Some of the other data that contains panning was not usable to test the algorithm due to the panning only containing less than 10 frames and would be less than half a second in actual video speed. This resulted in the feature points moving out of the mesh cell so the RANSAC algorithm used to fit homography is encountering issues.

On our series of images of the engineering building, the original series of photos are relatively stable, and there are only a few people people passing by. This is a fairly easy test and our method is successfully able to separate the background from people walking by and create a stable background. See Figure 4.

In our test of walking towards the arch between Caswell and Metcalf, there is a slight horizontal shift of the camera. Our method successfully identified that and warped the image so that it looks as if no shift has occurred. See Figure 5.

In our test of walking towards Faunce Arch on the Main Green, there are people walking towards and past the camera, which makes tracking more difficult than previous test data. After increasing number of iterations optimizing the camera path, the results become fairly acceptable in the first half of the series, but the second half remain sub-optimal. We suspect this is caused by the extreme shift of tracked points as the camera approaches and enters the arch, which messes with the tracking algorithm. See Figure 6.

4.1. Discussion

In the first stage **Optimal Frame Selection**, we are able to remove most of the camera jiggling from human movement. However, since the camera is handheld and we were moving on foot when the video was taken, the camera jiggling was quite consistent in a left-right-left-right movement. Thus, the uniform random sampling from the naive approach functions as a frequency pass that was also able to filter out more than half of the jiggling due to foot steps. We believe that a more diverse database that includes more inconsistent camera jiggling, i.e. videos taken on bikes or in cars could show a more dramatic contrast between the naive and the optimal approach.

For the second stage **Color Consistency Across Frames** of our pipeline, we are able to achieve what we have planned to do. Given a series of images or extracted frames as input, we are able to efficiently compute a color correction matrix (I) that ensures all inputs have similar values of color tone, gamma, and white balance. However, we rely on an external C program (MACE [9]) compiled in Windows 10 environment, which can be burdensome if we want our pipeline to run in other OS. Moreover, our method is not "smart" enough to determine which color standard we want all the input images to adhere to. For example, if the majority of the input images are underexposed, then the program will try to brighten all the images, causing the normally-exposed images to appear overexposed. This can be alleviated by increasing the number of SIFT features kept, but doing so would also drastically increase the computational cost of the algorithm.

In the third stage **Video Stabilization** of our pipeline, we are mostly able to get good result on forgiving footages. We discovered that if the sequence of images contain fast side-to-side motion or when a new scene is entered, the stabilization algorithm doesn't perform as well, or even refuses to work properly (RANSAC not having enough matches). We suspect that this is caused by the tracking algorithm having a tendency to refuse tracking fast-moving feature points. This caused it to lose track of many features when dealing with fast-moving sequences of images, or when approaching and entering narrow passageways.

Another issue is distortion. It is still difficult to control distortion in our result, even though the original paper [7] claims better distortion control than other methods, we find that even with extensive hyper-parameter tweaking, distortion is still visible. To achieve lesser distortion, we may use perspective warping that causes less distortion. We are so far unable to find one such algorithm.

A solution to achieve better results may be to utilize the camera lens information to construct a 3d representation of the camera path and orientation, and smooth the camera path according to that. This may achieve better results, but would incur a lot more calculation.

5. Conclusion

As shown in **Results** section, our three-stage pipeline is able to greatly improve the quality of the hyperlapse video in terms of frame selection, color consistency, and video stabilization. The input was shot handheld. We did not use any physical stabilization method or carefully align photos. Our results show that our implementations allow people to transform amateur video footage or image sequence into quality hyperlapse videos without the need of expensive equipment or softwares like Adobe Premiere Pro. This would make hyperlapse video creation easier and more feasible for more people, allowing it to thrive as an photography tool and an art form. Content creators, film makers, or just people interested in making travel vlogs now have a more accessible option to create hyperlapse videos.

References

[1] J. B. M. L. Alexandre Karpenko, David Jacobs. Digital video stabilization and rolling shutter correction using gyroscopes. 2013. 1

[2] R. Cabral, F. D. L. Torre, J. P. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *2013 IEEE International Conference on Computer Vision*, pages 2488–2495, Dec 2013. 2

[3] R. Chen and C. Gotsman. Generalized As-Similar-As-Possible Warping with Applications in Digital Photography. *Computer Graphics Forum*, 35(2):081–092, 2016. 2, 3

[4] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Optimizing color consistency in photo collections. *ACM Trans. Graph.*, 32(4):38:1–38:10, July 2013. 1

[5] S. N. S. Jaesik Park, Yu-Wing Tai and I. S. Kweon. Efficient and robust color consistency for community photo collections. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 5

[6] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen. Real-time hyperlapse creation via optimal frame selection. *ACM Trans. Graph.*, 34:63:1–63:9, 2015. 1, 2, 5

[7] S. Liu, L. Yuan, P. Tan, and J. Sun. Bundled camera paths for video stabilization. *ACM Trans. Graph.*, 32:78:1–78:10, 2013. 2, 4, 5

[8] J. Park. Color consistency for community photo collections. `https://github.com/syncle/photo_consistency`, 2018. 2

[9] T. Uno. Mace: Maximal clique enumerator. 2, 4

Appendix

Team contributions

Please describe in one paragraph per team member what each of you contributed to the project.

Jiaju Ma Researched and implemented second stage of the pipeline (**Color Consistency Across Frames**) based on the method proposed in [5]. Contributed to our own database used to create hyperlapse videos. Worked on the final presentation slides and the final report.

