Robust Video Stabilization by Optimization in CNN Weight Space

Jiyang Yu
University of California, San Diego
jiy173@eng.ucsd.edu

Ravi Ramamoorthi
University of California, San Diego
ravir@cs.ucsd.edu

Abstract

We propose a novel robust video stabilization method. Unlike traditional video stabilization techniques that involve complex motion models, we directly model the appearance change of the frames as the dense optical flow field of consecutive frames. We introduce a new formulation of the video stabilization task based on first principles, which leads to a large scale non-convex problem. This problem is hard to solve, so previous optical flow based approaches have resorted to heuristics. In this paper, we propose a novel optimization routine that transfers this problem into the convolutional neural network parameter domain. While we exploit the general benefits of CNNs, including standard gradient-based optimization techniques, our method is a new approach to using CNNs purely as an optimizer rather than learning from data. Our method trains the CNN from scratch on each specific input example, and intentionally overfits the CNN parameters to produce the best result on the input example. By solving the problem in the CNN weight space rather than directly for image pixels, we make it a viable formulation for video stabilization. Our method produces both visually and quantitatively better results than previous work, and is robust in situations acknowledged as limitations in current state-of-the-art methods.

1. Introduction

Video stabilization has been an important part of both professional and amateur video processing tools like Adobe Premiere, After Effects and Deshaker. There are also numerous researches on video stabilization. These works are typically summarized into two categories according to the dimensionality of the model used to interpret the frame motion. The 2D methods analyze the video frame motion by tracking features in 2D image space. The frame motion is modeled as a full-frame 2D image transformation like Matsushita et al.[22] or a grid of local homographies like Liu et al.[21]. 3D methods seek to explore the 3D location of feature points in the scene while calculating the camera pose in 3D space. Works following this direction include Liu et al.[18], Bhat et al.[1] and Sun[25]. However, all previous methods involve various heuristics. In real scenarios, complex effects (e.g. motion blur, occlusion, parallax) may unexpectedly break the assumptions in these methods and therefore produce artifacts in the results.

It would be ideal if we could make no assumptions about the physics and directly optimize the appearance change of the frames in the results so that the video is stabilized. To model the appearance change, dense optical flow fields between consecutive frames are needed. We seek to apply pixel-wise offsets to these optical flow fields, and smooth the motion of each pixel. However, modeling the frames with optical flow brings three major challenges. First, modern videos are usually high-definition. This means that the number of pixels in each frame is large. As the length of video grows, optimization quickly becomes intractable since too many pixel motions need to be solved. (For example, for a 100-frame standard 480P video(854×480), the number of motion vectors to be solved is $854 * 480 * 100 = 4.1 \times 10^7$). Second, as the problem size becomes large, the energy landscape of the non-convex optimization becomes complex. General gradient-based optimization algorithms may easily get stuck in local minima and yield unsatisfactory results. Third, the performance of the video stabilization is affected by the quality of optical flow. Local errors in the optical flow map will also be blindly treated as actual pixel motions, causing artifacts in the final results. Therefore, the regularization needs to be carefully designed to enforce spatial consistency and maintain robustness to the errors in the optical flow field.

Instead of trying to directly solve for the pixel-wise warp field, we propose a novel method that optimizes in the space of neural network parameters. Note that unlike standard CNN approaches, we don't use large datasets or learning of parameters a-priori. We train the CNN from scratch on a single input video. In fact, there is no traditional training in our method; the CNN is simply used as a robust way to do global optimization with a physically-based objective function. The other important difference from traditional CNN training is that we seek to overfit the data as much as possible since our sole goal is to produce the best results for the single input video. By optimizing the parameters of the CNN rather than directly for pixels in a single test video, we make the dense warp field optimization tractable. A similar idea has been employed in image generation and restoration by Ulyanov et al.[27], but is applied by us for the first time in video stabilization. This idea may be applicable in many other image and video processing applications where the physically-based problem is intractable for traditional optimization algorithms.

The pipeline of our method is shown in Fig. 1. We first pre-stabilize the video to reduce the frame motion (Appendix A). We use optical flow between consecutive frames to generate

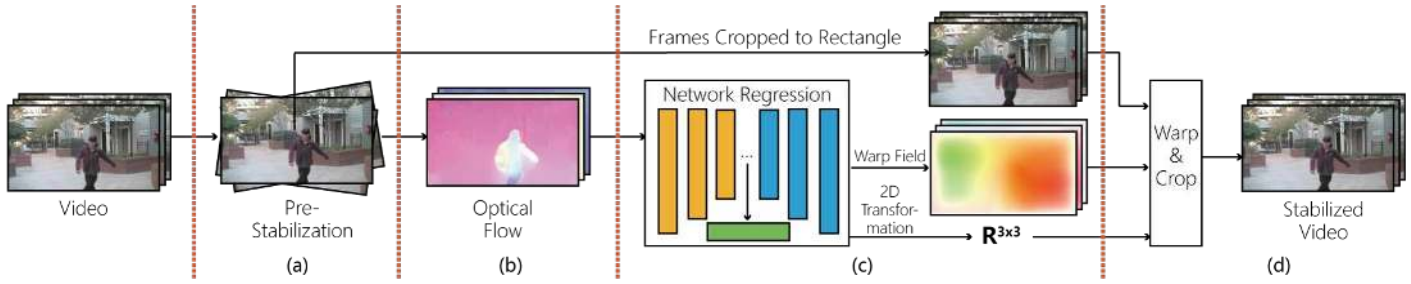

Figure 1. *The pipeline of our method. (a) We first pre-stabilize the video using basic 2D affine transformation. (b) Using the optical flow between consecutive frames of the pre-stabilized video, we formulate the video stabilization as the minimization of the distance between corresponding pixels. (c) We use a convolutional neural network as the optimizer to solve for a 2D affine transformation and a warp field for each frame. (d) The video frames are warped and cropped to a rectangle to produce the stabilized result.*

dense correspondence of all pixels between the two frames. The stabilization is achieved by minimizing the distances between corresponding pixels. We seek to solve a full frame 2D image transformation and a dense warp field for each frame, so that the original frames can be warped and stabilized. We will discuss our formulation of the video stabilization problem in Sec. 3.

