Applying Video Magnification to Stabilized Handheld Video

Daniel Peterson
pet01704@umn.edu

Andrea Walker
walk0798@umn.edu

Claire Yang
yang5063@umn.edu

Abstract

Video magnification allows us to see small motions which are unnoticeable to the human eye. The existing techniques assume that there is a fixed point of perspective for the camera. With the presence of camera movement, current video magnification techniques fail. We extend Oh et al.*'s learning-based video motion magnification method so that it is robust for the motion magnification of handheld video.*

1. Introduction

Human vision is limited in its ability to track small vibrations and subtle changes in our everyday environments. Motion magnification can be used to emphasize the motion between frames of a video, allowing humans to recognize changes through viewing the magnified video. This task becomes more difficult with larger magnification factors, since the subtle motions eventually become virtually indistinguishable from noise. Thus, all of the past studies have focused on motion magnification with the assumption that the camera is completely still. However, many videos are shot through handheld cameras like cellphones, which results in natural oscillations from the camera person's hand. Our work focuses on how we might combine stabilization with motion magnification so that it can be applied to handheld videos.

Motion magnification has applications across diverse fields, including healthcare, engineering, and manufacturing. In the healthcare industry, a heartbeat can be extracted from a short video of the patient through amplifying the subtle Newtonian head motion caused by the pulse [4] [27] [11] [1]. This could be an effective way of reducing patient contact while treating infectious diseases. In addition, motion magnification techniques have been used to monitor infant respiration [2], which could be implemented as an additional monitoring system in a NICU or for general application in improved baby monitors. In engineering and construction, motion magnification can be used to measure building sway, which can be utilized to test the structural integrity of buildings [5]. A similar application uses motion magnification to amplify the vibration of wind turbines, revealing underlying structural compromise or damage [24]. In the realm of manufacturing, 3D printing has emerged as a new, fast method of developing prototypes and even final products; authors Prakash *et al*. present an application of motion magnification in quality assurance, demonstrating that analyzing magnified videos of the printing process can reveal structural flaws in 3D printed components [3] .

In each of these applications, the quality of the motion magnification hinges on the stability of the camera recording the video footage. Extraneous noise from the motion of the camera can be amplified by the motion magnification algorithm, causing blurring and inaccuracies in the amplification of the magnification target. This requires the use of a tripod or a similar stabilization method for the recording device. However, even using a tripod, instabilities in the video can be introduced. For example, at a construction site, the ground may be shaken by machinery, causing vibrations in the video; in manufacturing, 3D printers have an inherent vibration that can cause motion of the video capture device. In further applications like healthcare, using a hand-held device like a smartphone to capture video is more flexible than requiring the use of a tripod. Thus, developing an algorithm that compensates for camera motion in video magnification would contribute a much more robust approach than the current literature. This would broaden the scope of applications for motion magnification and could bring this technology into more everyday use.

The primary challenges in developing such an algorithm lie in accurate stabilization, and the reduction of blurring and noise resulting from the motion magnification. In addition, it must be sensitive to and accurate for small motions to work in manufacturing domains. The quality of the stabilization is crucial to combating blurring in the motion magnification, since the motion of the camera can be of the same or larger magnitude than the motion we are trying to amplify. Finally, when magnifying videos at large magnification factors, the algorithm cannot distinguish small motions in the video that are not part of the targeted motion magnification, which causes blurring.

In this paper we present a novel end-to-end algorithm which aligns the frames and magnifies the motion in an Eu-

lerian motion magnification algorithm based on the deeply learned methods of Oh *et al*. [21].

2. Related Work

Video Stabilization. Much work has already been done in the area of stabilizing video motion caused by handheld recording devices [7] [13] [14] [16]. The main goal of these video stabilization methods is to smooth the camera motion in order to remove jitter and higher frequencies [17]. There are two categories of approaches that are used to achieve this: 2D and 3D video stabilization techniques [17].

2D techniques represent the changes between frames through 2D transformations (e.g. homography). These transformations are then smoothed and used to warp the frame [17] [14]. Research into smoothing methods has produced a variety of different methods, from Morimoto and Chellappa's simple Gaussian smoothing [19], to using Kalman Filters to estimate the motion trajectory from the past frames [9]. The advantages of a lot of 2D methods are that they are quick and can be used in real time. The disadvantages, however, are their inability both to fully capture the complexity of a scene's depth and to robustly handle translational shifts in viewpoint [13]. However, techniques like SteadyFlow [17] address this problem by using spatial-temporal analysis to get rid of discontinuous regions of optical flow.

3D video stabilization methods reconstruct a camera's position in 3D space so that they can replicate a smoother and more directed motion. [17] [13]. 3D video stabilization is more computationally expensive than 2D video stabilization due to the need to estimate the reconstruction of a 3D scene (if there is no depth sensor data), but it produces superior results for scenes with changes in depth [31] [17]. Recent works [14][31] have introduced new techniques that reduce the computational expense of the reconstruction step.

Video Motion Magnification. The two main existing approaches to motion magnification include the Lagrangian approach and the Eulerian approach. The Lagrangian approach uses optical flow to track, and subsequently magnify, the displacements of individual pixels, so that the motion of any given pixel can be temporally traced from frame to frame using the optical flow model [6] [20] [12]. Since each pixel's motion is tracked explicitly, the initial Lagrangian approach given by Liu *et al*. in [12] was unable to implement real-time motion magnification due to the heavy computational overhead [6]. Their method also suffered from low quality magnification due to amplifying noise [1]. Since [12] was authored in 2005, deep learning has improved optical flow estimation [10], reducing the computational overhead for Lagrangian approaches. However, the Eulerian approach remains the most prevalent motion magnification schema.

