Seeing More Clearly: A Video Stabilizer using CNN with Speeded-up Robust features

Junting Wang

Peddie School

Hightstown, NJ

[jwang-25@peddie.org](mailto:jwang-25@peddie.org)

**Abstract: This paper proposes a video stabilization system based on Convolutional Neural Network (CNN). This system could theoretically achieve real-time video stabilization with a powerful hardware. Currently, users can use this system for offline video stabilization. The stabilization page includes two monitor area where one shows the unprocessed video and the other showing the stabled video making the difference clear to users. Compared with similar products, our product is low-cost and user-friendly. The stabilization is achieved through deep learning using the ResNet-50 structure and Siamese Network. I also used SURF, TV-L1, PyQt5, and ORB in this project. Our testing outputs show that all proposed functions are realized.**

***Keywords: video stabilization, CNN, deep learning***

1. **Introduction**

*A. Motivation*

Nowadays, video stabilization is a huge need for many people. We can get a ton of results when searching for “video stabilizer” on google play or app-store. Most people have this need to a fair extent since a lot of us take videos. A video can be very shaky if filmed with a hand-held camera or phone. In this case, user could use a stabilization system to process the video in order to improve its stability.

Personally, I also take videos during my day, and I hate bringing the bulky stabilizers with me when I plan to take a video. Furthermore, there are also a lot of cases when I just took a video without planning. It’s not possible for me to take a stabilizer with me wherever I go. Having a free and convenient way to stabilize videos is essential.

In the past few years, convolutional neural network (CNN) is widely used in processing images and videos. CNN performs well in vision and graphics field which makes building this system plausible. Based on my knowledge, most video stabilization studies are still centered around improving on traditional video stabilization methods instead of the deep learning method which has more potential in the future.

Because the lack of stabilization software that utilizes deep learning, I designed this system that could take advantage of our hardware’s resources and the advancement our society made in deep learning, especially visual source processing with CNN. This system provides video stabilization to users in a user friendly and free way.

*B. Commercial*

Currently, there are several similar commercial products in the market but they all have some disadvantages.

1. Physical camera/phone stabilizer

Using a physical stabilizer would most likely provide users with the best outcome. However, it is huge and bulky which increases the load for users. A good stabilizer is also costly with a stabilizer for mobile phones (figure 1) at $150 and a small camera stabilizer (figure 2) for $280.



A close-up of a camera

Description automatically generated with medium confidence

Figure 1 Figure 2

1. Adobe Premiere Pro

This is a very common application used for video editing and provides a built-in video stabilization function. It gives a fair outcome after processing. However, Adobe Premiere Pro is expensive with a cost of $21/month and it utilizes traditional video stabilization methods.

Icon

Description automatically generated

Figure 3. Adobe Premiere Pro

1. Other applications

All the other video stabilization applications can only stabilize a video to a fair extent, and I didn’t find any existing commercialized products that utilizes deep learning to achieve video stabilization.

A picture containing text, iPod, electronics

Description automatically generated

Figure 4. example of other stabilization apps

*C. Consideration*

* Convenient to use

All the processing algorithms are included in a system is clear and user friendly. The UI is simple without any unnecessary parts.

* Outcome

By comparing with Adobe Premiere Pro 2020, we have achieved a decent outcome for a new approach to video stabilization.

* Cost: None

We don’t include any hardware, so users could simply download the code onto their computer.

1. **Related Work**

*A. Traditional Methods (2D & 3D)*

2D and 3D video stabilization methods are currently still the most used commercialized video stabilization method. These two methods track key points in the video and build a smoothed 2d and 3d camera paths, respectively. These methods are very well developed so they can achieve satisfying results. However, these methods may not work if the key feature is hard to identify. They also require multiple frames, more the better to recognize the trajectory to stabilize videos, which makes it impossible for these methods to be used in real time video stabilization. Almost all commercialized products utilize one of these two video stabilization methods.

