ARL-TR-8776 ● SEP 2019

Motion Amplification in Video Using Optical Flow

by John S Hyatt, Samuel Edwards, and Michael S Lee

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8776 ● SEP 2019

Motion Amplification in Video Using Optical Flow

by John S Hyatt and Michael S Lee
Computational and Information Sciences Directorate, CCDC Army Research Laboratory

Samuel Edwards
Parsons, Inc.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2019	Technical Report	April–July 2019

4. TITLE AND SUBTITLE
Motion Amplification in Video Using Optical Flow

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
John S Hyatt, Samuel Edwards, and Michael S Lee

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
CCDC Army Research Laboratory
ATTN: FCDD-RLC-NB
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-8776

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID(s): John S Hyatt, 0000-0003-1265-9788; Michael S Lee, 0000-0002-0419-6069

14. ABSTRACT
This is a guide and demonstration of motion amplification in video using optical flow. The technique amplifies small motions that are invisible in the original video, rendering them visible to the naked eye. Restricting the amplification to specific frequency bands (e.g., those corresponding to peaks in the video's power spectral density) eliminates noise and isolates individual modes of vibration, precession, and so on.

15. SUBJECT TERMS
video analysis, optical flow, video signal processing, optical imaging, small motion amplification

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Michael S Lee
Unclassified	Unclassified	Unclassified	UU	26	19b. TELEPHONE NUMBER (Include area code) (410) 278-5888

Contents

List of Figures iv

Acknowledgments v

1. Introduction 1

2. Motion Amplification Process 1

2.1 Video Data 2

2.2 Calculating Optical Flow 2

2.3 PSD and Frequency Isolation 3

2.4 Video Warping 5

3. Conclusions 6

4. References 8

Appendix. Algorithm and Pseudocode 9

List of Symbols, Abbreviations, and Acronyms 18

Distribution List 19

List of Figures

Fig. 1 Original video of a) phone and b) engine 2

Fig. 2 PSDs obtained from the phone video, averaged over all pixels and all frames, and vertically scaled for clarity. The PSD of the pixel intensity is plotted in red and the PSDs of the angular and magnitude components of the optical flow in blue and green, respectively. Frequency bands selected for amplification are marked in gray. 3

Fig. 3 The total magnitude of the optical flow in the two frequency windows marked in gray in Fig. 2. The two PSD peaks correspond to different types of motion: a) low-frequency translation of the center of mass and b) high-frequency oscillation. Most of the high-frequency movement is localized to small regions of the object, while other regions are nearly stationary. 4

Fig. 4 Optical flow calculated from the video in Fig. 1a, after isolating a) low-frequency and b) high-frequency components from the PSD in Fig. 2. Hue represents the direction of motion and saturation the magnitude. Saturation has been increased for clarity. 5

Fig. 5 Optical flow calculated from the video in Fig. 1b, after isolating the low-frequency component of interest. Hue represents the direction of motion and saturation the magnitude. Saturation has been increased for clarity. 5

Fig. 6 Final amplification of the video in Fig. 1a, after isolating a) low-frequency and b) high-frequency components from the PSD in Fig. 2 6

Fig. 7 Final amplification of the video in Fig. 1b, after isolating the low-frequency component of interest 6

Acknowledgments

We would like to thank the US Army Combat Capabilities Development Command's Army Research Laboratory OSD Supercomputing Resource Center for providing computing hours for this work, and Brian Simmonds for assembling the video figures.

1. Introduction

This is a guide to and demonstration of motion amplification in video using optical flow. We calculate the optical flow between adjacent frames of video recorded of objects undergoing small-amplitude oscillation that is not visible to the naked eye. After multiplying the optical flow by an amplification factor we use it to warp the video, resulting in a large-amplitude representation of the original small-amplitude motion. Further, by analyzing the video's power spectral density (PSD), we can restrict the amplification to frequencies of interest, reducing noise and isolating individual modes.