Unlike Lagrangian approaches, Eulerian approaches decompose the video frames into alternate representations from which the motion magnification is derived. Four influential papers implementing different decomposition methods are by Wu *et al*. [32], Zhang *et al*. [33], and Wadhwa *et al*. [30] and [29] . In [32], Wu *et al*. represent the relationship between consecutive frames as a first-order Taylor expansion with an amplification factor added to the expansion to enact the motion magnification. Alternatively, Zhang *et al*. use a second derivative band-pass filter and a complex steerable pyramid filter to isolate and magnify acceleration [33]. Lastly, Wadhwa *et al*. contribute two other Eulerian approaches: in [30] the authors use complex steerable pyramids in a method similar to Zhang *et al*., replacing the second derivative filter with a first derivative filter; and in [29] the authors utilize Riesz pyramids to magnify motion in the dominant direction of pixel motion. While our approach falls into the domain of Eulerian motion magnification, it leverages the deep learning method developed by Oh *et al*. to implement the motion magnification [21].

Our Contributions. Our work contributes to the field by introducing a more robust algorithm for processing videos where the raw video footage contains noise from the recorder's motion. This will be achieved through integrating stabilization techniques with deeply-learned Eulerian motion magnification. Our algorithm expands the range of input videos that can be processed to output reliable, accurate motion magnification by no longer requiring a fixed camera perspective. We aim to develop a methodology that is robust enough to handle both small and large motions of the recording device.

3. Methodology

3.1. Baseline

Learning-Based Video Magnification. Our baseline implementation of video magnification is provided by Oh *et al*. in [21]. Oh *et al*.'s work implements a learning based approach to video magnification. One innovation of this approach is the simplification of the problem to consider only two frames of the input video for each frame of the output video. In contrast, other approaches like phase-based video magnification rely on more temporal information. This allowed Oh *et al*. to create a large synthetic data set to train their model.

Their model's representation of frames includes two components, texture and shape. The goal of the model is to preserve the texture information while manipulating the shape information in the output video. This two-part model is resistant to noise, which is generally contained in the texture information. However, this is also a weakness of the model because in some test cases useful movement data is filtered out as texture data.

3.2. Novel Algorithm

3.2.1 Preliminary Work.

Naïve Implementation. To mitigate the effects of camera movement, video stabilization is a logical place to start. Our naïve approach makes no changes to the model from the baseline approach. We first simply align the video using homography before applying the baseline algorithm. The method of aligning the video is described in Appendix G. The result is an aligned video which has had video magnification applied to it.

Intermediate Explorations. After our naïve implementation, we explored four other methods for stabilizing the input video for effective motion magnification. The first two techniques, L1 norm and MeshFlow stabilization, employed camera path smoothing, which aimed to reduce the noise from the camera motion without completely stabilizing the video to a fixed vantage point [8] [15] [18] [23] [28]. Results from these two methods are discussed in Appendix C. In the second two approaches, we sought to leverage 3D scene reconstruction techniques to stabilize the video to a fixed vantage point. Our exploration into using COLMAP and stereo rectification are also discussed in Appendix C [25] [26].

3.2.2 Double Warp Implementation.

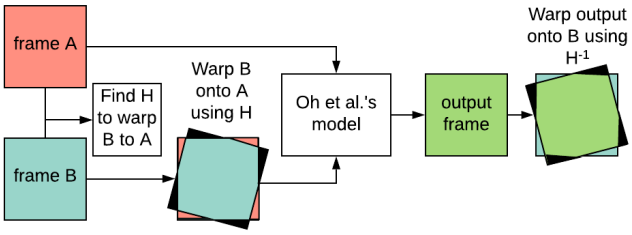

Figure 1. The double warp process

In order to improve upon the naïve approach, we developed a more mature approach which we called the double warp implementation. Frame B is aligned to frame A before being sent into the original model. After magnification, the result is warped back using the inverse perspective transform. This process is shown in Figure 1. This implementation makes 3 improvements to the naïve approach.

No pre-processing. Because the frames are aligned and magnified on an individual basis, no pre-processing of the video is necessary. This is superior to the naïve approach because it allows for potential real-time applications of this technology in the future.

Fixing the alignment artifacts. Through the double warp process, only the shared areas between frame A and frame B are magnified in the final output. The edges are unaffected.

Does not affect the alignment of the video. Because the inverse warp is applied after magnification, the perspective of the output video is unaffected, which was our original goal. We are aiming for a similar effect in the learning-based model.

3.2.3 Learning-based Implementation

Our learning-based implementation extended upon Oh *et al.*'s method in three ways: we added a warping function that takes in new inputs to the existing architecture, created new synthetic data that includes changes in camera perspective, and trained the model from scratch using the new data.

Augmenting the Model. Our new model builds upon Oh *et al.*'s baseline model by adding an additional input and an intermediate warping function. As can be seen in Figure 8, Oh *et al.*'s model takes two input frames X_A and X_B which are each processed by an encoder that outputs the two frames' shape and texture representations. The manipulator then magnifies the difference between their shape representations in order to output the magnified shape representation. The magnified shape representation is then combined with the texture representation in the decoder, which outputs the final magnified frame. Our improved model retains Oh *et al.*'s original encoder and decoder, but augments the existing manipulator by adding a warping function. This requires the 3×3 transformation matrix in addition to the other inputs. The overall architecture of our model can be seen in Figure 2. The two additions to the model, the additional input and the warping function are discussed subsequently.

