Motion Detection and Segmentation Using Optical Flow

Andrew Chase, Bryce Sprecher, James Hoffmire

Abstract:

This paper discusses the detection, analysis and segmentation of motion in video using optical flow algorithms. Optical flow detects motion within each neighborhood of pixels by registering changes in the color and intensity of pixels from frame to frame. Vectors indicating the direction and magnitude of detected motion are created, and groups of similar vectors are then segmented from each other. The primary purpose of optical flow is to use segmentation to isolate individual moving objects within a video. This paper first introduces the fundamentals of optical flow and offers an overview of existing literature related to general motion detection. We next discuss the theory behind optical flow, followed by a description of our method and implementation in MATLAB. Finally, we share our results along with our takeaways about optical flows applications and limitations.

Introduction:

As the name may imply, motion segmentation divides a video into different regions based on detected motion. That is, neighborhoods of similar motion can be grouped together in a single layer, allowing for a dynamic scene to be broken into components of individual motion. The utility of this operation relies on the fact that objects in the real world tend to undergo smooth movements as coherent wholes. Thus, detecting these regions of similar motion allows us to detect moving objects in the scene.

There are many motion detection and segregation methods. Our approach relies on dense optical flow to detect and characterize the motion. We define motion as a detected change in intensity or color of a pixel and its surrounding neighborhood. This method requires several assumptions in order to work correctly, and even if these assumptions hold true this method may still generate noise and errors based on scene complexity. Our implementation is most effective at detecting and segmenting rigid motion. That is, any motion that preserves the size and shape of an object, allowing for rotations and translations. Our implementation does not correctly segment objects with non-rigid or differential motion. Such objects are not accurately characterized as a single region of motion and hence they are often segmented into multiple layers.

After detecting motion, we then analyze motion within the scene to define regions of interest across frames. The objects that undergo smooth motions tend to be of interest. Lastly we isolate these regions based on the analysis of the detected motion.

Motivation:

We were first introduced to the concept of motion analysis and manipulation in lecture, where we were shown working examples of motion magnification. This concept, albeit complex in implementation, had amazing results due to the extent and accuracy to which motion was amplified. The amount of information in a video that our eyes couldn't perceive fascinated us. One thing that many of these techniques relied on was segmenting the scene for further analysis. Though techniques like chroma keying can produce excellent results in isolating moving objects, they require very artificial scenes to produce good results. We were interested in the idea of isolating moving objects in more natural and arbitrary scenes and how we could extract such motion data for ourselves. From there, we began wondering how we might be able to apply motion detection to construct interesting and useful results.

Problem statement:

In our research we found that many sophisticated video based algorithms require data be separated into isolated regions of interest for further analysis. Our goal is to understand the fundamentals of optical flow and use it to detect motion from video. From this, we can experiment with different motion analysis techniques to segmentation motion into different layers. These layers will represent moving and non moving objects in the scene. We also wish to explore some applications of this information by creating our own masking and filtering effects based on the optical flow results.

Related work:

We began our topic research with motion magnification in mind. The papers read were from MIT CSAIL's Video Magnification group. State of the art phase based methods were discussed by Neal Wadhwa et. al. These phase based methods are far more robust to noise than their previous works and capable of magnifying motions to a greater extent than their earlier optical flow based methods. In Frédo Durand's "A World of Movement" they outlined possible applications for motion magnification including visual pulse detection methods, material structural properties in response to vibration, and sound recovery from silent video footage. We attempted to implement the first work on the topic as described by Ce Liu et. al in "Motion Magnification." Due to complexity and time constraints we were unable to implement their methods, but this paper did lead us to our topic of scene segmentation as it was a crucial step in their process.

We looked at the first work in scene segmentation in Wang, J.y.a., and E.h. Adelson's "Representing Moving Images with Layers." They define a video as a combination of layers containing individual moving objects which helped guide our approach to segmenting our video. Their motion analysis step clusters motion data and optimizing models to describe multiple moving object layers; this allows them to segment multiple moving objects. Manjunath Narayana, Allen Hanson, and Erik Learned-Miller introduce a more sophisticated probabilistic based analysis in their 2013 paper. This method addresses the issue of depth based segmentation caused by parallax; their results show that a probabilistic method can effectively pick the correct number of moving objects and isolate them regardless of the depths they span.