High-quality motion amplification of video data has been performed previously using Eulerian approaches[1,2] and marketed commercially for the purpose of detecting flaws in structures and equipment.[3] These methods do not calculate the optical flow, which can be used to generate an arbitrarily large amplification through image warping. Similarly, optical flow calculations are well established, but applications are almost always restricted to large displacements, such as in object tracking.[4] Calculating the optical flow allows us to isolate components of the observed motion (e.g., magnitude vs. angular). It further opens up the possibility that, with a machine learning (ML) model that can extract optical flow from video data,[5] the process could be substantially accelerated. We have briefly explored ML-based optical flow for use in motion amplification without immediate success. This is perhaps because, as with conventional optical flow calculators, the available pre-trained ML models are targeted toward *large* displacements for use in object tracking. As a result, the video data they are trained on do not belong to the same distribution as our target data.

2. Motion Amplification Process

We have observed that optical flow can only be reliably calculated from uncompressed video, particularly in the case of small-amplitude motion, where compression artifacts erase the small differences between adjacent frames. Note that this does not necessitate high spatiotemporal resolution: the frame rate of the video must be at least twice the frequency of the motion to be amplified,[6] while the spatial resolution need only be high enough to distinguish the features of interest.

Pseudocode for our algorithm is presented in the Appendix.

2.1 Video Data

We include example results of motion amplification using two videos that we recorded from a Phantom Miro LAB 3a10 high-speed digital camera[7]:

- **Phone.** A smartphone with a piece of paper loosely taped to the top, laid flat on a surface and set to vibrate.
- **Engine.** The engine of a car (2012 Ford Focus) idling with the hood up.

The original videos are shown in Fig. 1. Note that they are virtually indistinguishable from static images.

a **b**

Fig. 1 Original video of a) phone and b) engine

2.2 Calculating Optical Flow

Optical flow is the apparent velocity of pixel intensity in a video, defined by

$$(\boldsymbol{\nabla} I) \cdot \mathbf{V} = -\partial_t I \ . \tag{1}$$

$I(x, y, t)$ is the intensity of a pixel at coordinates (x, y) and time t, and $\mathbf{V}(x, y, t) = V_x \hat{\mathbf{x}} + V_y \hat{\mathbf{y}}$ is the optical flow. This is an ill-posed problem as there are two unknowns, V_x and V_y. Different methods for calculating optical flow make different assumptions to constrain the equation; we use the Farnebäck method,[8] which efficiently computes the optical flow over the entire frame. Specifically, we use the pyramidal implementation calcOpticalFlowFarneback in OpenCV. Rather than only calculating the flow once, across the base image pair, it repeatedly downsamples the images, obtains the flow at the lowest resolution, and updates it with the residual flow obtained at each successively higher resolution.

The algorithm outputs the optical flow in the (x, y)-coordinate system, but it is more useful to visualize in the polar (r, θ)-system, where r is the magnitude and θ is the direction of the flow.

2.3 PSD and Frequency Isolation

The calculated optical flow often contains a substantial amount of noise in addition to any noise present in the original video and may also contain multiple distinct motions we want to amplify individually. To reduce noise and isolate specific frequency components, we calculate the PSD of the pixel intensity in the original video using Welch's method,[9] implemented in SciPy as signal.welch. We show this for the phone video, together with the PSDs of the magnitude and angle components of the calculated flow, in Fig. 2.