In summary, our contributions include:

Optical flow based formulation: We use the optical flow to track the actual motion of all the pixels in the video instead of pixel profiles in Liu et al.[20], which enables high robustness, universal stabilization over any part of the scene (regardless of foreground or background), and flexible non-parametric frame warping (Sec. 3). Our novel formulation of video stabilization leads to a large scale non-convex problem, which is addressed as discussed below with a CNN-based optimization.

Neural network based regression: We propose a new idea to transfer the video stabilization problem into a neural network based regression (Sec. 4). We also discuss the implementation details of the neural network regression in Sec. 5. We analyze the effect of different network structures on the final results, and propose a network structure that is best for video stabilization. Our network structure significantly simplifies the optimization process and generates compelling results (Sec. 6).

2. Related Works

In this section, we summarize related works in video stabilization and neural network based regression.

2.1. Video Stabilization

2D Methods The 2D methods in general have low computational complexity and can be solved efficiently. However, 2D methods suffer from potential problems. First, the tracked 2D features can be unreliable due to motion blur and illumination change. Obtaining long feature tracks is also difficult in videos with significant occlusions. Liu et al.[21] tracks the 2D feature points and solves for a grid that smoothes its enclosing feature tracks. Grundmann et al.[7] requires a camera path calculated from feature tracks as an initialization of the algorithm. Buehler et al.[2] also requires long feature tracks and simple motion scenarios. Some methods seek to explore the relative position of feature points. Liu et al.[19] perform stabilization on the extracted eigen-trajectories. Goldstein and Fattal[6] utilize the epipolar geometry to maintain the relative position of feature points. Wang et al.[29] also seek to keep the relative position of feature points. These works' performances are still subject to the quality of tracked features.

Second, using a parametric motion model is usually insufficient to stabilize videos with parallax effects, since the motions of pixels in the same frame are not subject to the same homography constraint. Matsushita et al.[22] and Gleicher and Liu[5] treat the scene as a plane and use a full-frame homography to stabilize the video. Liu et al.[21] and Yu and Ramamoorthi[31] divide the frames into grids and apply a local homography, but essentially cannot handle complex depth variation in the scene. Liu et al.[20] uses optical flow to warp the original frames. However, the pixel profile proposed in Liu et al.[20] is very sensitive to motion discontinuities. Therefore, they still need heuristics to identify the foreground/background and carefully inpaint the regions where the motion is different from the background.

Our method is more robust than general 2D methods in terms of feature tracking, since our method tracks all the pixels and is robust to local errors in the optical flow. We also enable a non-parametric frame warping, which handles parallax effects without reconstructing the 3D structure of the scene. Although we used optical flow to synthesize stabilized frames like Liu et al.[20], our method is fundamentally different from their work. In our work, we track the actual motion of each pixel instead of the pixel profile which only collects the motion vectors at each pixel position. This makes our method robust to parallax and does not require the filling in of the motion discontinuity regions. However, our formulation results in a large scale non-convex problem which cannot be written as a simple quadratic form as in Liu et al.[20] To solve this non-trivial problem, we discuss our novel neural network based optimization routine in Sec. 4.

3D Methods Unlike 2D methods, 3D methods seek to explore the 3D location of feature points in the scene while calculating the camera pose in 3D space. These works in general handle parallax better than 2D methods, since the motion is physically analyzed in actual 3D space. In these works, the camera path is smoothed and the 3D feature points are reprojected to new camera positions in order to guide the warping[18, 25] or the methods use image-based rendering[1] to synthesize a frame from the original frames. However, the 3D methods suffer from robustness and complexity issues in Structure from Motion.

There are also video stabilization methods that require spe-

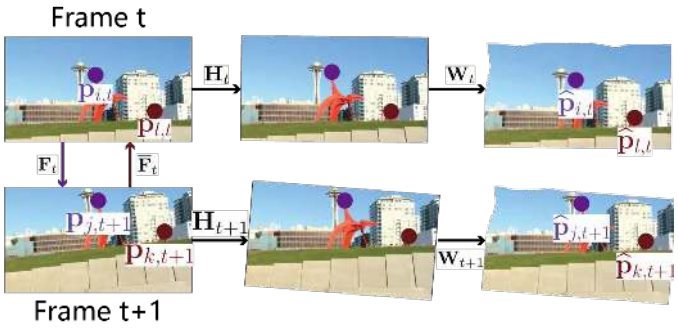

Figure 2. *The objective function of our method. The computed bidirectional original optical flow provides correspondence among all pixels in two consecutive frames. In this figure,* $\mathbf{p}_{i,t}$ *is associated with* $\mathbf{p}_{j,t+1}$ *by* $\mathbf{F}_t$ *and* $\mathbf{p}_{k,t+1}$ *is associated with* $\mathbf{p}_{l,t}$ *by* $\overline{\mathbf{F}}_t$*. Our goal is to solve a 2D affine transformation* $\mathbf{H}$ *and a warp field* $\mathbf{W}$ *to minimize the distance between the associated pixels.*

cific hardware information or focus on video captured with a specific camera. Sun[25] requires a depth camera for video stabilization. Smith et al.[24] requires a light field camera. Karpenko et al.[10] uses gyroscope information to help in stabilizing the video. Kopf[12] focuses on videos captured with a 360° camera. Some of these works show strong results, but have limited application since most videos do not include the extra information required in these algorithms.

Deep Learning Based Methods Some recent works seek to use a pre-trained network in video stabilization tasks. Wang et.al.[28] train a two-branch Siamese network and try to directly predict the homography transformation from the current frame and previous stabilized frames. Xu et.al.[30] also try to use spatial transformer networks (STN) to predict the affine transformation. Moreover, they use an adversarial network to directly generate previous stabilized frames instead of the real frames in Wang et.al.[28]. Although these works utilize pre-trained CNNs, they still use a simple full-frame transformation as the motion model, which cannot handle parallax effects. Lin et.al.[16] proposed a mesh deformation algorithm by enforcing the photo consistency between images. Their semi-dense photometric alignment provides better robustness compared to methods using feature points. However, the photometric based metric can be unreliable in homogeneous regions, object boundary and occluded regions. These regions are also challenging for optical flow algorithms, but as we will discuss in Sec. 3, our regularization can help avoid visual artifacts in these regions.