Michael Mao Researched and implemented first stage of the pipeline(**Video Stabilization**) based on the method proposed in [7]. Managed project environment and maintained automatic docs generation for the project. Contributed to our database and worked on the final presentation slides and the final report.

James Li Researched and implemented first stage of the pipeline(**Optimal Frame Selection**) based on method proposed in [6]. Contributed to our database and worked on the final presentation slides and the final report.

Figures

Figure 2. Salomon-Main Green Photo Series *Upper:* Selected Input Images. *Lower:* Corresponding Outputs After Color Adjustments.

Figure 3. Extracted Frames from the Main Green Video. *Upper:* Selected Input Images. *Lower:* Corresponding Outputs After Color Adjustments.

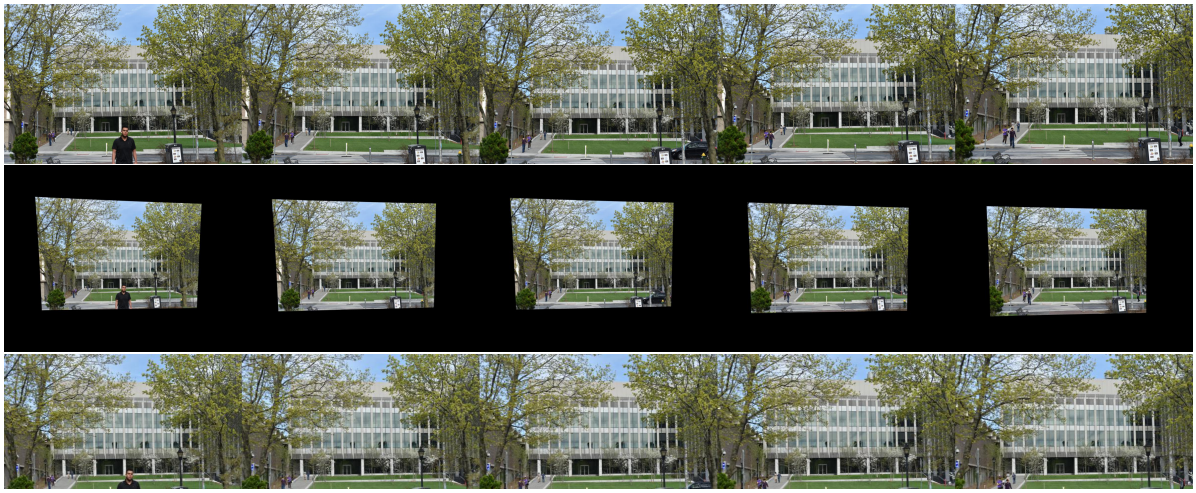

Figure 4. Selected frames from the Engineering Building photo series. *Upper:* Input Images (Camera JPEG). *Middle:* Corresponding Outputs After Path Smoothed Warping. *Lower:* Corresponding Outputs After Cropping.

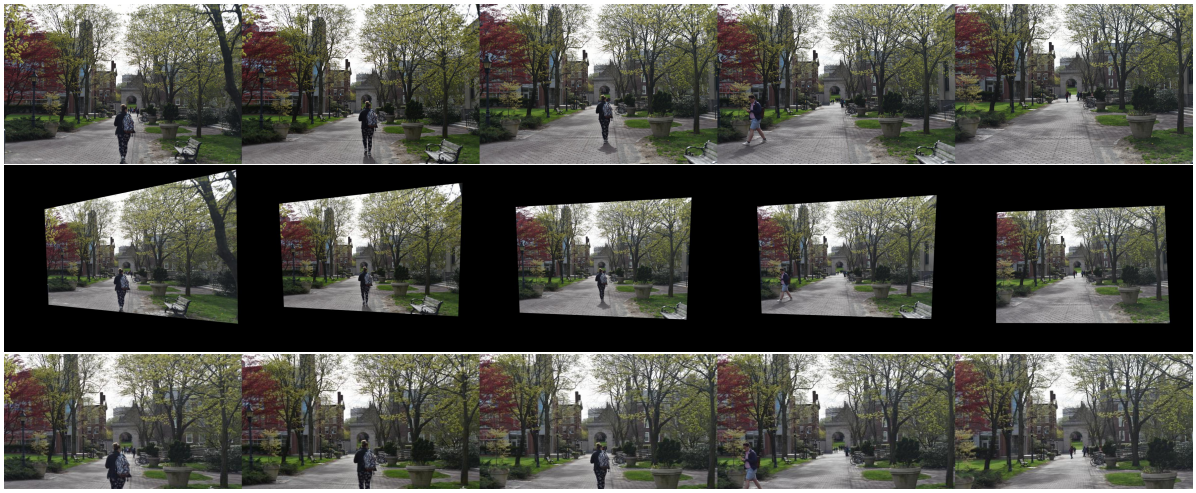

Figure 5. Selected frames from the Metcalf-Caswell Arch photo series. *Upper:* Input Images (Camera JPEG). *Middle:* Corresponding Outputs After Path Smoothed Warping. *Lower:* Corresponding Outputs After Cropping.

Figure 6. Extracted frames from the Main Green photo video. *Upper:* Extracted Images (After Color Correction). *Middle:* Corresponding Outputs After Path Smoothed Warping. *Lower:* Corresponding Outputs After Cropping.