Theory:

Video motion data extraction techniques generally fall into two categories: dense and point-based computation. Dense optical flow methods attempt to calculate the magnitude and direction of motion vectors for each pixel neighborhood, and from the result can determine whether the neighborhood moved or not from frame to frame. Although this method can be computationally expensive, it allows for gradient maps to be made of the entire frame. The gradient data can be used to isolate regions of similar motion and create motion layers. Point-based optical flow detects feature points in each frame, and attempts to match them together. This technique is similar to SIFT point matching in panorama stitching. Once a feature point has been found in two adjacent frames, its change in position is used to determine the magnitude and direction of the motion. Because we were interested in creating layers based on motion, and because point-based methods have a lot of overhead in feature detection, we opted to implement a dense optical flow approach.

Optical flow algorithms detect any change in color or intensity as motion, and this introduces several potential sources of error. Specifically, camera movement and changes in lighting cause significant errors in motion detection. As a result, it is impossible for an algorithm to distinguish between genuine motion and noisy data without additional information and assumptions. To explain how each pixel in a frame's magnitude and direction can be determined, we must make an assumption called the optical flow constraint. In conceptual terms, this constraint assumes that a scene's overall illumination intensity is constant throughout the lifetime of the exposure. Under this assumption, any changes detected in a pixel neighborhood are in fact due to motion and are not the result of an object changing color or brightness. This assumption allows for all changes detected in the *x* and *y* directions of the frame to be correlated with the change in *t.* We can use a first-order Taylor polynomial to estimate this change, given in the form

$$I_t + vI_x + uI_y = 0$$

Equation 1: *Optical flow solution when the constraint is satisfied* where I_x, I_y, *and* I_t are the derivatives in the *x, y,* and *t* directions respectively. The solutions for *v* and *u* are the *x* and *y* components of the movement vector associated with the current pixel neighborhood. We can rewrite the above as

$$I_t = -|\vec{v}|\left(\cos(\phi)I_x + \sin(\phi)I_y\right)$$

Equation 2: *Optical flow with angle representation*

where the vector $\vec{v}$ is the combined components *v* and *u* as before, and ϕ is the angle of motion. Solving for either equation for every pixel neighborhood creates a vector field for the current frame. This data represents the detected motion and we can analyze this data to segment the frame.

Method:

We used a dense optical flow MATLAB toolbox that was developed by Stefan Karlsson and Josef Bigun. Within this toolbox we used an algorithm based on the Lucas and Kanade method. The toolbox also includes a simpler gradient-based implementation of dense optical flow which we did not use, but which helped give us a foundational understanding of optical flow. Although the package consists of forty-two MATLAB scripts, only five are used to calculate optical flow. The rest of the files perform illustrations of key steps or build a framework to process videos effectively in MATLAB. Using this framework, we were able to detect motion in each frame by its change in color and intensity in the *x*, *y*, and *t* directions.

The approach begins by calculating derivatives in the *x*, *y*, and *t* directions for each pixel of each frame, corresponding to the change in color and intensity detected. This was performed by using normalized kernel filters within MATLAB, because this was the simplest and most efficient solution. To improve results, these filters were dynamically convoluted to prevent boundary effects along the edges of the frame where pixels don't have complete neighborhoods of values. The implementation and commented details can be found in grad3D.m. Using these gradient values, we performed edge detection for each frame. Although normalized magnitude would have been sufficient in creating an edge image, the final implementation used recursive temporal integration to reduce noise in a way that doesn't consume excessive memory (i.e. without performing computation across previous and next frames).