2.2. Neural Network Regression

Deep convolutional neural networks have been used in various image/video processing tasks. Such applications include image super-resolution[3, 13, 14, 26], image denoising[15], HDR reconstruction[4], panorama video loop generation[8] and video interpolation[9]. Some of these methods are designed for processing a single image/video, but their networks are essentially trained on a large dataset of images/videos. Unlike traditional deep learning, our method treats the network purely as an optimizer over one single input video. We train the network from scratch for each specific video, and try to overfit and obtain the best result for the input.

Notation	Meaning	Size
w	Frame width	1
h	Frame height	1
t	Frame index(time)	1
T	Total number of frames	1
$\mathbf{I}_t$	Input RGB frame at t	$w \times h \times 3$
x, y	Pixel coordinate	1
i, j, k, l	Pixel ID	1
$\mathbf{p}_{i,t}$	Spatial location of pixel i at t	2×1
$\mathbf{p}^h_{i,t}$	Homogeneous version of $\mathbf{p}_{i,t}$	3×1
$\widehat{\mathbf{p}}_{i,t}$	Warped $\mathbf{p}_{i,t}$	2×1
$\mathbf{P}_t$	Coordinates of all pixel in frame I_t	$wh \times 2$
$\mathbf{S}_t$	The coordinates of four corner pixels	4×2
$\mathbf{D}$	Weight of all pixels in the 2D interpolation representation w.r.t. $\mathbf{S}$	$wh \times 4$
$\mathbf{F}_t$	Optical flow from frame I_t to I_{t+1}	$w \times h \times 2$
$\overline{\mathbf{F}}_t$	Optical flow from frame I_{t+1} to I_t	$w \times h \times 2$
$\mathbf{H}_t$	2D affine transformation	2×3
$\mathbf{W}_t$	2D warp field	$w \times h \times 2$
θ	Neural network parameters	
$\mathbf{G}(\theta)$	Neural network as a function of θ	

Table 1. Notations used throughout the paper.

Ulyanov et al.[27] recently proposed that a randomly-initialized neural network can be used in learning image priors on a single image. The idea is that the variables of a typical optimization task can be replaced by the output of a neural network. The neural network is randomly initialized and trained on a single image to minimize the loss function designed for a specific task, e.g. denoising, superresolution, and inpainting. This enables the optimization in the neural network parameter space instead of image space, while dramatically improving the result. In this work, we expand this idea into video processing, which is more difficult than the single image processing scenario in terms of the problem formulation and complexity. By transferring the video stabilization problem into neural network parameter space, we can easily solve the large scale problem, which is difficult to solve using traditional optimization methods. Moreover, we further analyze the effects of different types of architectures on the final optimization result. We will discuss this idea in detail in Sec. 4.

3. Optical Flow Based Objective Function

It is well-known that videos usually have multiple factors that cause difficulties for video stabilization algorithms, e.g. lens distortion, motion blur, dynamic objects, parallax, low-illumination etc. These effects can be individually modeled using hand-crafted physically based models. However, in real-world videos, effects usually couple with each other, making algorithms specifically designed for one single effect fail in other cases. Instead of trying to physically model these complex effects, we treat the video stabilization task as a pure 2D image processing problem. In this paper, we seek to minimize the appearance change among video frames.

3.1. Optical Flow Objective Function

To model the appearance change, an intuitive approach is to calculate original optical flow between consecutive video frames and find frame transformations to minimize the motion

Figure 3. *Video stabilization results using regularization vs. no regularization. (Left) Due to the moving object in the scene and the inaccurate original optical flow, unexpected artifacts are introduced if the video is stabilized purely according to optical flow. (Right) Applying proper regularization helps reduce the visual artifacts.*

of pixels. Note that in Fig. 1, we first perform a pre-stabilization step, where we track sparse feature points and preliminarily stabilize the video as discussed in Appendix A. The remainder of the paper takes the output from this pre-stabilization step as its input, and discusses subsequent optimization to stabilize the resulting video. To make the following discussion clear, we define the notations in Table 1.

To simplify the notation, we unroll the pixel coordinate x, y to a single pixel ID i. Denote the original optical flow from t to $t+1$ as $\mathbf{F}_t$, a two-channel image that encodes the shift of all the pixels in frame $\mathbf{I}_t$ to frame $\mathbf{I}_{t+1}$. For example, denote the position of pixel i in frame $\mathbf{I}_t$ as $\mathbf{p}_{i,t}$. Its corresponding pixel in frame $\mathbf{I}_{t+1}$ can be represented as

$$\mathbf{p}_{j,t+1} = \mathbf{p}_{i,t} + \mathbf{F}_t(\mathbf{p}_{i,t}) \tag{1}$$

Similarly, a backward optical flow can be computed and maps the pixels in frame $\mathbf{I}_{t+1}$ to frame $\mathbf{I}_t$

$$\mathbf{p}_{l,t} = \mathbf{p}_{k,t+1} + \overline{\mathbf{F}}_t(\mathbf{p}_{k,t+1}) \tag{2}$$

We illustrate our approach in Fig. 2. Our goal is to warp each original frame so that the output frames are stabilized. The warping operation consists of two components: a 2D affine transformation $\mathbf{H}_t$ and a per-pixel warp field $\mathbf{W}_t$. Therefore, the warped pixel i and l in frame $\mathbf{I}_t$ can be represented as

$$\begin{aligned} \widehat{\mathbf{p}}_{i,t} &= \mathbf{H}_t \mathbf{p}^h_{i,t} + \mathbf{W}_t(\mathbf{p}_{i,t}) \\ \widehat{\mathbf{p}}_{l,t} &= \mathbf{H}_t \mathbf{p}^h_{l,t} + \mathbf{W}_t(\mathbf{p}_{l,t}) \end{aligned} \tag{3}$$

where $\mathbf{p}^h_{i,t}$ and $\mathbf{p}^h_{l,t}$ stands for the homogeneous representation of $\mathbf{p}_{i,t}$ and $\mathbf{p}_{l,t}$. Similiarly, its warped correspondence in frame I_{t+1} is

$$\begin{aligned} \widehat{\mathbf{p}}_{j,t+1} &= \mathbf{H}_{t+1} \mathbf{p}^h_{j,t+1} + \mathbf{W}_{t+1}(\mathbf{p}_{j,t+1}) \\ \widehat{\mathbf{p}}_{k,t+1} &= \mathbf{H}_{t+1} \mathbf{p}^h_{k,t+1} + \mathbf{W}_{t+1}(\mathbf{p}_{k,t+1}) \end{aligned} \tag{4}$$

