Bayesian Optical Flow

Nicholas Bach
nbach@wustl.edu

Abstract

When implementing optical flow in circumstances with high uncertainty, a Bayesian algorithm has the advantage of retaining the uncertainties after it's flow calculation. Comparing a Bayesian Horn-Schunck algorithm to a frequentist Lucas-Kanade algorithm shows that the Bayesian algorithm has a lot of promise, and performs better in certain situations.

1 Definitions and Terms

Optical Flow: The computer vision problem of tracking an object in a video or between two frames.
Sparse Optical Flow: Optical flow over a subset of pixel of an image. For example, if I wanted to track a person crossing a street but ignore any other movement from cars or other people.
Dense Optical Flow: Optical flow over every pixel in a frame.

2 Introduction

Optical flow is a very simple concept, but it is also a very powerful one. The idea of optical flow is old, as far as computer visions problems go, with its introduction in 1981.[1] Berthold Horn and Brian Schunck laid the groundwork for optical flow, and created the Horn-Schunck method in this paper, and yet optical flow is still being researched and optimized today. This is because the useful applications of optical flow have expanded incredibly as robot, drone, and video technology has developed. While the primary goal of optical flow is the tracking of objects, it can be used for the differentiating and 3-D mapping of objects as well.

3 Background

3.1 Master's Thesis

3-D mapping is how I have been using optical flow in relation to my master's thesis. In Dr. Silva's lab I have been using optical flow in order to track specific anatomical features during the microscopic video of a cochlear implant surgery. Through tracking these features, Dr. Silva and I hope to compute a three-dimensional mapping of the structures of the ear, and then combine those with a CT scan in order to compute the three-dimensional mapping of the cochlea itself. The cochlea would then be displayed in real time on the microscopic view in order to aid the surgeons during the surgery. This would be useful to doctors because cochlear implant surgeries are difficult and delicate. They require a surgeon to insert an electrode into the cochlea and coil it through the cochlea's inner structures, but the electrode can't scrape along the inside of the cochlea or it might cause damage. While doing this surgery a surgeon will feel constant pressure on the electrode, but they need to know when that pressure is too much so they can stop and try a different angle. And keep in mind, this surgery requires the surgeon to insert an electrode about 1 mm in diameter, so any differences in pressure are very slight. These factors make up a surgery that takes a lot of practice to do well and is very difficult for novice surgeons. To show this in his grant proposal, Dr. Silva conducted an experiment where he had surgeons with varying degrees of experience perform a cochlear implant surgery on an ear in a force transducer.1 This experiment showed that novice surgeons took longer to perform the surgery, applied more force over the course of the surgery, and applied more maximum force.

3.2 Basis for Bayesian Optical Flow

When Dr. Silva first came up with the idea to do this project he needed a way to track three anatomical structures: the incus, round window, and posterior ear canal wall. A master's student in the lab last year decided to use sparse optical flow trackers, because she wanted the location of the specific structures. She used seven tracking algorithms available through openCV, and the best of those had an average error of 41 pixels, which is very high. These algorithms were unable to track the structures well enough to be useful. When I joined the project at the beginning of this year, I started redoing her work to find where she went wrong. What I learned was that she hadn't gone wrong anywhere, the classifiers were just unable to track the structures due to the videos not having much color variation. Calculating the flow of an object requires finding that object pixels in one frame and