The final step solves for *u* and *v* in the optical flow constraint. We cannot solve for individual pixels accurately, as that depends on having the detected gradients of the pixel showing clear motion in an observable direction; this is less likely than we first thought. The established method for determining the gradient quality of a neighborhood of pixels is through the use of 2D structure tensors, which are simply 2D matrices that represent the predominant direction of the gradients of a neighborhood of pixels (Figure X). Although complex algorithms may have dynamic neighborhoods that change size based on the 2D structure tensor, our toolbox opted for a static approach. The user sets the optical flow resolution before runtime. Thus, if a pixel is of poor quality or is surrounded by a poor neighborhood, the resulting vector could still be non-existent (under constrained problem) or highly incorrect (bad solutions). The effective solution we used was the Lucas and

Kanade (LK) method, which simply makes the neighborhoods larger, thus giving coarser optical flow resolution to overconstrain the problem. A least squared method, similar to past homeworks and lecture, rejects outliers and finds the solution the entire neighborhood most agrees on. This solution is then included in the final optical flow matrices and used in frame filtering manipulations, including masking, blurring, and pixel replacement (superposition).

$$S_{2D} = \begin{pmatrix} \iint I_x^2 \mathrm{d}\vec{x} & \iint I_x I_y \mathrm{d}\vec{x} \\ \iint I_x I_y \mathrm{d}\vec{x} & \iint I_y^2 \mathrm{d}\vec{x} \end{pmatrix} = \begin{pmatrix} m_{200} & m_{110} \\ m_{110} & m_{020} \end{pmatrix}$$

Equation 3: *2-Dimensional Structure Tensor*

Because this method is considered the naive approach, the solutions we found have great sources of error. Namely, even with a large neighborhood size, a pixel may reside in an area of either uniform color and intensity, or symmetric texture repetition, which limits the amount of information we receive from the gradients alone. This problem, called the *Aperture Problem* (Figure 1), is a common issue with most motion-detection algorithms, as we are trying to predict motions of a large group of pixels simply from the information from a neighborhood. Thus, we also have issues when dealing with non-rigid motions within these neighborhoods. Optimizations are made in the toolbox to help compensate for these issues, such as using local regularization, vectorized coding (for parallelism), and more temporal integration. However, these methods go beyond the scope of this paper (and our mathematical knowledge). In conclusion, these issues still persist, but we are able to circumvent them by selecting optimal test samples to record, as shown below.

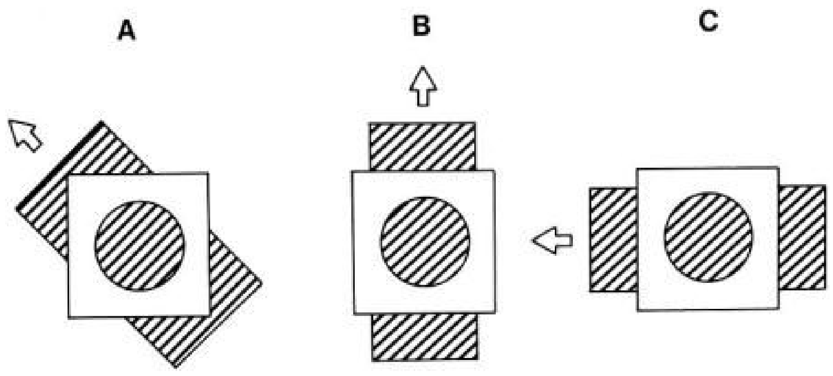

Figure 1: *Aperture Problem*
Regardless of the three motion directions, the lines in the window will appear to move in the same direction

Experimental results:

Our best results came from samples that conformed to our constraints of constant illumination and a non-moving camera. To compensate for these issues, low noise environments and proper exposure were essential for minimizing errors. Violations of these constraints and their consequences highlight the importance of these assumptions.

The constant illumination assumption failed when using high speed photography under incandescent and halogen lighting. At framerates between 400 and 1000 frames per second, the pulsing of fluorescent lighting is visible and causes variable illumination across frames. This causes motion analysis to identify the entire scene as moving, and the resulting mask is constructed with excessive error.