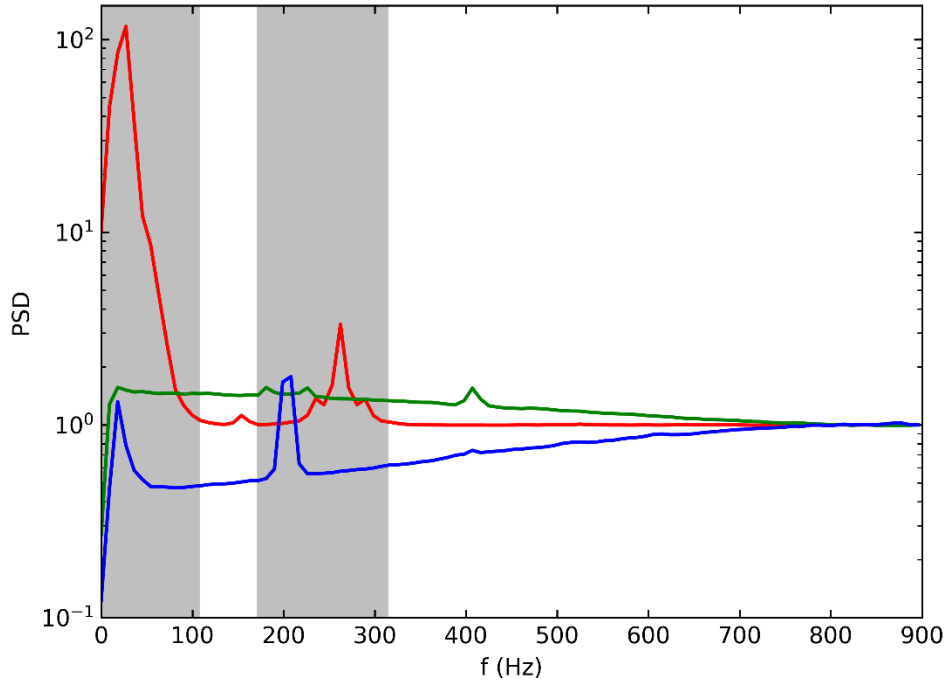


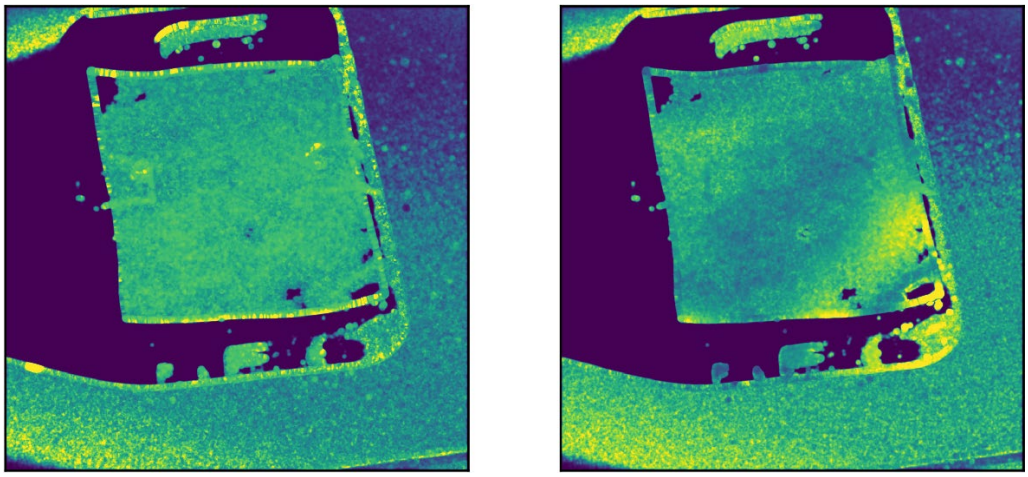
Fig. 2 PSDs obtained from the phone video, averaged over all pixels and all frames, and vertically scaled for clarity. The PSD of the pixel intensity is plotted in red and the PSDs of the angular and magnitude components of the optical flow in blue and green, respectively. Frequency bands selected for amplification are marked in gray.

The two peaks we observe in the intensity PSD are reproduced in the PSD of the optical flow's angular component, but not the magnitude component. This indicates that the corresponding features in the video are moving back and forth across the frame rather than brightening and darkening, as expected for a vibrating phone.

At high frequencies, the angular PSD is dominated by noise. To eliminate this and disentangle the motion at the peak frequencies, we apply simple band-pass filters

to the optical flow, corresponding to the frequency windows marked in gray in Fig. 2. Each resulting optical flow describes the motion within that frequency window.