The objective is to minimize the Euclidean distance between the warped pixel positions:

$$\begin{aligned} E_o(\mathbf{W}, \mathbf{H}) = \frac{1}{wh(T-1)} \sum_{t=1}^{T-1} \\ (\sum_{i=1}^{wh} \|\widehat{\mathbf{p}}_{i,t} - \widehat{\mathbf{p}}_{j,t+1}\|^2 + \sum_{k=1}^{wh} \|\widehat{\mathbf{p}}_{l,t} - \widehat{\mathbf{p}}_{k,t+1}\|^2) \end{aligned} \tag{5}$$

where wh is the total number of pixels in a frame and T is the total number of frames. Note that the mapping from $\mathbf{p}_{i/l,t}$ to $\mathbf{p}_{j/k,t+1}$ is 1-to-1, so we only need to average over i and k.

3.2. Regularization

Due to the complexity of scenes, the original optical flow could be inaccurate in some regions. Moreover, objects in the scene might be moving regardless of the motion of the camera. Blindly optimizing the objective function (5) could introduce artifacts. An example of these artifacts is shown in Fig. 3. Therefore, we seek to enforce the local continuity of the output warp field $\mathbf{W}_t$.

The four corner pixels define a rectangular region, in which each pixel position $\mathbf{p}_{i,t}$ can be represented by a linear interpolation of the coordinates of the four corner pixels: $\mathbf{P}_t = \mathbf{D}\mathbf{S}_t$, where each row of $\mathbf{P}_t$ is the coordinate of pixels, each row of $\mathbf{D}$ is the 2D interpolation weight, and each row of $\mathbf{S}_t$ is the 2D coordinates of the four corners. Note that moving the corner position correspondingly changes all the pixel locations:

$$\Delta\mathbf{P}_t = \mathbf{D}\Delta\mathbf{S}_t$$

A warp field obeying this linear warping rule should satisfy:

$$\|\mathbf{W}_t - \mathbf{D}\Delta\mathbf{S}_t\|_2 = 0 \tag{6}$$

However, our output warp field $\mathbf{W}_t$ will not exactly be a linear warping. Our goal is to keep the term in (6) as small as possible. The least squares representation of $\Delta\mathbf{S}_t$ is:

$$\Delta\mathbf{S}_t = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{W}_t$$

This estimation of $\Delta\mathbf{S}_t$ leads to the error of:

$$E_r(\mathbf{W}) = \mathbf{W}_t - \mathbf{D}(\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{W}_t \tag{7}$$

We use this error as a constraint to enforce the output warp field close to a linear warp field. Note that this formulation allows us to control the linear warping constraint at a pixel-level, simply by changing $\mathbf{D}$. For example, in our experiment, we cover each frame with a 20x20 grid and fill $\mathbf{D}$ with the weight of each pixel in its enclosing grid cell.

Moreover, the original optical flow $\mathbf{F}_t$ is less reliable for regions with large motions. To take this into consideration, we tend to increase the regularization value (7) for large motion regions to obtain $\mathbf{W}_t$ with fewer discontinuities; on the other hand, for small motion regions, we tend to trust the optical flow and decrease the regularization value. The measurement of motion scale can be estimated using the pixel motion obtained from the original optical flow:

$$E_p = \mathbf{F}_t^2 + \overline{\mathbf{F}}_t^2 \tag{8}$$

3.3. Final Objective Function

Combining (5), (7) and (8), our optimization problem can be written as:

$$\min_{\mathbf{W},\mathbf{H}} E_o(\mathbf{W}, \mathbf{H}) + \lambda \left\| E_p \cdot E_r(\mathbf{W}) \right\|_1 \tag{9}$$

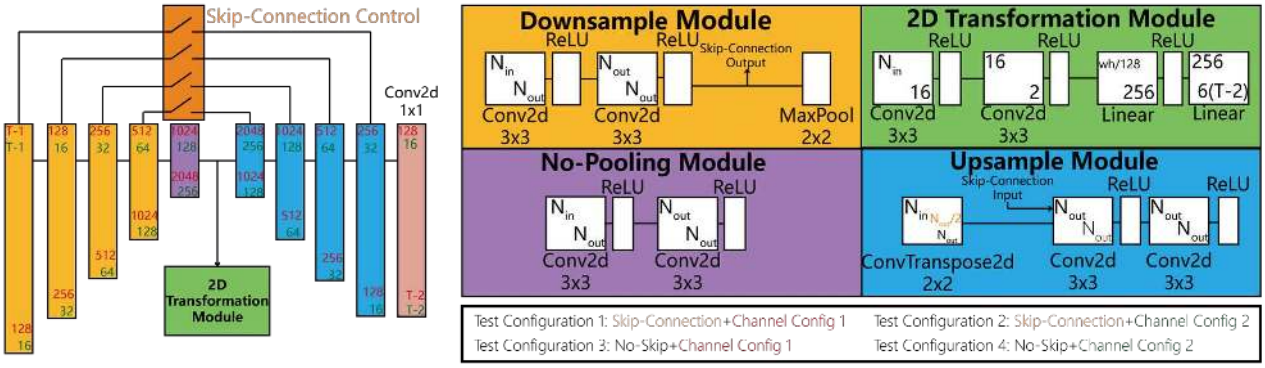

Figure 4. *Our network structure. The overall structure is shown on the left. The details of each type of module are shown on the right. The numbers shown on the upper-left/lower-right corner of each module represent the number of input/output channels. The numbers in red/green represent two different channel configurations of the network. The orange box represents optional skip-connections between downsample modules and upsample modules. Combining the channel configurations and skip-connection options, a total of 4 different configurations are used in our experiment (listed on the lower-right of the figure).*

where λ is a hyperparameter controlling the amount of regularization in general. Since the magnitude of original optical flow E_p has the same size as the regularization E_r, we use it as a pixel-wise weight to E_r. Note that to encourage sparsity in the warp field and avoid over compensation to erroneous regions in the original optical flow, we use L_1 norm for this regularization.