*B. Deep Learning (DL) Methods*

As the deep learning technologies develop, more people try to use DL methods to stabilize videos and even achieve real-time video stabilization. The common deep learning methods include utilizing convolutional neural networks (CNN) and deep neural networks (DNN). Xu et al. 2018 used generative and spatial transformer networks, which is better than the Adobe Premier stabilizer in multiple areas but not in the stabilization quality. Wang et al. 2018 proposed the StabNet stabilization network based on CNN and Siamese Network. StabNet only uses past frames in the process so in theory it is capable of real time video stabilization. Xu et al. 2020 developed a Deep Unsupervised Trajectory (DUT) based stabilization framework that utilizes DNN to predict camera trajectory. This method shows strong stabilization result but still relies on trajectory estimation.

In this paper, we analyze the method used behind StabNet and quantized both the advantages and disadvantages of this method. We also incorporated StabNet into a user-friendly software.

1. **System Description**

#### A. Overall System Logic

Training:

Using:

In each of the choices:

#### B. Convolutional Neural Network (CNN)

CNN is a feed forward neural network that is widely used for dealing with images. It consists of three kinds of layers: convolutional layers, pooling layers, and fully connected layers. In the network, convolutional layers and pooling layers make up the first phase of preprocessing the data. During this phase, the two kinds of layers work together to extract important features from the image. The image is than sent to the fully connected layers where the image will be classified. The goal of CNN (figure 5) can be simply understood as it is trying to find a minimum loss value (y-value) by changing its w1 (x-value). It will get the derivative of the current point and see if the derivative is positive or negative. If the derivative is positive, the network will decrease w1 value. On the other hand, if the derivative is negative, the network will increase w1 value. This set of rules create a problem that the network may have found a local minimum instead of the global minimum. To avoid this, a momentum is used to push the result somewhat so it may be able to jump out of the local minimum.

Chart, line chart

Description automatically generated

Figure . the difference between local min and global min (try to find)

*C. ResNet50*

ResNet50 is a kind of CNN. It is the training network structure of StabNet. The ResNet50 network contains 4 blocks including 3, 4, 6, and 3 units (figure 6). ResNet is special because of its skip connections, which allows it to jump over some layers (figure 7). These skip connections allow ResNet to train faster.

Table

Description automatically generated

Figure 6. Chart of ResNet50's layers

Diagram

Description automatically generated

Figure 7. A visualization of how ResNet jumps over layers

*D. Siamese Network*

Siamese Network (figure 8) is a structure that contains two identical networks. In this structure, the two identical networks work simultaneously with two different inputs and at last compare the output with each other to get the loss feature, including stability loss and temporal loss. We can get the stability loss by comparing our processed frame with the stable frame. We can get the temporal loss through comparing our current processed frame with the previous processed frame.

Diagram

Description automatically generated

Figure 8. Process map for Siamese Network

*E. Loss Calculation*

During the training stage, the StabNet model calculates stability loss & temporal loss, which are a part of the two-term loss function. The overall loss function is:

Text

Description automatically generated

Figure 9. Overall loss function

The Lstab and Ltemp represents the stability loss and temporal loss, respectively.

1. Stability Loss

The stability loss is the loss between processed frame and stable frame. The stability loss function is:

Text

Description automatically generated

Figure 10. Stability loss function

1. Temporal Loss

The temporal loss is the loss between current processed frame and previously processed frame. The temporal loss function is:

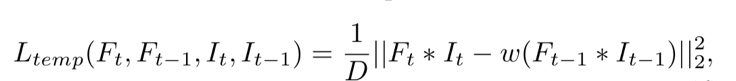


Figure 11. Temporal loss function

*F. Other functions & methods*

1. SURF & ORB

SURF’s full name is “Speeded-Up robust features.” As implicated by its name, SURF is a fast and robust algorithm that extracts important features (figure 12) from an image and comparing features between two images. It can effectively solve the problem that features representing the same thing may have different sizes in different images.

I tried to use OpenCV with SURF to visualize the output using the command “cv2.xfeatures2d.SURF\_create”. Unfortunately, OpenCV doesn’t have the right to use xfeatures2d because of patent issues. The most obvious solution is to download an older version of OpenCV (3.4.1.15 or before). However, pip doesn’t support installing such an old version and I failed when trying to install locally with the “.whl” file. I changed to using another way, the ORB (Oriented Fast & Rotated Brief) method.