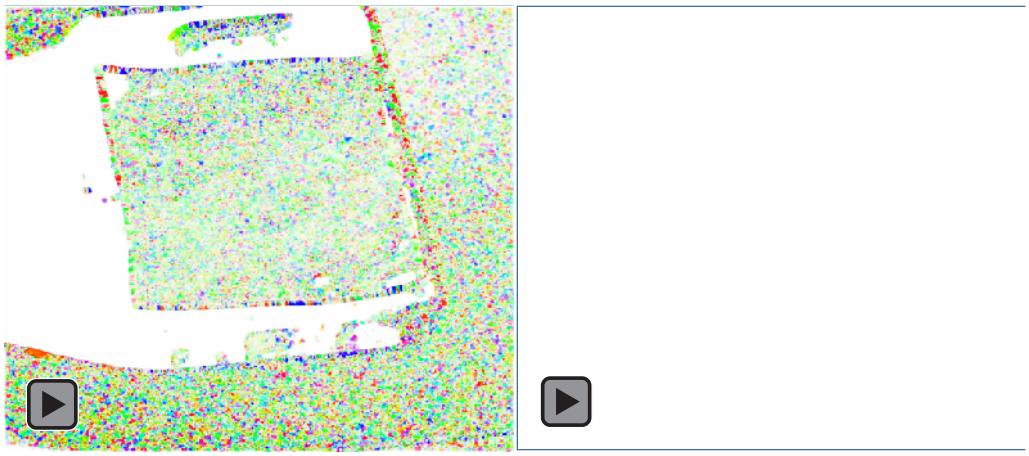
In Fig. 3, we plot the optical flow magnitude, averaged over all frames of the phone video, for both frequency windows. The low-frequency motion corresponds to the slow translational "walking" motion of the phone's center of mass as it vibrates, and the optical flow in this frequency window is distributed across the entire object. The high-frequency motion corresponds to the phone's stationary vibration, and is localized to the corners of the phone and the flexing of the paper taped to the screen. The combination of lighting conditions and the phone's matte black plastic case create large regions in the video without apparent flow. However, features such as the edges of the phone and its shadow, the sockets on the bottom of the phone, and the camera lens at the top of the phone give enough information to fully identify the object and its amplified motion. For reference, the unbalanced vibration motor for the Samsung S7 is in the upper-left corner.



a b

Fig. 3 The total magnitude of the optical flow in the two frequency windows marked in gray in Fig. 2. The two PSD peaks correspond to different types of motion: a) low-frequency translation of the center of mass and b) high-frequency oscillation. Most of the high-frequency movement is localized to small regions of the object, while other regions are nearly stationary.

The frequency-isolated optical flow videos are shown for the phone in Fig. 4 and the car engine in Fig. 5. PSD analysis of the car engine (not shown) identifies only one frequency window of interest, but performing a band-pass on that window reduces noise.



a b

Fig. 4 Optical flow calculated from the video in Fig. 1a, after isolating a) low-frequency and b) high-frequency components from the PSD in Fig. 2. Hue represents the direction of motion and saturation the magnitude. Saturation has been increased for clarity.

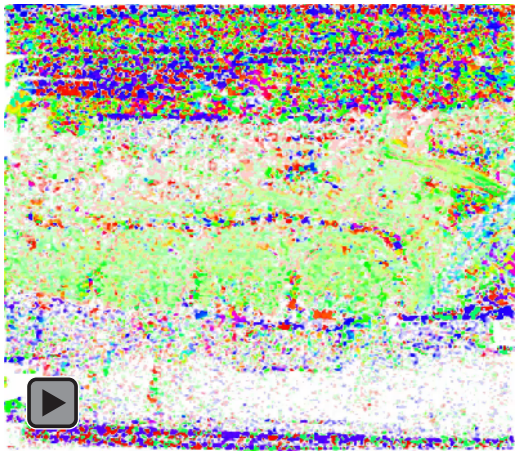
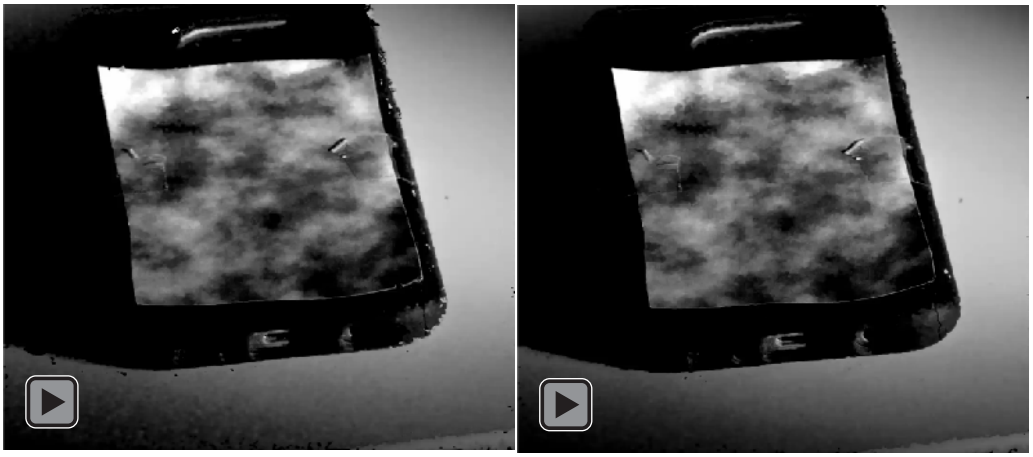