Discussion Our formulation directly models the motion of every pixel in a video using dense optical flow. Unlike previous works that use various heuristics, our method is based on the first principles that we should stabilize what we finally perceive. The most similiar idea is the SteadyFlow proposed by Liu et al.[20]. However, they only collect the motion vector on fixed pixels. The motion vectors on a single pixel correspond to the motion of different locations in the scene. The accumulation of these vectors does not match the true motion of the camera. Our formulation is novel since we physically model the motion of every visible point in the scene. This leads to a more difficult optimization problem, as we will discuss in Sec. 4.

4. Convolutional Neural Network Regression

Note that the unknowns in (9) are a per-frame optical flow field $\mathbf{W}_t$ and a per-frame 2D affine transformation $\mathbf{H}_t$. For a 300-frame video clip with a standard 480p resolution, the total number of unknown motion vectors is approximately 123 million. Directly optimizing a problem of this size is typically prohibitive due to the computation cost and limited memory. Moreover, the optimization will be difficult due to the complex high-dimensional energy landscape with a large number of local minima.

Our main idea to solve this problem is to search for the answer in the neural network parameter space instead of in the problem space. In fact, we are using the neural network as an optimizer. Our method is different from traditional learning on large datasets. There is no training set, and the network weights are directly optimized on the input video with the objective function in (9). Using a network makes this non-convex high-dimensional optimization problem practical, enabling us to directly use a robust optical-flow based stabilization formulation.

To our knowledge, our method is the first work that uses this idea in video stabilization tasks. Another insight is that although the optical flow field is represented pixel-wise, it is spatially smooth for real-world scenes. Therefore, our warp field $\{\mathbf{W}, \mathbf{H}\}$ can be described well by a parameterized function. However, given the complexity of this process, a complex and differentiable parameterized function needs to be designed. Instead of hand-crafting this function, we select the convolutional neural network as an ideal out-of-the-box solution for this task.

Denote the neural network as a function $\mathbf{G}(\theta)$ where θ represents the parameters of the network. We seek a set of network weights so that the output of the network is the desired warp field $\{\mathbf{W}, \mathbf{H}\} = \mathbf{G}(\theta)$. Therefore, the optimization problem (9) can be reformulated as:

$$\min_{\theta} E_o(\mathbf{G}(\theta)) + \lambda \left\| E_p \cdot E_r(\mathbf{G}(\theta)) \right\|_1 \tag{10}$$

The goal becomes searching for the parameters θ by training the network on a single video clip. Note that since our objective function consists of simple linear and quadratic functions of $\{\mathbf{W}, \mathbf{H}\}$, (10) is differentiable with respect to network parameters θ.

Our network structure is shown in Fig. 4. The input of the network is a set of $T-1$ original optical flow fields $\mathbf{F}$ computed from input video frames. The frames of optical flow fields are sent in as different channels. The input is encoded by 5 layers of downsample modules. Each downsample module downsamples the frame size by 2 but doubles the number of channels, except the last one that only doubles the number of channels. The decoder consists of 4 layers of upsample modules followed by an output convolutional layer with kernel size 1×1. The output of the decoder is the desired warp field $\mathbf{W}$. In addition, we fed the encoded information into a 2D transformation module consisting of two convolutional layers and two linear layers. This module produces the desired 2D affine transformation matrix $\mathbf{H}$. The number of output $\{\mathbf{W}, \mathbf{H}\}$ pairs is $T-2$. We will explain why we have $T-1$ input channels and $T-2$ output channels, and discuss the selection of T in Sec. 5. Since we only input the optical flow of a single video and try to optimize the network parameters, we seek to overfit the single input video as

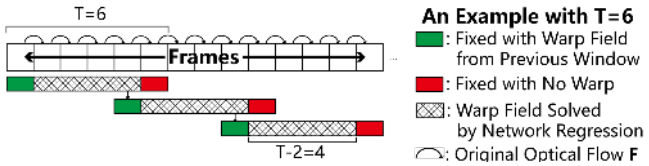

Figure 5. *The sliding window example with window size $T = 6$. For each window, the warp field of the first frame is fixed as computed from the previous window. The last frame is fixed so that it is not warped. We optimize the objective function discussed in Sec. 3 for the entire length T window and solve for $T - 2$ warp fields.*

much as possible. Therefore we avoid inserting dropout layers and any regularization on network weights.

As noted in Fig. 4, we have two different channel configurations. The channel configuration 1 requires more network parameters, while the channel configuration 2 leads to a simpler network. In addition, the network can optionally include skip connections. Combining these choices, we have four different network configurations in this paper. We will compare the performance of these configurations in Sec. 6. Specifically, we will show the relation between the network configuration and the regression error (9) in Fig. 11 and discuss how the network configuration will affect the optimization performance. We will also show the network configuration's effect on the final stabilization result in Fig. 11.

5. Implementation Details

Sliding Window We now explain the details about why we have $T - 1$ optical flow fields as the network input and $T - 2$ warp fields as the network output. Since the input video may have different lengths, we stabilize a video using a sliding window approach. The process is demonstrated in Fig. 5, where an example with $T = 6$ is shown. In this case, the window covers $T = 6$ video frames and $T-1 = 5$ original optical flow frames. Note that since the input of our network is the original optical flow, the number of input channels is $T-1$. The desired number of estimated $\{\mathbf{W}, \mathbf{H}\}$ pairs should be 6, which can warp each frame and generate stabilized frames. However, to enforce temporal consistency, we make the windows overlap by two frames and fix the warp field of the first frame. We also fix the last frame to retain the global motion of the original video. The warp field of the first frame is copied from the estimation of the previous window. The warp field of the last frame is fixed to $\mathbf{W} = \mathbf{0}, \mathbf{H} = \mathbf{I}$ for the current window, but will be re-optimized as the second frame of the next window. The last frame of the last window remains unwarped. Therefore, for each window, we have $T - 2 = 4$ pairs of warp fields $\{\mathbf{W}, \mathbf{H}\}$ as the network output.