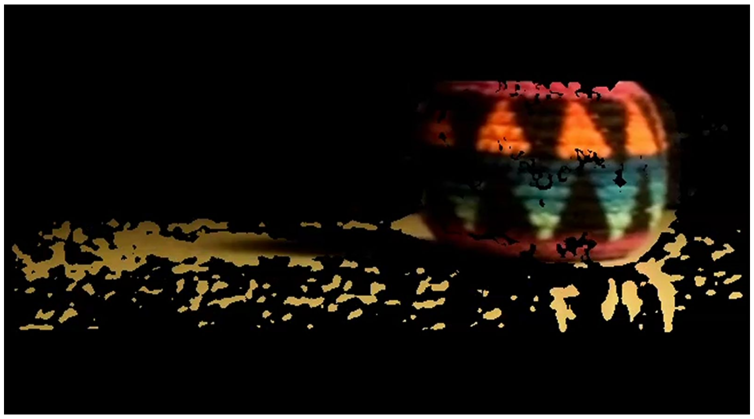

Figure 2: *Illumination Error*
All static pixels are supposed to be masked out, though due to change in illumination the algorithm mistakens table pixels as moving, and doesn't mask them out

Camera autofocus unexpectedly violated the static camera assumption in many of the dynamic scenes we recorded. The expected change in the amount of blur was detected as motion, but our camera also linearly shifted the frame while focusing, causing jumps in the image which were detected as motion. This resulted in a motion mask that shows the whole scene during the focal jumps. The detected motion vectors and the resulting mask can be seen in Figure 3.

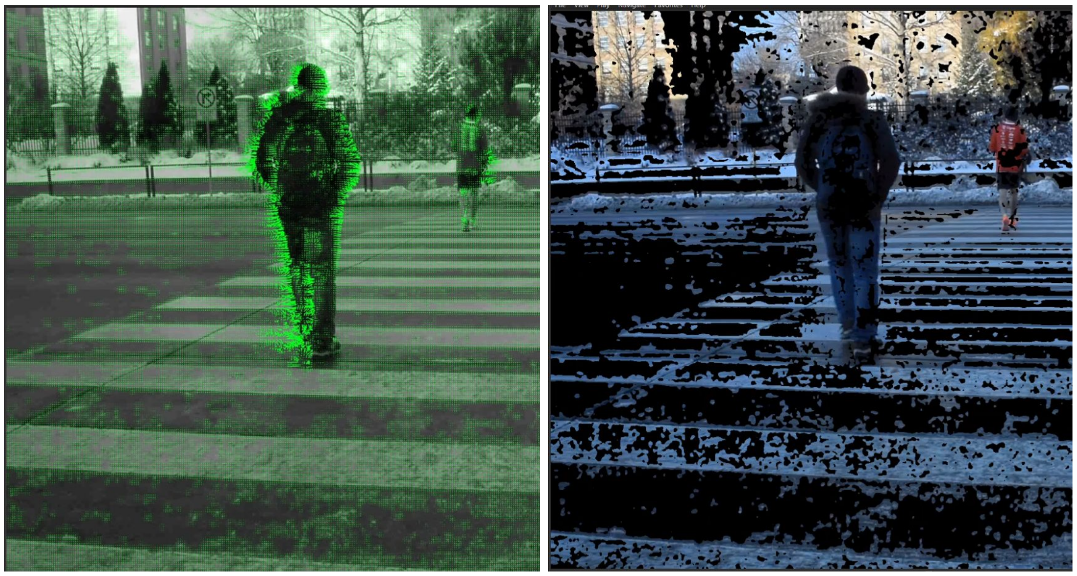

Figure 3: *Optical flow and corresponding mask errors due to autofocus*

A final an unavoidable failure in motion detection was due to areas of constant intensity on the interior of a moving object. For example in Figure 4; the overexposed areas of the flame have no defining texture or edges and cannot be detected as moving due to the aperture problem.