Fig. 5 Optical flow calculated from the video in Fig. 1b, after isolating the low-frequency component of interest. Hue represents the direction of motion and saturation the magnitude. Saturation has been increased for clarity.

2.4 Video Warping

We multiply the optical flow by an amplification factor and use that vector field to warp the original video with the OpenCV function remap. The calculated optical flow approximates the motion observed between two frames of video, while the amplification factor exaggerates this motion so that it is visible to the naked eye. The appropriate amplification factor must be determined on a case-by-case basis: too small and the motion is insufficiently amplified; too large and the objects in the video are warped beyond recognition. Additionally, the amplification factor must correct for the decrease in optical flow magnitude occasioned by the frequency filtering.

The final amplified video for the phone is shown in Fig. 6 and the car engine in Fig. 7. Both videos contain some distortions, caused by the large amplification factors used to ensure that the motions are clearly visible. These could likely be ameliorated by localizing the amplification factor, for example, by decreasing it in regions where the magnitude of the optical flow is large.



a b

Fig. 6 Final amplification of the video in Fig. 1a, after isolating a) low-frequency and b) high-frequency components from the PSD in Fig. 2

Fig. 7 Final amplification of the video in Fig. 1b, after isolating the low-frequency component of interest

3. Conclusions

We have demonstrated successful amplification of motion in video based on the explicit calculation of optical flow, a step not taken in previous implementations. While it is unclear whether this affords a computational speed up compared to Eulerian approaches in its current form, it is potentially amenable to advances in

fast ML-based optical flow algorithms. The amplification makes even small-amplitude motion, not apparent to the naked eye in the original video, clearly visible. We also isolate the contributions from different flow components (direction vs. magnitude) and frequency windows, which has the side effect of reducing noise.

4. References

1. Wadhwa N, Rubenstein M, Durand F, Freeman WT. Phase-based video motion processing. ACM Trans Graph. 2013;32(4). doi: 10.1145/2461912.2461966.

2. Kielkopf JF, Hay J, inventors; University of Louisville Research Foundation, Inc., assignee. System and method for precision measurement of position, motion and resonances. United States patent US 8,693,735. 2014 Apr 8.

3. RDI Technologies home page. Knoxville (TN): RDI Technologies; c2013–2019 [accessed 2019 June 17]. https://www.rditechnologies.com.

4. Agarwal A, Gupta S, Singh DK. Review of optical flow technique for moving object detection. Paper presented at: IC3I 2016. Proceedings of the 2nd International Conference on Contemporary Computing and Informatics; 2016 Dec 14–17; Noida, India.

5. Sun D, Yang X, Liu M, Kautz J. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. Paper presented at: CVPR 2018. 2018 IEEE Conference on Computer Vision and Pattern Recognition; 2018 June 18–22; Salt Lake City, UT.

6. Grenader U, editor. Probability and statistics: the Harald Cramér volume. Almqvist and Wiksell. In: Wiley publications in statistics. Hoboken (NJ): Wiley; 1959. 434 p.

7. Miro LAB 3a10. Wayne (NJ): Vision Research; c1992–2019 [accessed 2019 June 18]. https://www.phantomhighspeed.com/products/cameras/miromidsize/mirolab3a10.