The additional input was the estimated 3×3 matrix H_{BA} describing the transform from frame X_A to frame X_B:

$$\mathbf{X_B} = H_{BA} \cdot \mathbf{X_A} \tag{1}$$

In order to standardize between the training and testing data, we cropped the input video frames by 50 pixels on all edges.

The new manipulator modifies the original architecture as follows. Borrowing notation from Oh *et al.*'s paper [21], let the encoder be represented by $G_e(\cdot)$ and the manipulator by $G_m(\cdot)$. Likewise, denote the texture and shape representations output by the encoder as $\mathbf{V} = G_{e,texture}(\mathbf{X})$ and $\mathbf{M} = G_{e,shape}(\mathbf{X})$. Our implementation processes $\mathbf{M_A}$ with a warping function that transforms $\mathbf{M_A}$ to the perspective of $\mathbf{M_B}$ using the pre-calculated homography transform from Equation 1. This warped $\mathbf{M'_A}$ is then input into the manipulator and compared to $\mathbf{M_B}$ as before.

Oh *et al.*'s network performs a non-linear magnification through two functions, $g(\cdot)$ and $h(\cdot)$, which represent a 3×3 convolution followed by ReLU, and a 3×3 convolution

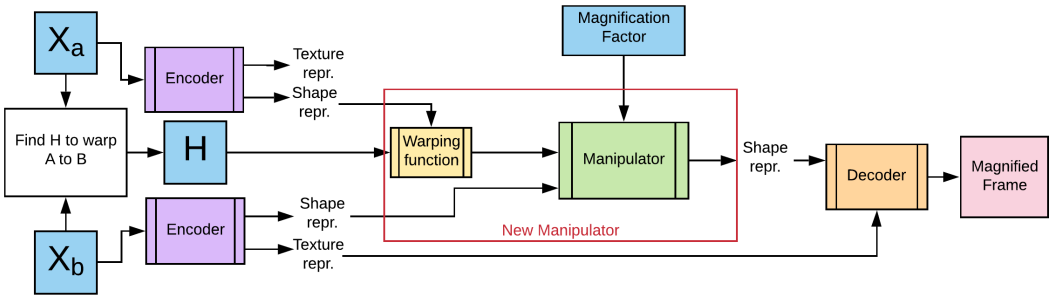

Figure 2. Our new model architecture.

followed by a 3×3 residual block, respectively. Then the non-linear magnification performed by Oh *et al.*'s network is expressed as follows:

$$G_m(\mathbf{M_a}, \mathbf{M_b}, \alpha) = \mathbf{M_a} + h(\alpha \cdot g(\mathbf{M_b} - \mathbf{M_a})) \quad (2)$$

where α is the magnification factor.

By combining the warping function and manipulator together to define the new manipulator m' in our model, the magnification is now expressed as

$$G_{m'(\mathbf{M_a},\mathbf{M_b},\alpha,H_{BA})=f(\mathbf{M_a},H_{BA})+h(\alpha \cdot g(\mathbf{M_b}-f(\mathbf{M_a},H_{BA})))} \quad (3)$$

with $f(\cdot)$ referring to the transformation performed by the warping function, defined below.

To apply the homography transform to $\mathbf{M_A}$, we must account for two things. First, the training data is trimmed by 50 pixels after being warped. Second, the encoder downsamples the input frames, so that each channel of the shape representation has dimensions $\frac{h}{2} \times \frac{w}{2}$ where h and w are the dimensions of the cropped input frames.

To account for these two things, we upsampled the image to dimensions $h \times w$, and then translated the pixel locations by 50 pixels in each direction. After applying the homography, the pixels were transformed back to the cropped size and then downsampled to the original $\frac{h}{2} \times \frac{w}{2}$ image space. Mathematically this transform is expressed:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & -25.25 \\ 0 & 0.5 & -25.25 \\ 0 & 0 & 1 \end{bmatrix} H_{BA} \begin{bmatrix} 2 & 0 & 50.5 \\ 0 & 2 & 50.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

which is the pixel-by-pixel transform implemented by the function $f(\cdot)$ in the warping function.

Generating New Training Data In addition to modifying the network architecture, we designed a new dataset for training the model. In the baseline approach by Oh *et al.*, the model had been trained using a synthetic dataset that was robust to illumination changes between frames, blurry backgrounds, and low contrast objects. We noticed, however, that this dataset does not include changes of camera perspective between pairs of frames. We generated new synthetic training data that includes changes in perspective to improve the results of the deeply-learned approach applied to handheld video.

Figure 3. Sample training data. The top row contains (left to right) the original Frames A,B,C, and magnified output from Oh *et al.* [21]. The bottom row contains the same frames from our new generated training data.

Our new training data was based on Oh *et al.*'s original training dataset that consisted of 100,000 384×384 image samples. Each sample consisted of four frames, denoted frame A, frame B, frame C, and the amplified frame. Frame A represents the first frame, while frame B and frame C represent a frame at a different timestep in a video. The foreground objects shift slightly between frame A and frame B. Frame C is identical to frame B, except for some illumination changes. The foreground objects in the amplified frame are each moved by a known factor of their original movement between frame A and frame B. Thus, the amplified output is the ground truth for the magnified motion output between frame A and frame B (see Figure 3). We artificially introduced camera motion to each data sample in the dataset by generating a random homography matrix (see Appendix H), and applying it to frame B, frame C, and the amplified frame.