Selection of Window Size It is clear that the selection of T will affect the complexity of the optimization problem. The more frames we want to stabilize at the same time, the more complex the energy landscape will be. In Fig. 6, we show the error descent of optimization using T from 10 to 80 frames with a step size of 10. The y axis represents the percentage of the error of current iteration with respect to the initial error. Each curve is the averaged result for all segments of our examples in Sec. 6. It shows that in smaller T cases, the optimization converges faster but yields higher error after convergence. This is because in a shorter video segment, we have fewer degrees of freedom in the warp field. Although a larger window size leads to better error performance, more memory and iterations are required to stabilize a video segment. Taking all these into consideration, we select $T = 60$ in our experiments.

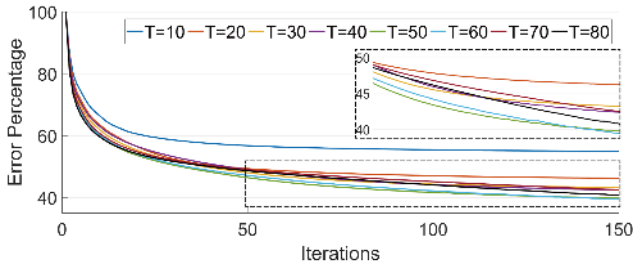

Figure 6. *Comparison of results stabilized with window size $T = 10$ to $T = 80$. The inset shows a zoom-in of the region circled by the dotted box. Smaller window cases converge faster but result in higher error. Larger window cases yield lower error but converge slower.*

Miscellaneous We use the Liu[17] to compute the original bidirectional optical flow. Before computing the original optical flow, we pre-stabilize the video to eliminate large motions. The reason for the pre-stabilization stage includes two aspects: the quality of optical flow is undermined by large motions; the pixels in the boundary regions do not have correspondence in their neighboring frame, and this effect is significant in large motion cases. We will discuss details about pre-stabilization in Appendix A. The regularization value λ in (9) is set to be 0.5 for all the examples shown in the supplementary video. The optimizer we used is Adam[11] with $\beta_1 = 0.5$, $\beta_2 = 0.59$ and a learning rate of 10^{-4}. We optimize for 150 iterations for each $T = 60$ window.

Figure 7. *Example stills of our examples. The example numbers are labeled above the frames. In the right table, we also summarize their properties that have significant effects on video stabilization algorithms.*

6. Results

In Fig. 7, we show example frames of the video clips used in our paper. In order to collect a large enough set of examples for comparison, we combined datasets from many previous papers. In our dataset, numbers 1-8 are taken from Steadyflow[20], numbers 9-15 are taken from Liu et al.[21], numbers 16-20 are taken from Liu et al.[18], and numbers 21-25 are taken from Yu and Ramamoorthi[31]. We also summarize their properties that have significant effects on video stabilization algorithms in the right table in Fig. 7.

We use five metrics to evaluate the quality of the results. Our result is generated with Config 1 mentioned in Fig. 4. We will

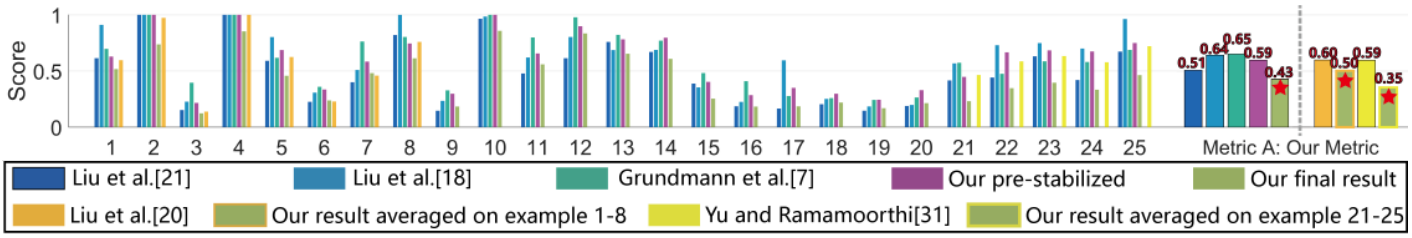

Figure 8. *Quantitative comparison using our metric. In this figure, a lower bar indicates a better result. The result is normalized with the score of the input video. In the left figure, results that have a normalized score greater than one are clamped in this figure. The right figure shows the averaged results over the entire 25 examples. On the rightmost part of the figure, we only compare to Liu et al.[20] using their data (examples 1-8) since we don't have their implementation. We compare to Yu and Ramamoorthi[31] only on selfie videos (examples 21-25). We mark our results with the red stars. The exact scores are marked on top of each bar.*

further discuss the effect of network configuration later in this section.

Quantitative Results Using our Objective Function: In Fig. 8, we show quantitative comparison of the result quality of the input video, our result and Steadyflow[20], Grundmann et al.[7], Liu et al.[18], Liu et al.[21] and Yu and Ramamoorthi[31]. Metric A is our metric, which is defined as the accumulated optical flow over the entire video:

$$\frac{1}{wh(T-1)} \sum_{t=1}^{T-1} \sum_{x=1}^{h} \sum_{y=1}^{w} (\|\mathbf{F}_t(x,y)\|_2 + \|\overline{\mathbf{F}}_t(x,y)\|_2)$$

which evaluates the appearance change between consecutive frames in the results. The essence of this metric is similar to our objective function (5). However, the metric is different from (5): the optical flow it uses is computed from the resulting video. Note that to compare videos with different frame sizes, we normalize the optical flow by its frame size. A smaller score indicates a better result in metric A. To show the amount of stability improvement, all the scores are normalized by the score of the input video.