8. Farnebäck G. Two-frame motion estimation based on polynomial expansion. In: Bigun J, Gustavsson T, editors. SCIA 2003. Image Analysis: 13th Scandinavian Conference; June 29 – July 2, 2003; Halmstad, Sweden. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 363 – 370.

9. Welch PD. The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Trans Audio Electroacoust. 1967;AU-15(2):70–73.

Appendix. Algorithm and Pseudocode

Our algorithm is as follows:

1) Create a mask for the parts of the video where no motion occurs.
2) Obtain the optical flow between frames of the video.
3) Locate peaks in the power spectral density (PSD) and isolate the desired frequency band in the optical flow.
4) Amplify the original video using this flow.

The pseudocode for these processes is shown below.

Process 1. Create a mask for the parts of the video where no motion occurs.

1. **Input:** A, w, m
2. $F \leftarrow$ number of frames in A
3. $N \leftarrow$ a number much larger than the maximum pixel intensity in A
4. **for** $i \in [0, F-1)$ **do**
5. $\quad f_i \leftarrow i$th frame in A
6. $\quad a_i \leftarrow |f_i - f_{i+1}| + N$
7. **end for**
8. $a \leftarrow \text{gmean}(a) - N$
9. $u \leftarrow$ number of window sizes in w
10. **for** $i \in [0, u)$ **do**
11. $\quad b_i \leftarrow \text{cv2.bilateralFilter}(a, w_i, \text{sigmaColor}, \text{sigmaSpace})$
12. **end for**
13. $c \leftarrow \text{gmean}(a, b)$
14. **for** pixel $\in c$ **do**
15. $\quad$ **if** pixel $\geq m$ **then**
16. $\quad\quad$ pixel $\leftarrow 1$
17. $\quad$ **else if** pixel $< m$ **then**
18. $\quad\quad$ pixel $\leftarrow 0$
19. $\quad$ **end if**
20. **end for**
21. **Return** c

Process 1 takes the video A, a list of bilateral filter window sizes w, and the mask cutoff value m as inputs. We use $w = [2,3,4, \ldots 20]$. It outputs a mask c that has a value of 1 wherever a pixel in the video shows measurable frame-by-frame differences (even though these may be invisible to the naked eye), and 0 elsewhere.

From line 4 to line 7, we calculate a_i, the absolute value of the difference between each pair of adjacent frames f_i and f_{i+1}. In line 8, we take the geometric mean of all a_i. N resolves numerical issues that occur in regions where no motion occurs: large areas that have effectively 0 difference between frames nevertheless can vary from pixel to pixel by many orders of magnitude due to high machine precision.

From line 9 to line 12, we create bilateral smoothings of a with several different window sizes, which are then averaged together with a in line 13. Out of all the smoothing methods we tried, this was the most effective in reducing noise while retaining the frame differences of small edge features.

From line 14 to line 20, we set all pixels in c with value above the threshold parameter m to 1, and all pixels with value below m to 0.

Process 2. Obtain optical flow between frames of the original video.

Input: A, c
$F \leftarrow$ number of frames in A
$F_{\text{skip}} \leftarrow 100$
for $i \in [0, F-1)$ **do**
    $f_i \leftarrow i$th frame in A
    if $i = 0$ **then**
        guess $\leftarrow 0$
    else if $i > 0$ **then**
        guess $\leftarrow v_{i-1}$
    end if
    $v_i \leftarrow \text{cv2.calcOpticalFlowFarneback}(f_i,$
        $f_{i+1},$
        flow = guess,
        pyr_scale,
        levels,
        winsize,
        iterations,
        poly_n,
        poly_sigma,
        flags = cv2.OPTFLOW_FARNEBACK_AUSSIAN)
end for
discard the first F_{skip} frames of v
for $i \in [0, F - F_{\text{skip}} - 1)$ **do**
    $(v_{i,r}, v_{i,\theta}) \leftarrow \text{cv2.cartToPolar}(v_{i,x},$
        $v_{i,y},$
        angleInDegrees = True)
end for
$v_{r,\text{masked}} \leftarrow v_r c$
$d \leftarrow$ list of nonzero pixel values in $v_{r,\text{masked}}$
$h \leftarrow$ histogram of d
$v_r^* \leftarrow$ cutoff value between the "main body" of h and the high-value tail
for pixel in v_r **do**
    if pixel value $> v_r^*$ **then**
        pixel value $\leftarrow v_r^*$
    end if
end for
Return (v_r, v_θ)

Process 2 takes the video A and the mask c obtained in Process 1 as inputs. It outputs the optical flow after removing spurious, extremely high pixel values and converting to polar coordinates (magnitude and direction of flow).