Because applying a random homography transform to each sample creates black borders around the edges, every sample was cropped by 50 pixels on all sides. Thus our new training dataset consisted of 100,000 284×284 samples.

Training the Model After generating the new training image dataset, we associated each sample with its homography transform and used it to train our model from scratch. Our model's loss function was the same as Oh *et al*.'s, where we are trying to minimize the l_1-loss between the network output and the ground truth amplified frame for each sample. In order to minimize this loss, we used stochastic gradient descent with a batch size of 4. We trained the network for 274,000 steps on the 100,000 sample dataset in order to get our deeply-learned network.

4. Testing Data Description

Our input videos range in length between one and four seconds. Each video varied in level of camera movement, the ratio of keypoints that were moving in the scene (see Appendix E), and the amount of magnification necessary to properly magnify its contents.

5. Results

Figure 4. Sample output with 10 $\times$ magnification. Top: Original frame vs. learning-based implementation; Bottom: Baseline implementation vs. double warp implementation

Figure 5. Sample output with 10 $\times$ magnification. From left to right are pictured: The original frame, baseline implementation, double warp implementation, and learning-based implementation output.

We ran the baseline algorithm, our naïve implementation, our double warp implementation, and our learning-based implementation on a variety of videos and compared the results. We discuss the generalizability of the different approaches across different degrees of camera movement in the input video and the magnification factor. We specifically demonstrate the effectiveness of the double warp implementation on magnifying handheld videos, compared to the baseline. We also discuss the limitations and potential future work concerning our learning-based implementation. A list of comparison videos and frames is available in Appendix F.

5.1. Qualitative Analysis

We define a better result as one that appears less blurry, less noisy, and smoother in the transition between frames. This is similar to the way Oh *et al*. make initial qualitative comparisons between their methods' results, in which they compare the blurriness of the edges and the amount of ringing artifacts.

Effects of Camera Movement. In the naïve implementation, if the camera had wide movements across a scene, homography markedly improved the ability of the video magnification algorithm to magnify movements in the scene, while decreasing the amount of noise and ringing artifacts in the output videos. Despite this improvement, if the camera was translated greatly in space rather than purely rotated about its axis, then the property of homography does not hold between the two images. We investigated using 3D video stabilization techniques and structure from motion techniques in an attempt to fix this problem, as discussed further in Appendix C, but these techniques were less robust than homography and did not give us a satisfactory result. Thus, we continued to use a homography transform in our double warp implementation.

When the camera viewpoint of the video was fairly stable, the naïve implementation resulted in an irregular and jumpy magnified video. We theorized that this is because additional noise may be introduced, as we pre-process the video for alignment before sending it into Oh *et al*.'s model. The double warp implementation improved upon this, as an inverse warp is applied to the magnified frame, which removes irregularities from the final result.

Across all degrees of camera movement, our double warp implementation resulted in less distortion, more preserved edges, and less ringing artifacts than the baseline algorithm. Figure 6 shows that compared to the baseline, the result obtained from the double warp implementation is much clearer, especially for the objects in the scene that are not moving. This is especially noticeable in videos with lots of camera movement, since the baseline algorithm distorts everything in the image, and is not able to preserve any edges. We noticed that even in a video with very subtle

camera motions, the baseline algorithm creates ringing artifacts around the edges that are not present in the results of our double warp implementation. These results show that unlike the baseline, our double warp implementation is able to separate the camera motion and the scene motion, so that the camera motion is not being magnified and distorting the overall result.

Figure 6. Top row: Example of distortion in non-moving objects in the scene. Bottom row: Example of ringing artifacts in the output video. Left image: baseline implementation result. Right image: double warp implementation result.

Magnification Factor. As the magnification factor increased, the noise increased in the results of the baseline implementation, our initial implementation, and our double warp implementation. This finding is consistent with Oh *et al*. 's analysis that the larger the magnification, the harder it is to tell movement apart from noise. This is an existing problem within video magnification that should be addressed in future work, however, our work focuses on enhancing the current methods of video magnification so that they can work for videos taken with unstable cameras.

5.2. Quantitative Analysis

We compare the focus measures of the output video frames by identifying the edges using a Laplacian operator and calculating the resulting variance for each frame. This technique is known as variance of Laplacian, and can be used to calculate a measure of blurriness in the image [22]. The Laplacian operator is defined as convolving a single channel of an image with a 3x3 negative Laplacian kernel.

After convolving the image with this kernel, we calculate the variance of the response. As the variance increases, the amount of blurriness in the image decreases. Figure 7 shows that the resulting frames of the double warp implementation have a higher focus measure than the resulting frames of the baseline algorithm across the whole video. This means that our double warp implementation quantitatively outperforms the baseline algorithm, in that the resulting magnified video has less noise and more preserved edges.