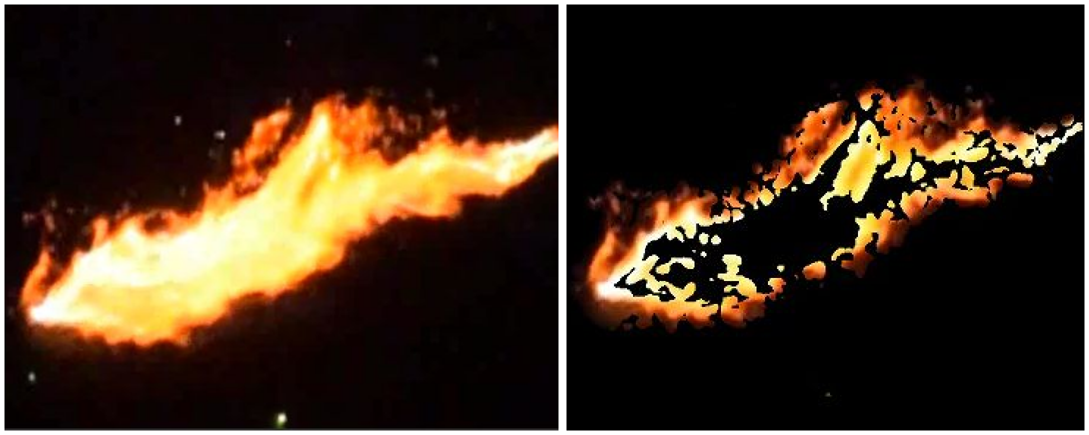

Figure 4: *Regions of constant intensity cannot be detected as moving*

When these sources of error are minimized we can produce very compelling results with our segmentation. With a mask that separates the foreground and background, we can use use filters for creative effect. An artificial depth of field decrease can be generated by blurring the background and sharpening the foreground. Another visually compelling process is to use color coded direction information from the flow vectors to colorize moving objects. This draws attention to moving objects and helps distinguish between them via color as seen in Figure 5.

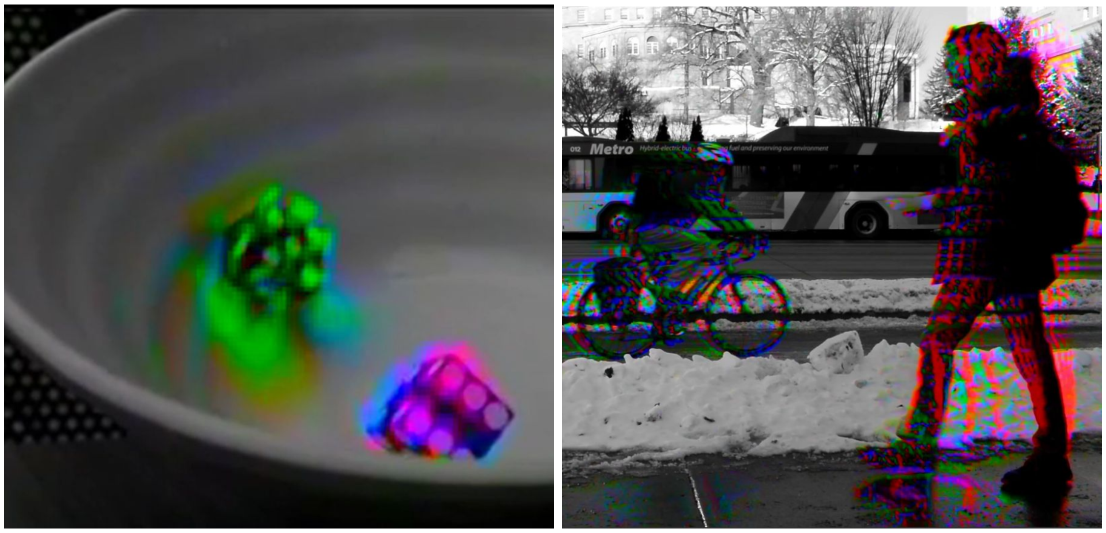

Figure 5: *Regions of similar motion are colorized together*

We were also able to perform image compositing both inside and outside MATLAB. For internal compositing we can layer a foreground moving object onto a different moving background or a static image as in Figure 6.