From line 4 to line 12, we calculate the optical flow between each pair of adjacent frames. Each calculation uses the previous frame pair's output flow as an initial guess, except for the first pair. We use $\text{pyr\_scale} = 0.5, \text{levels} = 5, \text{winsize} = 20, \text{iterations} = 10, \text{poly\_n} = 5,$ and $\text{poly\_sigma} = 1.1$. In line 13, we discard the first 100 optical flow frames to avoid transients resulting from the lack of a good initial guess for the first pair.

From line 14 to line 16, we convert the optical flow from Cartesian to polar coordinates.

In line 17, we mask the magnitude component of the optical flow. From line 18 to line 20, we histogram the nonzero values of the masked flow and find the cutoff value that separated the bulk of the histogram from a small number of spurious, extremely high pixel values. From line 21 to line 25, we cap the magnitude of the calculated (unmasked) optical flow at this cutoff value.

Process 3. Obtain the frequency windows of interest

1. **Input:** A, v_r, v_θ
2. $\text{PSD}_A \leftarrow \text{scipy.signal.welch}(A)$
3. $\text{PSD}_r \leftarrow \text{scipy.signal.welch}(v_r)$
4. $\text{PSD}_\theta \leftarrow \text{scipy.signal.welch}(v_\theta)$
5. **Return** $\text{PSD}_A, \text{PSD}_r, \text{PSD}_\theta$

Process 3 takes the video A and the r- and θ-components of the optical flow as inputs. It outputs the power spectral densities PSD_A (of pixel intensities), PSD_r (of optical flow magnitude), and PSD_θ (of optical flow direction).

Some of the calculated optical flow is noise, which appears as a plateau in the PSD. Features of interest appear as peaks in the PSD and allow us to determine the desired frequency range(s) over which to amplify the motion. The three PSDs each show different features depending on the specifics of the type of motion. For example, simple translation will not produce peaks in PSD_r, but an alternately brightening and dimming object would.

Process 4. Amplify the motion in the original video corresponding to a particular frequency range

1. **Input:** $A, v_r, v_\theta, \text{taps}, \text{freqs}, f_{\min}, f_{\max}, \text{amp}$
2. $F \leftarrow$ number of frames in A
3. $F_{\text{skip}} \leftarrow 100$
4. discard the first F_{skip} frames of A
5. discard the first and last $(\text{taps} - 1)/2$ frames of A
6. $F \leftarrow F - F_{\text{skip}} - (\text{taps} - 1)$
7. **for** $f \in \text{freqs}$ **do**
8. **if** $f \in [f_{\min}, f_{\max}]$ **then**
9. $g(f) \leftarrow 1$
10. **else then**
11. $g(f) \leftarrow 0$
12. **end if**
13. **end for**
14. $\text{bandpass} \leftarrow \text{scipy.signal.firwin2}(\text{taps}, \text{freqs}, g)$
15. $\text{bandpass} \leftarrow \text{bandpass}/\text{sum}(\text{bandpass})$
16. **for** $i \in [0, F + \text{taps} - 2)$ **do**
17. $(v_{i,r}, v_{i,\theta}) \leftarrow \text{cv2.polarToCart}(v_{i,r}, v_{i,\theta}, \text{angleInDegrees} = \text{True})$
18. **end for**
19. **for** $i \in [0, F - 1)$ **do**
20. $w_{i,x} \leftarrow (v_{i,x}, v_{i+1,x}, \dots, v_{i+\text{taps},x})$
21. $w_{i,y} \leftarrow (v_{i,y}, v_{i+1,y}, \dots, v_{i+\text{taps},y})$
22. $w_{i,x} \leftarrow \text{bandpass} * w_{i,x}$
23. $w_{i,y} \leftarrow \text{bandpass} * w_{i,y}$
24. $v_{i,x} \leftarrow \text{sum}(w_{i,x})$
25. $v_{i,y} \leftarrow \text{sum}(w_{i,y})$
26. **end for**
27. discard the last $(\text{taps} - 1)$ frames of v
28. **for** $i \in [0, F - 1)$ **do**
29. $B_i \leftarrow \text{cv2.remap}(A_i, -\text{amp} * (v_{i,x}, v_{i,y}), \text{None}, \text{cv2.INTER\_LINEAR})$
30. $(v_{i,r}, v_{i,\theta}) \leftarrow \text{cv2.cartToPolar}(v_{i,x}, v_{i,y}, \text{angleInDegrees} = \text{True})$
31. **end for**
32. $v_r \leftarrow \text{cv2.normalize}(v_r, \text{None}, \text{alpha} = 0,$

```
                                beta = 255,
                                norm_type = cv2.NORM_MINMAX)
33. v_θ ← cv2.normalize(v_θ,
                                None,
                                alpha = 0,
                                beta = 255,
                                norm_type = cv2.NORM_MINMAX)
34. for i ∈ [0, F − 1) do
35.     C = cv2.cvtColor((v_θ, v_r, 255),
                                cv2.COLOR_HSV2BR  )
36. end for
37. Return B, C
```