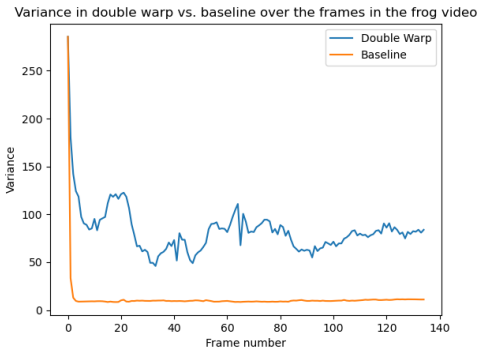

Figure 7. Comparison of the variances between the baseline frames and the double warp frames in the frog video

Results of the Learning-Based Implementation Our learning-based implementation did not give us the results we were looking for, since there was no discernible difference in motion between the original video and the output video. However, there is promise in future work being able to train the network. The edges were well preserved in the output, but in every output, there existed a blue artifact that tracked with the edges of the video (see Appendix F.3). We are not sure about the cause of this blue artifact, but it might be related to the format of the training data. Additionally, we know that our method didn't completely fail, since it is able to reconstruct the scene using the representations that it learned. We hypothesize that the network would learn better if we increased the disparity between the camera movement and the scene's movement. This could be done by developing a method to represent camera movement within the network and creating a different training dataset that includes an additional intermediate frame that has camera movement but no movement of the objects in the scene.

6. Conclusion

Previous video magnification techniques required a perfectly stable camera to be effective. Our double warp implementation improved significantly upon the baseline, as it can magnify motion in handheld video across a range of camera motions. Motions within the scene that were lost in the baseline because of camera movement were recovered and magnified in the double warp technique. Since all of this was achieved without affecting the alignment of the original video, this development allows video magnification technology to be used on more real world applications. In addition to our double warp implementation, our experiment into a newly trained learning-based technique shows promise and is informative for further work for building a superior learning-based model for magnifying motion in handheld videos.

References

[1] Ali Al-Naji and Javaan S. Chahl. Contactless cardiac activity detection based on head motion magnification. *Int. J. Image Graphics*, 17:1–18, 2017.

[2] Ali Al-Naji, Sang-Heon Lee, and Javaan Chahl. An efficient motion magnification system for real-time applications. *Mach. Vision Appl.*, 29(4):585–600, May 2018.

[3] Sakthi Kumar Arul Prakash, Tobias Mahan, Glen Williams, Christopher McComb, Jessica Menold, and Conrad S. Tucker. Detection of System Compromise in Additive Manufacturing Using Video Motion Magnification. *Journal of Mechanical Design*, 142(3), 12 2019. 031109.

[4] Guha Balakrishnan, Fredo Durand, and John Guttag. Detecting pulse from head motions in video. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, page 3430–3437, USA, 2013. IEEE Computer Society.

[5] Young-Jin Cha, Justin G. Chen, and Oral Buyukozturk. Output-only computer vision based damage detection using phase-based optical flow and unscented kalman filters. 2017.

[6] Philipp Flotho, Mayur J. Bhamborae, Lars Haab, and Daniel J. Strauss. Lagrangian motion magnification revisited: Continuous, magnitude driven motion scaling for psychophysiological experiments. *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3586–3589, 2018.

[7] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, page 225–232, USA, 2011. IEEE Computer Society.

[8] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed video stabilization with robust l1 optimal camera paths. In *CVPR 2011*, pages 225–232, 2011.

[9] Yu-Feng Hsu, Cheng-Chuan Chou, and Ming-Yu Shih. Moving camera video stabilization using homography consistency. pages 2761–2764, 09 2012.

[10] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017.

[11] Ramin Irani, Kamal Nasrollahi, and Thomas B. Moeslund. Improved pulse detection from head motions using dct. *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, 3:118–124, 2014.

[12] Ce Liu, Antonio Torralba, William T. Freeman, Frédo Durand, and Edward H. Adelson. Motion magnification. *ACM Trans. Graph.*, 24(3):519–526, July 2005.

[13] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph.*, 28(3), July 2009.

[14] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Trans. Graph.*, 30(1), Feb. 2011.

[15] Shuaicheng Liu, Ping Tan, Lu Yuan, Jian Sun, and Bing Zeng. Meshflow: Minimum latency online video stabilization. In *ECCV*, 2016.

[16] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM Trans. Graph.*, 32(4), July 2013.

[17] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Steadyflow: Spatially smooth optical flow for video stabilization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4209–4216, 09 2014.

[18] S. Liu, L. Yuan, P. Tan, and J. Sun. Steadyflow: Spatially smooth optical flow for video stabilization. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4209–4216, 2014.

[19] Carlos Morimoto and Rama Chellappa. Evaluation of image stabilization algorithms. volume 5, pages 2789 – 2792 vol.5, 06 1998.

[20] Anh Cat Le Ngo, Alan Johnston, Raphael C.-W. Phan, and John See. Micro-expression motion magnification: Global lagrangian vs. local eulerian approaches. *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 650–656, 2018.

[21] Tae-Hyun Oh, Ronnachai Jaroensri, Changil Kim, Mohamed Elgharib, Frédo Durand, William T Freeman, and Wojciech Matusik. Learning-based video motion magnification. *arXiv preprint arXiv:1804.02684*, 2018.

[22] Jose Luis Pech Pacheco, Gabriel Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia. Diatom autofocusing in brightfield microscopy: A comparative study. volume 3, pages 314–317 vol.3, 02 2000.

[23] Zhihang Ren, Jiajia Li, Shuaicheng Liu, and Bing Zeng. Meshflow video denoising. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2966–2970. IEEE, 2017.

[24] Aral Sarrafi, Zhu Mao, Christopher Niezrecki, and Peyman Poozesh. Vibration-based damage detection in wind turbine blades using phase-based motion estimation and motion magnification. *ArXiv*, abs/1804.00558, 2018.

[25] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[27] Li Shan and Minghui Yu. Video-based heart rate measurement using head motion tracking and ica. volume 1, pages 160–164, 12 2013.

[28] Sudheerachary. sudheerachary/mesh-flow-video-stabilization, Sep 2019.