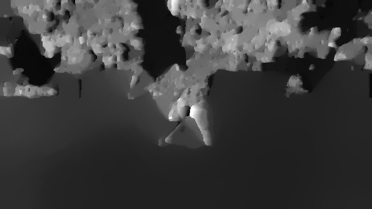
A picture containing shape

Description automatically generated

Figure 12. A visual description of potential feature points

1. TV-L1 visualization

TV-L1 is an algorithm that estimates the optical flow. Optical flow is used to describe a thing that changes it’s position between two frames because of movement. This movement includes the object’s movement and the camera’s movement. It is like a two dimensional vector field, each vector represents the change in position of the object from the previous frame to the current frame. In simple explanation, TV-L1 is used to estimate how an object moves.



A picture containing image

Description automatically generatedA picture containing several

Description automatically generated

Figure 13. Visualization of TV-L1

1. Batch Normalization (BN)

BN effectively reduces internal covariate shift, which is the change in distribution of network activations due to change in network parameters during training, through feature scaling. BN is used to make artificial neural networks faster and more stable. In some cases, BN is able to replace dropouts (randomly deactivating some nodes when processing a batch) in solving overfitting.

*G. PyQt5 system development*

To make a more convenient way for users to stabilize their videos, I created a system using Qt designer and PyQt5. I designed the start page to have two choices (figure 14), normal stabilization (figure 15) and record and stabilize (figure 16).

Graphical user interface

Description automatically generated

Figure 14. Start page

A screenshot of a computer

Description automatically generated

Figure 15. Inside the normal stabilization mode

A computer screen with a blue background

Description automatically generated with low confidence

Figure 16. Inside the record & stabilize mode

1. **Experiments and Results**

*A. Metrics*

1. Field of View (FOV) & Sharpness

Existing video stabilization methods often reduce the FOV and sharpness of the original video during the process. With a smaller FOV, more information are cut from the original video (the four sides), which may cause the user to lose an important part of the video. The sharpness of the video is also important since it determines whether the content in the video is clear.

We decided to use a simple yet powerful metric to measure the FOV and sharpness of our video at the same time, the number of pixels in a frame of the video. This is directly linked to the sharpness and contributes to identifying the FOV.

1. Stability

Stability is undoubtfully an important metric for video stabilization since the main goal of any video stabilization method is to make the video more stable.

We decided to use a pre-made algorithm that utilizes OpenCV to determine the stability of a video. This algorithm processes single frames at a time and measures the frame’s blurriness. A high blurriness means the video is less stable.

*B. Results and Evaluation*

The comparison included the first three unstable videos from the DeepStab dataset as the original video. It compares the results of stabilization through our method and through Adobe Premiere Pro 2020 (Pr). As shown in figure 17, the average stableness of our method is close to the traditional method of Pr, and both stabilization methods greatly increase the stableness when compared with the original unstable video. As shown in figure 18, both stabilization methods decrease the pixels per frame of video, leading to less FOV and sharpness.

Figure 17. Stability comparison

Figure 18. FOV & Sharpness comparison

1. **Final Product**

*A. Normal Stabilization*

In normal mode, users can choose the input and output path through the select button and control playing / stopping the video.

*B. Record & Stabilize*

The record mode is provided so there is an easy way for users to record in this system and the video will be automatically stabilized after the recording is stopped. In this mode, users can choose the camera through the dropdown menu. They can control the start and stop of recording.

1. **Future**

While our system is able to use CNN based deep learning method to achieve a near-commercialized method stability, it decreases the sharpness of the video more than Pr, the commercialized video stabilizer. The future work includes modifying the StabNet model to achieve less compression. Another limitation of our system is that it’s unable to stabilize videos in real-time while it is theoretically possible. The current system’s processing time is too long for real-time stabilization. Another future work could be modifying both the model and pre-processing stages to decrease the time complexity of the code to achieve real-time video stabilization.