In our work, we directly tried to minimize the overall appearance change. Therefore, our method achieves the best result on average and performs better than comparison methods under this metric. Note the benefit gained by using our CNN based optimization framework, comparing to the pre-stabilized result. We also perform significantly better in example 8, which contains large foreground occlusion and is claimed as a limitation case in SteadyFlow[20]. For selfie videos (example 21-25) in which large occlusion exists, we are also able to achieve smoother appearance change. Our method obtains a larger score comparing to Liu et al.[21] in example 12, but their result contains large visual distortion as we will discuss later in this section. For some examples (6, 7, 9, 11, 17, 20), our result has a slightly higher score than the comparison methods, but there are no visible quality differences with the other methods.

In Fig. 10, we also compare our method with the other video stabilization methods over the commonly used NUS dataset[21]. We randomly select 5 videos from each category and average the Metric A of the results. Our method performs better on this more general video stabilization dataset.

Quantitative Results Using Other Metrics: The metrics B, C and D were proposed by Liu et al.[21], which evaluates the results' cropping ratio, global distortion and frequency domain

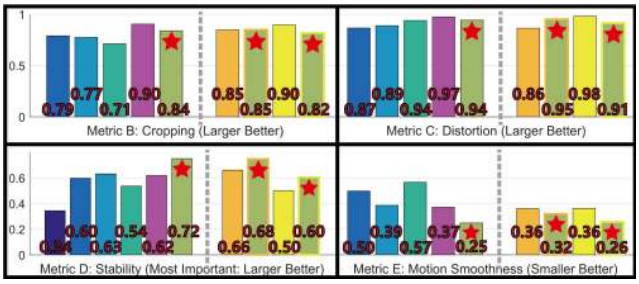

Figure 9. *Quantitative comparison using metrics proposed by Liu et al.[21] and Yu and Ramamoorthi[31]. Metric B measures the cropping ratio compared to the input video, metric C measures the global distortion, metric D measures the frequency domain stability and metric E measures the motion smoothness. We mark our results with the red stars. The exact values are marked at the bottom of each bar. Metric D is considered as the most important metric.*

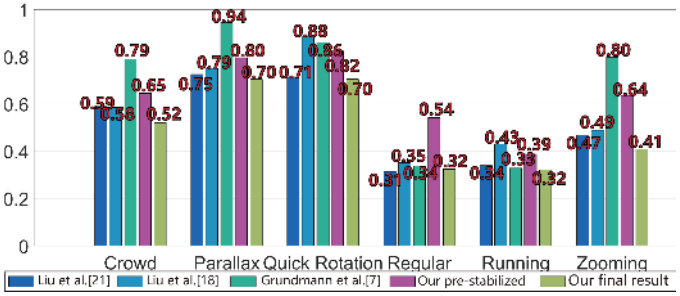

Figure 10. *Metric A evaluation over the results for the NUS dataset. We randomly select 5 videos from each category and average the results. Note that our network optimization significantly improves the pre-stabilization results. Our method is also better than comparison methods in challenging categories like Crowd, Parallax and Running.*

stability. In metrics B, C and D, a higher value indicates a better result. Metric E was proposed by Yu and Ramamoorthi[31], which evaluates the smoothness of the frame motion in the result. In metric E, a smaller value indicates a better result. In Fig. 9, we show comparison of averaged score over the entire 25 examples. The full comparison on each individual video is provided in the supplementary material.

In metrics B and C, since we warp the pre-stabilized video using the warp field and crop to a rectangle, we expect the final result to be slightly worse than the pre-stabilized result in terms of cropping and distortion. However, we are still achieving better results than comparison methods in metric B (cropping) and comparable result in metric C (distortion). In terms of metric

D (stability), which is the most important aspect of video stabilization, we outperform the comparison methods on average. For metric E (motion smoothness), our method also performs significantly better. Note that comparing to the method specifically designed for selfie video[31], we are also able to achieve both better metric D (frequency stability) and metric E (motion smoothness) without explicitly modeling the human face.

Visual Comparisons in Video: Besides the quantitative metrics, we also show visual comparison in the supplementary video. For videos with large occlusion (example 2, 4, 8, 14, 21-25), feature track based 2D methods[7, 21] fail due to the difficulty in obtaining long feature tracks. They also produce artifacts in videos coupled with other effects: extreme motion (example 15), motion blur (example 5), rolling shutter (example 3 and 5) and parallax (example 12). 3D methods[18] also cannot produce satisfactory results since structure from motion is not suitable for dynamic scenes in general. The optical flow based method[20] failed in challenging cases like example 8, since its heuristic on motion completion cannot handle large foreground occlusions. Our method is more robust in these cases. We do not explicitly handle the motion discontinuity, but resort to the continuity regularization (7) and (8) and make it part of the optimization. We are able to handle this complex optimization problem thanks to optimizing the neural network parameters instead of the warp field itself.

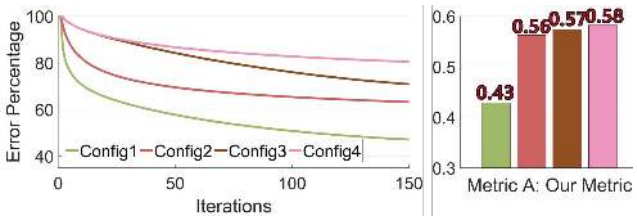

Figure 11. *Effect of network structures. The figure on the left shows the regression error using different network configurations shown in Fig. 4. The right figure shows the effect on the final result using our metric.*

Evaluation of Network Configurations: Now we discuss the effect of neural network configuration on the video stabilization result. As noted in Fig. 4, we have 4 different network configurations in our experiment. In the left part of Fig. 11, we compare the regression errors defined in (9) over 150 optimization iterations using these configurations. Each curve represents the average regression error on a single video segment with length $T = 60$. Networks with more channels (Config 1 and 3) can achieve lower error than networks with the same structure but fewer channels (Config 2 and 4). Fig. 11 also shows that networks with more complex structure (Config 1 and 2) can descend to lower error than networks with the same channels but with simpler structure (Config 3 and 4). We also compare the quantitative evaluation of the result videos using different network configurations in the right part of Fig. 11. The values are averaged over the 25 example videos in our paper. The simple networks (Config 2, 3 and 4) cannot achieve equal quality results as the most complex network (Config 1), and the difference in quality among these simple networks are less significant. This proves that the precise network architecture is important in the our case, and we find the Config 1 network is the best for video stabilization.