[29] N. Wadhwa, M. Rubinstein, F. Durand, and W.T. Freeman. Riesz pyramids for fast phase-based video magnification. pages 1–10, 2014.

[30] Neal Wadhwa, Michael Rubinstein, Frédo Durand, and William T. Freeman. Phase-based video motion processing. *ACM Trans. Graph.*, 32:80:1–80:10, 2013.

[31] Yu-Shuen Wang, Feng Liu, Pu-Sheng Hsu, and Tong-Yee Lee. Spatially and temporally optimized video stabilization. *IEEE transactions on visualization and computer graphics*, 19:1354–61, 08 2013.

[32] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman. Eulerian video magnification for revealing subtle changes in the world. *ACM Trans. Graph.*, 31(4), July 2012.

[33] Yichao Zhang, Silvia L. Pintea, and Jan C. van Gemert. Video acceleration magnification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 502–510, 2017.

A. Project Contribution Breakdown

Daniel implemented the naïve implementation and the double warp implementation. We all collaborated on the initial presentation and written proposal. Andrea and Claire explored using 3D stabilization techniques and structure from motion in order to supplement the homography matrix transform. Daniel wrote the data generation script and generated the new synthetic training data. Daniel also modified the network architecture in order to include the warping function and new input. Andrea worked on verifying the transform mathematics for the new implementations. Claire debugged and trained the model for the learning-based method. Everyone contributed to running the algorithms to get the results. Everyone collaborated on the final presentation and final report. Andrea created a lot of the figures, equations, and Appendix in this report.

B. Oh *et al.* Network architecture

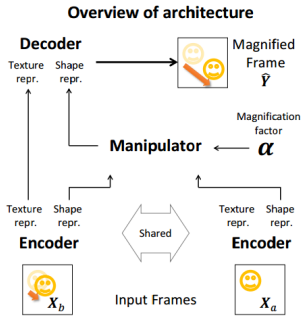

Figure 8. Overview of Oh *et al.*'s network architecture, taken from [21].

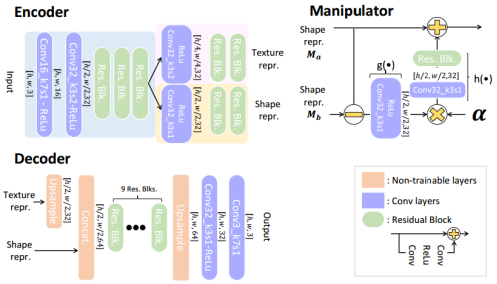

Figure 9. Internal structure of Oh *et al.*'s encoder, manipulator, and decoder, taken from [21].

C. Intermediate Explorations

After our naïve implementation, we explored four main methods for stabilizing the input video for effective motion magnification. The first two techniques employed camera

path smoothing, which aimed to reduce the noise from the camera motion without completely eliminating it. For the first such method (see Figures 10 and 11) we stabilized the video using Grundmann *et al.*'s L1 norm algorithm which used a 2D technique to estimate the camera path and smooth the output video [8]. Secondarily, we used a MeshFlow algorithm based off three MeshFlow papers, [15] [18] [23] and sourced from an open-source repository [28]. This stabilization technique uses optical flow to estimate camera motion and smooth the camera path.

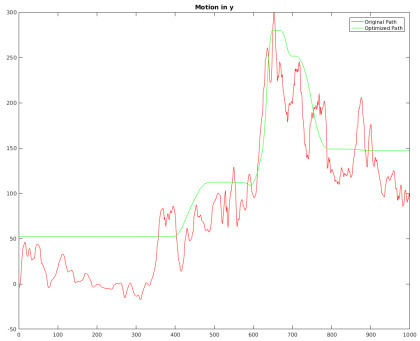

Figure 10. L1 normal path stabilization motion in the y direction

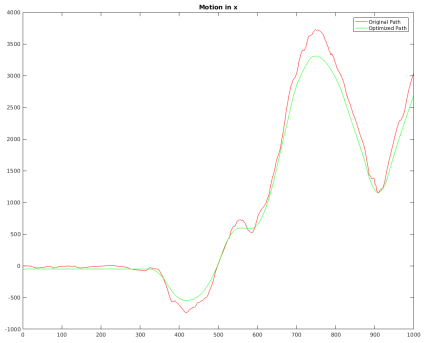

Figure 11. L1 normal path stabilization motion in the x direction

The next two methods we investigated used 3D scene reconstruction techniques to estimate the camera poses relative to the initial frame's perspective. We intended to use these calculated camera poses to stabilize the video as if it was seen from a single camera's vantage point. Initially we decided to use COLMAP to determine the camera poses of each frame in the input video using sparse reconstruction [25] [26]. However, COLMAP is designed to generate intermediate camera poses in addition to the original frames' camera poses. Because of this, there was no way to distinguish the correct correlation between a given input frame and an output camera pose matrix. In lieu of using COLMAP to estimate the camera poses, we explored using stereo rectification to recover the camera poses. We compared each frame with the initial video frame, and used stereo rectification to align the pair to a common perspective. Then we compute the disparity map between the two frames to get the full pixel-by-pixel transformation between the pair of images (see Figure 12 and 14, as well as the derivation in Appendix D).