1. **Conclusion**

In this paper, I present a graphical control panel that incorporates the StabNet deep learning network to achieve video stabilization. This system is designed to create a user-friendly stabilization system that utilizes advanced deep learning methods. I used StabNet model, ORB, PyQt5, and Qt designer. In normal mode, after users select the original video and clicked the “stabilize” button, the original video will be sent for processing. The record mode is the same only with the input source being the recording. During the processing phase, the video is first changed into frames and the frames are sent into the testing document. Through my experiments, I can conclude that our stabilization method can achieve the stability of commercialized products. When comparing with Premiere, data shows our method obtains one trial of better performance and two trial with 20% and 11% less performance. This method still needs to improve upon the FOV and resolution.

1. **Reference**

Zhang, Fang Lue, et al. “Coherent Video Generation for Multiple Hand-Held Cameras with Dynamic Foreground.” *Computational Visual Media*, vol. 6, no. 3, Tsinghua University Press, Sept. 2020, pp. 291–306, doi:10.1007/s41095-020-0187-3.

Xu, Yufei, et al. *DUT: Learning Video Stabilization by Simply Watching Unstable Videos*. https://github.com/Annbless/DUTCode.

Liu, Yu-Lun, et al. *Hybrid Neural Fusion for Full-Frame Video Stabilization*. https://alex04072000.github.io/FuSta/.

Wu, Rouwan, et al. “Robust Global Motion Estimation for Video Stabilization Based on Improved K-Means Clustering and Superpixel.” *Sensors*, vol. 21, no. 7, MDPI AG, Apr. 2021, doi:10.3390/s21072505.

*2019 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2019.

Institute of Electrical and Electronics Engineers, and IEEE Signal Processing Society. *2019 IEEE International Conference on Image Processing (ICIP) : Proceedings : September 22-25, 2019, Taipei International Convention Center (TICC), Taipei, Taiwan*.

Choi, Jinsoo, and In So Kweon. “DIFRINT: Deep Iterative Frame Interpolation for Full-Frame Video Stabilization.” *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 3732–36, doi:10.1109/ICCVW.2019.00463.

Su, Shuochen, et al. *Deep Video Deblurring for Hand-Held Cameras*. https://www.cs.ubc.ca/labs/imager/tr/2017/DeepVideoDeblurring.

Kim, Byeonghwi, et al. “Automated White Blood Cell Counting in Nailfold Capillary Using Deep Learning Segmentation and Video Stabilization.” *Sensors (Switzerland)*, vol. 20, no. 24, MDPI AG, Dec. 2020, pp. 1–22, doi:10.3390/s20247101.

Ma, Tiezheng, et al. “Effective Video Stabilization via Joint Trajectory Smoothing and Frame Warping.” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, IEEE Computer Society, Nov. 2020, pp. 3163–76, doi:10.1109/TVCG.2019.2923196.

Ali, Muhammad Kashif, et al. *Learning Deep Video Stabilization without Optical Flow*.

Huang, Chia-Hung, et al. *StableNet: Semi-Online, Multi-Scale Deep Video Stabilization*.

Niskanen, Matti, et al. “Video Stabilization Performance Assessment.” *2006 IEEE International Conference on Multimedia and Expo, ICME 2006 - Proceedings*, vol. 2006, 2006, pp. 405–08, doi:10.1109/ICME.2006.262522.

Yu, Jiyang, and Ravi Ramamoorthi. *Robust Video Stabilization by Optimization in CNN Weight Space*.

Zhao, Minda, and Qiang Ling. “Adaptively Meshed Video Stabilization.” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 9, Institute of Electrical and Electronics Engineers Inc., Sept. 2021, pp. 3504–17, doi:10.1109/TCSVT.2020.3040753.

Wang, Miao, et al. *Deep Online Video Stabilization*.

Guilluy, Wilko, et al. “Video Stabilization: Overview, Challenges and Perspectives.” *Signal Processing: Image Communication*, vol. 90, Elsevier B.V., Jan. 2021, doi:10.1016/j.image.2020.116015.

Shi, Zhenmei, et al. *Deep Online Fused Video Stabilization*.

Xu, Sen Zhe, et al. “Deep Video Stabilization Using Adversarial Networks.” *Computer Graphics Forum*, vol. 37, no. 7, Blackwell Publishing Ltd, Oct. 2018, pp. 267–76, doi:10.1111/cgf.13566.