Process 4 takes the video A and the r- and θ-components of the optical flow as inputs, along with the number of taps in the band-pass filter; a list of frequencies between 0 and the Nyquist frequency (half the sampling rate); the minimum and maximum frequencies of the range of interest; and the factor by which to amplify motion in the original video. It outputs the magnified, frequency-filtered video B and the frequency-filtered optical flow C.

In line 4, we discard the first 100 frames of the video, as the first 100 frames of the optical flow were discarded in Process 2. In line 5, we discard frames from the beginning and end of the video corresponding to half the number of taps (the -1 accounts for the fact that we use an odd number of taps). This is necessary because the frequency filtering will "use up" a number of optical flow frames equal to the number of taps minus 1.

From line 7 to line 15, we construct a simple band-pass filter that accepts only frequencies in the desired range. In our experience, we found this to reliably produce better amplified videos than more sophisticated filter types.

From line 16 to line 18, we convert the optical flow from polar to Cartesian coordinates. Note that, because in Process 2 we capped the value of the flow magnitude, this is not the same as the Cartesian flow originally calculated in Process 2.

From line 19 to 26, we perform the band-pass. The w_i defined in lines 20 and 21 are collections of a number of adjacent frames equal to the number of taps. In lines 22 and 23, the band-pass filter is applied to each pixel across the entire set of frames. In lines 24 and 25, these sets of frames are summed to create a single, frequency-filtered frame. In line 27, we discard the optical flow frames at the end that contributed to earlier frames' filtering but were not themselves filtered.

In line 29, we warp one frame in the original video with the corresponding frame of the frequency-filtered optical flow, multiplied by the desired amplification factor. Using $\text{amp} = 1$ would essentially warp frame A_i to frame A_{i+1}. By increasing the amount of warping, we amplify the resulting motion to the point that it can be seen by the naked eye.

In line 30, we convert the Cartesian optical flow back to polar coordinates for the purposes of visualizing the flow over the course of the video. In lines 32 and 33, we normalize the magnitude and direction of the flow to lie between 0 and 255. From line 34 to line 36, we convert the optical flow from HSV to RGB, using hue to represent direction and saturation to represent magnitude.

List of Symbols, Abbreviations, and Acronyms

HSV	hue, saturation, value
ML	machine learning
PSD	power spectral density
RGB	red, green, blue

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DIR ARL FCDD RLD CL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
10 (PDF)	CCDC ARL FCDD RLC B HENZ T PHAM FCDD RLC E B MACCALL FCDD RLC ED R RANDALL FCDD RLC N B RIVERA FCDD RLC NB E CHIN J HYATT M LEE FCDD RLV P A HOOD C KWEON