7. Conclusion

In this paper, we proposed a new video stabilization formulation based on first principles. This formulation leads to a large scale non-convex optimization problem that previous works tried to avoid by proposing various heuristics. We also proposed a novel CNN based optimization routine for this problem, which does not require a large dataset and is re-trained on each single video. Our method can be applied on any video regardless of the complexity of the scene.

The limitation of our method is the computation time. Our method is an offline method which requires about 30min to stabilize a 300-frame video on a GTX1080Ti graphics card. Since we do not focus on computation time in this paper, we believe the algorithm can be further speeded up, for example, using unidirectional optical flow and/or other network structures and channel configurations.

Our work is the first that explores the possibility of applying CNN techniques to video stabilization. An interesting future work would be a universal pre-trained neural network based on a large video dataset, followed by a fast video-specific fine-training pass. We have made preliminary efforts in this direction, but the training on a dataset of video segments does not yet converge. However, we believe that a CNN can be trained with a slight modification of our algorithm, and significantly speed up the video stabilization process.

Acknowledgements. This work was supported in part by the Ronald L. Graham endowed Chair and the UC San Diego Center for Visual Computing.

Appendix A. Pre-Stabilization

Although we have considered the inaccuracy of the original optical flow in Sec. 3, a pre-stabilization is still necessary to reduce the motion and improve the quality of the original optical flow. Moreover, for large motion videos, a large number of the boundary pixels have no correspondence in the next frame. This results in artifacts in the boundary region of the output warp field, since these pixels can be warped freely without any constraint from neighboring frames. Therefore, our method pre-stabilizes the video before processing the video with the optimization described in Sec. 3.

In the pre-stabilization phase, we first use KLT[23] to track minimum eigenvalue[23] feature points over all the frames. Denote a feature point at time t as $\mathbf{f}_{i,t}$ and its correspondence in $t+1$ as $\mathbf{f}_{i,t+1}$ respectively. We solve for a per-frame 2D affine transformation matrix K_t such that the integral of squared second derivative is minimized:

$$E(K) = \sum_{i,t} \|K_t f_{i,t} - K_{t+1} f_{i,t+1}\| \tag{11}$$

The solved K_t are used to transform the frames of the input video, and the result is cropped to a rectangle as the output of the pre-stabilization phase. Note that the objective (11) is essentially similiar to (5), but computed on only a sparse set of feature points. The output of this pre-stabilization is used as the input for Sec. 3 and the rest of the paper.

References

[1] P. Bhat, C. L. Zitnick, N. Snavely, A. Agarwala, M. Agrawala, M. Cohen, B. Curless, and S. B. Kang. Using photographs to enhance videos of a static scene. In *EGSR*, 2007.

[2] C. Buehler, M. Bosse, and L. McMillan. Non-metric image-based rendering for video stabilization. In *IEEE CVPR*, 2001.

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

[4] G. Eilertsen, J. Kronander, G. Denes, R. Mantiuk, and J. Unger. HDR image reconstruction from a single exposure using deep CNNs. *ACM Trans. Graph.*, 2017.

[5] M. L. Gleicher and F. Liu. Re-cinematography: Improving the camerawork of casual video. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5(1), Oct. 2008.

[6] A. Goldstein and R. Fattal. Video stabilization using epipolar geometry. *ACM Trans. Graph.*, 31(5), Sept. 2012.

[7] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *IEEE CVPR*, 2011.

[8] M. He, J. Liao, P. Sander, and H. Hoppe. HDR image reconstruction from a single exposure using deep CNNs. *ACM Trans. Graph.*, 2018.

[9] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR*, 2018.

[10] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. In *Stanford CS Tech Report*, 2011.

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[12] J. Kopf. 360° video stabilization. *ACM Trans. Graph.*, 2016.

[13] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017.

[14] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[15] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *CVPR*, 2016.

[16] K. Lin, N. Jiang, S. Liu, L. Cheong, M. Do, and J. Lu. Direct photometric alignment by mesh deformation. 2017.

[17] C. Liu. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2009.

[18] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3D video stabilization. *ACM Trans. Graph.*, 28(3), July 2009.

[19] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30(1), Feb. 2011.

[20] S. Liu, L. Yuan, P. Tan, and J. Sum. Steadyflow: Spatially smooth optical flow for video stabilization. In *IEEE CVPR*, 2014.

[21] S. Liu, L. Yuan, P. Tan, and J. Sun. Bundled camera paths for video stabilization. *ACM Trans. Graph.*, 32(4), July 2013.

[22] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Anal. Mach. Intell.(PAMI)*, 28(7), July 2006.

[23] J. Shi and C. Tomasi. Good features to track. In *IEEE CVPR*, 1994.

[24] B. M. Smith, L. Zhang, H. Jin, and A. Agarwala. Light field video stabilization. In *IEEE ICCV*, 2009.

[25] J. Sun. Video stabilization with a depth camera. In *IEEE CVPR*, 2012.

[26] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017.

[27] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. *CVPR*, 2018.

[28] M. Wang, G. Yang, J. Lin, S. Zhang, A. Shamir, S. Lu, and S. Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE Transactions on Image Processing*, 2019.

[29] Y. S. Wang, F. Liu, P. S. Hsu, and T. Y. Lee. Spatially and temporally optimized video stabilization. *IEEE Trans. Visual. and Comput. Graph.*, 19(8), Aug 2013.

[30] S. Xu, J. Hu, M. Wang, T. Mu, and S. Hu. Deep video stabilization using adversarial networks. *Computer Graphics Forum*, 2018.

[31] J. Yu and R. Ramamoorthi. Selfie video stabilization. In *ECCV*, 2018.