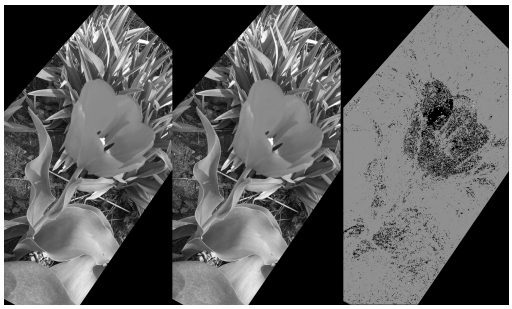

Figure 12. Example of a disparity map between two rectified images

The stereo rectification method for stabilizing the input video showed promise in theory, but we abandoned this approach for three main reasons. First of all, the rectification step warped both images in the image pair greatly, introducing greater potential for error as well as creating an unrepresentative, distorted version of the original video (Figure 13. Second, the stereo rectification method did not reliably choose the correct camera poses out of the four possibilities, resulting in an inability to process an entire video efficiently and accurately. Our final and most crucial reason for not pursuing the stereo rectification method further was its inability to accurately retain the signal of the motion we want to magnify. Stereo reconstruction assumes a static scene; therefore, the portions of the scene that contain the signal to be amplified will be blended between the two representations, resulting in a reconstructed image that does not accurately define the region of motion. Further, the disparity map seeks to align the two images pixel-wise, which does not conserve the translation due to the object motion in the reconstructed frame. Therefore, a video stabilized with stereo reconstruction will effectively have the motion we wish to magnify filtered out by the stabilization algorithm. This defeats our objective of motion magnification, so we abandoned this technique.

In future work, it could be an interesting approach to use stereo reconstruction techniques on a scene where the motion to be magnified is of a known, consistent periodic nature. Then stereo representations could be built up from frames that are sampled at times correlating to the same position within the periodic waveform. Each of these stereo

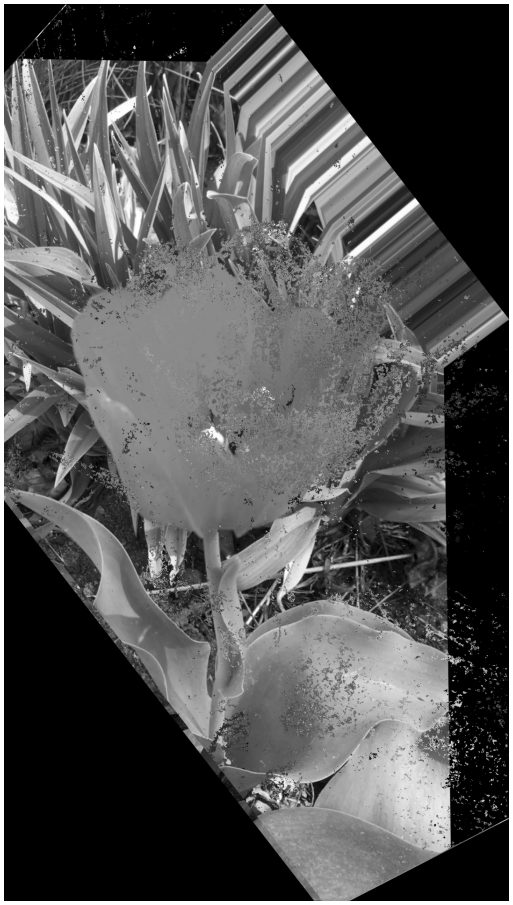

Figure 13. Example of applying the disparity warp transform. Notice the black edges and noise.

representations could then be used to reconstruct a stable version of the video without losing or blurring the motion signal.

D. Math for disparity warp transform

This section defines the mathematical relations for finding the transformation between two images using stereo rectification. First the pair of rectified images are computed using homography:

$$\mathbf{X_{Rec,A}} = H_{RA} \cdot \mathbf{X_A} \tag{5}$$
$$\mathbf{X_{Rec,B}} = H_{RB} \cdot \mathbf{X_B} \tag{6}$$

Solving for $\mathbf{X_B}$, we get

$$\mathbf{X_B} = H_{RB}^{-1} \cdot \mathbf{X_{Rec,B}} \tag{7}$$
$$\mathbf{X_B} = H_{BR} \cdot \mathbf{X_{Rec,B}} \tag{8}$$

Since $\mathbf{X_{Rec,A}}$ is related to $\mathbf{X_{Rec,B}}$ by the disparity map, we can express a pixel-wise correspondence between them as

$$\mathbf{X_{Rec,B}} = \mathbf{X_{Rec,A}} + \tilde{\mathbf{d}} \tag{9}$$

so the final expression for transforming Image A into the Image B perspective space is

$$\mathbf{X_B} = f(\mathbf{X_A}, H_{RA}, H_{BR}, \tilde{\mathbf{d}}) \tag{10}$$
$$= H_{BR} \cdot \left(H_{RA} \cdot \mathbf{X_A} + \tilde{\mathbf{d}}\right) \tag{11}$$

Visually this is represented in Figure 14.

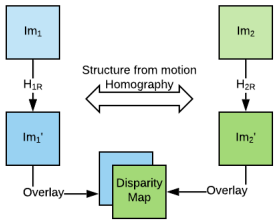

Figure 14. Diagram of obtaining the warping function from the two input frames using structure from motion.