Wang, Miao, et al. “Deep Online Video Stabilization with Multi-Grid Warping Transformation Learning.” *IEEE Transactions on Image Processing*, vol. 28, no. 5, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 2283–92, doi:10.1109/TIP.2018.2884280.

*Background Scene Enlargenment in Selfie Video through Deep Neural Network: A Survey*. doi:10.1089/cyber.2009.

Institute of Electrical and Electronics Engineers, et al. *2020 IEEE International Conference on Multimedia and Expo Workshops (ICMEW) : 06-10 July 2020, London, UK*.

Rosebrock, Adrian. “Blur Detection with Opencv.” *PyImageSearch*, PyImageSearch, 17 Apr. 2021, https://pyimagesearch.com/2015/09/07/blur-detection-with-opencv/.

1. **Appendix**

*A. ORB visualization*

Text

Description automatically generated

Figure 19. ORB full code

*B. TV-L1 visualization*

Graphical user interface, text

Description automatically generated

Figure 20. TV-L1 visualization full code

*C. Stability evaluation*

Graphical user interface, text, application

Description automatically generated

Figure 21. Stability evaluation code for mass amounts of frames

*D. Stabilizing model*

Sample code (block\_1, unit\_1):

|  |
| --- |
| self.resnet\_v2\_50\_conv1\_Conv2D = self.\_\_conv(2, name='resnet\_v2\_50/conv1/Conv2D', in\_channels=3, out\_channels=64, kernel\_size=(7, 7), stride=(2, 2), groups=1, bias=True)  self.resnet\_v2\_50\_block1\_unit\_1\_bottleneck\_v2\_preact\_FusedBatchNorm = self.\_\_batch\_normalization(2, 'resnet\_v2\_50/block1/unit\_1/bottleneck\_v2/preact/FusedBatchNorm', num\_features=64, eps=1e-5, momentum=0.003)  self.resnet\_v2\_50\_block1\_unit\_1\_bottleneck\_v2\_shortcut\_Conv2D = self.\_\_conv(2, name='resnet\_v2\_50/block1/unit\_1/bottleneck\_v2/shortcut/Conv2D', in\_channels=64, out\_channels=256, kernel\_size=(1, 1), stride=(1, 1), groups=1, bias=True)  self.resnet\_v2\_50\_block1\_unit\_1\_bottleneck\_v2\_conv1\_Conv2D = self.\_\_conv(2, name='resnet\_v2\_50/block1/unit\_1/bottleneck\_v2/conv1/Conv2D', in\_channels=64, out\_channels=64, kernel\_size=(1, 1), stride=(1, 1), groups=1, bias=None)  self.resnet\_v2\_50\_block1\_unit\_1\_bottleneck\_v2\_conv1\_BatchNorm\_FusedBatchNorm = self.\_\_batch\_normalization(2, 'resnet\_v2\_50/block1/unit\_1/bottleneck\_v2/conv1/BatchNorm/FusedBatchNorm', num\_features=64, eps=1e-5, momentum=0.003)  self.resnet\_v2\_50\_block1\_unit\_1\_bottleneck\_v2\_conv2\_Conv2D = self.\_\_conv(2, name='resnet\_v2\_50/block1/unit\_1/bottleneck\_v2/conv2/Conv2D', in\_channels=64, out\_channels=64, kernel\_size=(3, 3), stride=(1, 1), groups=1, bias=None)  self.resnet\_v2\_50\_block1\_unit\_1\_bottleneck\_v2\_conv2\_BatchNorm\_FusedBatchNorm = self.\_\_batch\_normalization(2, 'resnet\_v2\_50/block1/unit\_1/bottleneck\_v2/conv2/BatchNorm/FusedBatchNorm', num\_features=64, eps=1e-5, momentum=0.003)  self.resnet\_v2\_50\_block1\_unit\_1\_bottleneck\_v2\_conv3\_Conv2D = self.\_\_conv(2, name='resnet\_v2\_50/block1/unit\_1/bottleneck\_v2/conv3/Conv2D', in\_channels=64, out\_channels=256, kernel\_size=(1, 1), stride=(1, 1), groups=1, bias=True) |