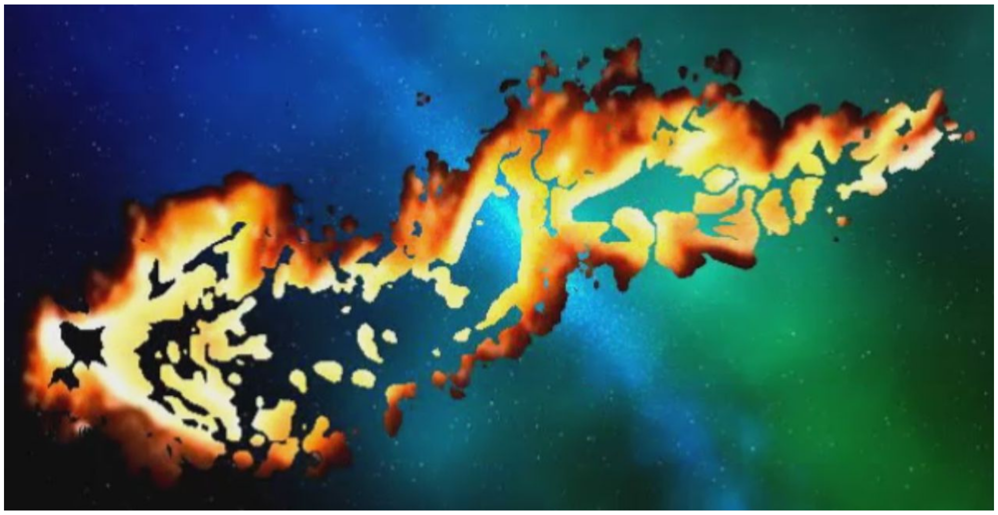

Figure 6: *Flame video composited on static background done in MATLAB*

For external compositing we write a green colored background to the isolated motion footage to act as an procedurally generated green screen. This footage is used with the chroma keying functionality of a dedicated video editor such as Blackmagic's DaVinci Resolve, giving the results in Figure 7. This technique allows for compositing of footage that would traditionally be impossible to acquire due to the need for a green screen.

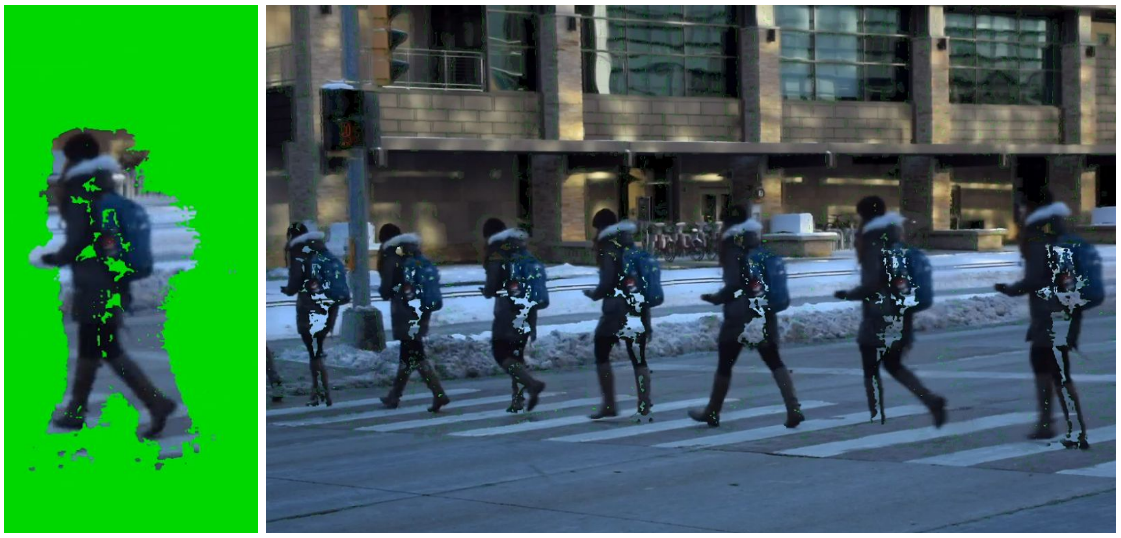

Figure 7: *Generated chroma frame and result from compositing in DaVinci Resolve*

Concluding remarks:

Video analysis presented some unique challenges, but it's rewards can be both visually compelling and useful. Temporal information provides a wealth of possibilities for applications that could not normally be extracted from static images. We initially planned to implement motion magnification, but after much reading and attempted implementation, we found it to be too complex for our time constraints. As we looked at other video based computational photography techniques, we found that many of them required the scene to be separated into different regions for further analysis. We decided that this common task was something worth understanding since so many other techniques rely on it. In the future, we would like to employ a more sophisticated motion analysis step. A technique like K-means clustering could allow us to be more resistant to camera movements, as this isn't based on threshold values but groups of relative optical flow values. Going forward, video will be captured with autofocus disabled to improve the quality of the motion masks. Though it wasn't our first topic, implementing scene segmentation has allowed us to understand one approach to an important initial step in many sophisticated video processing techniques.

The video results are available on our group's website, located here.

References:

Frédo Durand, William T. Freeman, Michael Rubinstein "A World of Movement" *Scientific American*, Volume 312, Number 1, January 2015

Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, William T. Freeman "Eulerian Video Magnification for Revealing Subtle Changes in the World" *ACM Transactions on Graphics*, Volume 31, Number 4 (Proc. SIGGRAPH), 2012

Liu, Ce, Antonio Torralba, William T. Freeman, Frédo Durand, and Edward H. Adelson. "Motion Magnification." *ACM SIGGRAPH 2005 Papers on - SIGGRAPH '05* (2005)

Neal Wadhwa, Michael Rubinstein, Frédo Durand, William T. Freeman "Phase-based Video Motion Processing" *ACM Transactions on Graphics*, Volume 32, Number 4 (Proc. SIGGRAPH), 2013

Narayana, Manjunath, Allen Hanson, and Erik Learned-Miller. "Coherent Motion Segmentation in Moving Camera Videos Using Optical Flow Orientations." *2013 IEEE International Conference on Computer Vision* (2013).

Wang, J.y.a., and E.h. Adelson. "Representing Moving Images with Layers." *IEEE Transactions on Image Processing* 3.5 (1994): 625-38.

Outside Code: We used a MATLAB toolbox/tutorial created by Stefan Karlsson and Josef Bigun, which was freely distributed through MATLAB's code distribution site MathWorks. All code was under license, and was free to use, modify, and redistribute with the included license. Although the toolbox consisted of 40+ MATLAB files, less than 10 were used for optical flow calculation and optimization, with most other files used to build a video-processing framework in MATLAB and for tutorials on the fundamentals of optical flow. All files of the toolbox were included in the code submission.

Code Written: We wrote 16 files of our own, all in MATLAB which either extended the framework to allow us to use the toolbox more effectively, or to apply our own custom per-frame effects through filter functions. Overall, around 300 lines of code without comments were written by all three group members. See the README in our code submission for the comprehensive list of files we created, and authorship for each specific file is specified at the top of the corresponding MATLAB script.

Individual Contributions:

Andrew Chase examined the toolbox to understand how it works both in theory and application. He created the framework that allowed us to process videos more easily by abstracting the toolbox specifics, and allowing custom filtering functions to be injected into the optical flow process to allow for easy change of effects. He also wrote scripts to display the original video and final results in a MATLAB figure, as well as wrote a few filtering effects to experiment with the optical flow data, including creating a general masking script based on the flow data to which a majority of other functions rely on. He also wrote the README for the program and cleaned up the code base to make it more understandable and efficient.

Bryce Sprecher researched possible papers and read 6 in preparation for project. Recorded and processed test footage for motion magnification. He also recorded all high definition outdoor footage for motion segmentation videos, as well as wrote optical flow color superposition scripts and processed videos. In addition, Bryce wrote a pseudo-green screen script, and used output for compositing in external video editor. He also contributed by optimizing the mask creation script so HD video could be processed effectively and with that processed and formatted videos to be used in the class presentation. He also built the website for the project.

James Hoffmire collaborated with Bryce in creating a filter that sharpens the foreground while blurring the background of a video. He helped to collect objects that demonstrate various types of motion and assisting in filming those objects in slow motion. He also contributed by editing and revising this final write up.