Borrowing notation from Oh *et al.*'s paper [21], let the encoder be represented by $G_e(\cdot)$, the manipulator by $G_m(\cdot)$, and the decoder by $G_d(\cdot)$. Likewise, denote the texture and shape representations output by the encoder as $\mathbf{V} = G_{e,texture}(\mathbf{X})$ and $\mathbf{M} = G_{e,shape}(\mathbf{X})$. Then a linear magnification performed by the manipulator would be expressed as

$$G_m(\mathbf{M_a}, \mathbf{M_b}, \alpha) = \mathbf{M_a} + \alpha(\mathbf{M_b} - \mathbf{M_a}) \tag{12}$$

where α is the magnification factor. However, Oh *et al.*'s network introduces some non-linearity in the magnification through two functions, $g(\cdot)$ and $h(\cdot)$, which represent a 3×3 convolution followed by ReLU, and a 3×3 convolution followed by a 3×3 residual block, respectively.

Thus the non-linear magnification performed by Oh *et al.*'s network is expressed as follows:

$$G_m(\mathbf{M_a}, \mathbf{M_b}, \alpha) = \mathbf{M_a} + h(\alpha \cdot g(\mathbf{M_b} - \mathbf{M_a})) \tag{13}$$

Considering the warping module and manipulator together as defining the new manipulator in our novel algorithm, the magnification function would be expressed as

$$\begin{aligned} &G_m(\mathbf{M_a}, \mathbf{M_b}, \alpha, H_{RA}, H_{BR}, \vec{d}) \\ &\quad = f(\mathbf{M_a}, H_{RA}, H_{BR}, \vec{d}) \\ &\qquad + h(\alpha \cdot g(\mathbf{M_b} - f(\mathbf{M_a}, H_{RA}, H_{BR}, \vec{d}))) \end{aligned} \tag{14}$$

with $f(\cdot)$ referring to the warping function defined by equation 11.

E. Keypoints in Testing Videos

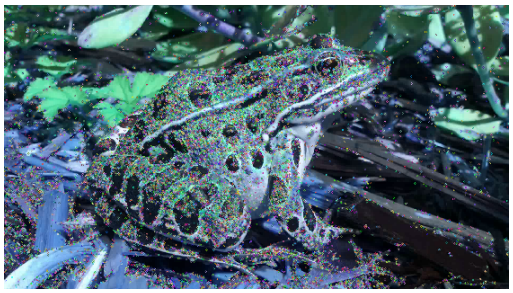

Figure 15. Frog Keypoints.

Figure 16. Purple Tulips Keypoints.

Figure 17. Red Tulip Keypoints.

F. Video Examples

F.1. Naïve Implementation Video Examples

We have included a folder of sample inputs and outputs that demonstrate our preliminary results compared to the baseline algorithm. Here is a listing of the videos and their features:

Name	Camera Movement	Magnification Factor
dishwasher	large	10
drilling_wood	none	10
plant	none	50
frog	small	10

F.2. Double Warp Implementation Video Examples

We have included a folder of sample inputs and outputs that demonstrate our double warp implementation results compared to the baseline algorithm. Here is a listing of the videos and their features:

Name	Camera Movement	Magnification Factor
dishwasher	large	5
tree	large	10
frog	small	10
purple_tulips	almost none	10

F.3. Learning-Based Implementation Video Examples

We have included a folder of sample inputs and outputs that show our results from the learning-based implementation. Here is a listing of the videos and their features:

Name	Camera Movement	Magnification Factor
red_tulip	small	10
frog	small	10

G. Naïve Video Alignment Code

```
def alignImages(img1,img2):
    # align img2 to img1
    # utilizes homography provided in opencv

def alignVideo(input_frames,output_frames):
  lastframe = None
  for nextframe in input_frames:
    if lastframe is not None:
      nextframe = alignImages(nextframe,lastframe)
    output_frames.append(nextframe)
    lastframe = nextframe
```

H. Random Homography Generation Code

```
def getPerspectiveTransform(shape, alpha, beta, gamma, dx, dy, dz, f):
    # convert to radians
    alpha = (alpha)*math.pi/180.
    beta = (beta)*math.pi/180.
    gamma = (gamma)*math.pi/180.
    # get width and height for ease of use in matrices
    w, h, channels = shape
    # Projection 2D -> 3D matrix
    A1 = np.matrix(
              [[1, 0, -w/2],
              [0, 1, -h/2],
              [0, 0,    0],
              [0, 0,    1]])
    # Rotation matrices around the X, Y, and Z axis
    RX = np.matrix([[
              1,          0,           0, 0],
              [0, math.cos(alpha), -math.sin(alpha), 0],
              [0, math.sin(alpha),  math.cos(alpha), 0],
              [0,          0,           0, 1]])
    RY = np.matrix([[
              math.cos(beta), 0, -math.sin(beta), 0],
              [0, 1,          0, 0],
              [math.sin(beta), 0,  math.cos(beta), 0],
              [0, 0,          0, 1]])
    RZ = np.matrix([[
              math.cos(gamma), -math.sin(gamma), 0, 0],
              [math.sin(gamma),  math.cos(gamma), 0, 0],
              [0,          0,           1, 0],
              [0,          0,           0, 1]])
    # Composed rotation matrix with (RX, RY, RZ)
    R = np.matmul(RX, np.matmul(RY, RZ))
    # Translation matrix
    T = np.matrix([[
             1, 0, 0, dx],
             [0, 1, 0, dy],
             [0, 0, 1, dz],
             [0, 0, 0, 1]])
    # 3D -> 2D matrix
    A2 = np.matrix([[
              f, 0, w/2, 0],
              [0, f, h/2, 0],
              [0, 0,   1, 0]])

    # Final transformation matrix
    trans = np.matmul(A2,np.matmul(T, np.matmul(R, A1)))

    return trans
```

Figure 18. Code for Generating a generic Homography transform.