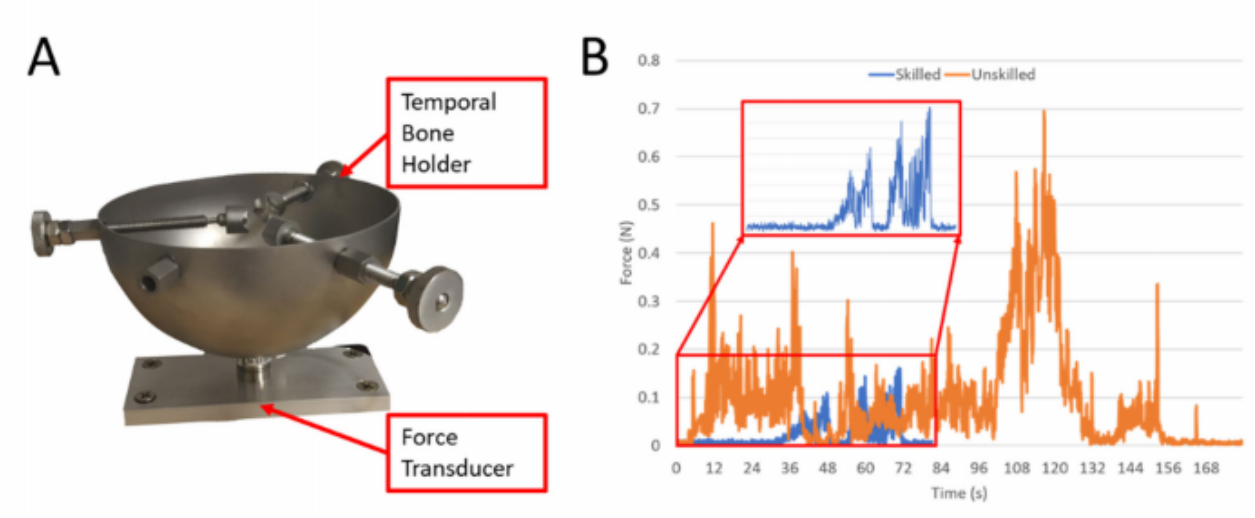

Figure 1: Measuring trauma during implant procedures by tracking force. A) A force transducer was coupled to a standard temporal bone holder to measure the force experienced by a cadaveric temporal bone during an implant insertion. B) A novice performer and an experienced physician each attempted the procedure, and forces experienced by the temporal bone were recorded. Inset shows magnified view of the physician attempt.

another, but when that objects pixel's colors are similar to their background it makes it hard to track them. If a white note card was moving in front of a white piece of paper, it would be very difficulty to track. This issue was confounded by the fact that the structures are small, and so the algorithms were trying to track only a few pixels. More information in computer vision almost always leads to more learning. In trying to track the structures alone, the algorithms had too small of a viewing window, and lost track of the structures easily. With this in mind, and utilizing the fact that the ear is one big structure and all parts of it move together, I started using dense optical flow algorithms instead, namely Lucas-Kanade. This allowed me to get more flow vectors, and I hoped that by averaging these flow vectors I would obtain the global flow I wanted. Unfortunately this wasn't the case. It turns out that the global optical flow algorithm was able to track some pixels well, but other pixels very poorly, again because a lot of the pixels were colored similarly. My average flow was impacted too much by noise, so I had to use RANSAC to attempt to get rid of it. RANSAC randomly samples points many times and then uses the grouping with the least error to form its prediction, which is useful when there is large noise in distance but not in quantity. This algorithm seemed to work well, but there isn't any guarantee that the noise will behave in a way that RANSAC can handle. Theoretically, if enough of the pixels were noisy then RANSAC might get high errors for every random sample it draws, resulting in the algorithm finding the flow essentially at random. Bayesian methods could fix this, though, because we retain the flow as a posterior, and could therefore calculate the uncertainty associated with each pixel. This would allow us avoid RANSAC all together, because the noisy pixels would be known once the algorithm ran, and they could be ignored instead of factored out after the fact.

3.3 Lucas-Kanade vs Horn-Schunck

In earlier sections I discussed how Horn-Schunck was the first optical flow algorithm to be created and how I decided to use Lucas-Kanade, but I didn't outline how those algorithms work. The difference in them, however, is very important when it comes to the Bayesian implementation. Horn-Schunck works by assuming smoothness in flow over the image, and formatting the flow as a global energy equation with a regularizer. This regularizer is how the equation enforces smoothness - without it two pixels right next to each other could move to opposite sides of the screen, which is very unlikely. Lucas-Kanade, on the other hand, assumes small movements. Using this assumption it can utilize windows around frames through convolutions, and looks for flow only within that small frame. This assumption won't be true in every case, but when it is true Lucas-Kanade is generally the best algorithm to use as it is very accurate and very fast. This comes into play when I discuss the Bayesian algorithm, because it utilizes a Bayesian form of the Horn-Schunck algorithm, whereas I use the frequentist Lucas-Kanade algorithm in comparison.

4 Proposed Approach

4.1 Bayesian Algorithm

For my algorithm, I followed along with the work and code from this paper by Jie Sun et al.[2] In this paper, Sun et al. discusses the Bayesian form of optical flow. The algorithm works by obtaining the two posteriors through the statistical inversion of the Horn-Schunk algorithm, formulated as a least squares problem with Tikhonov regularization. It generates two posteriors because, like the frequentist version, flow is calculated in the x direction and the y direction. As Sun et al. discusses, this poses a weird problem where pixels can have different uncertainties in different directions and calculating their overall uncertainty is unclear. I decided to simply sum the variances in the respective directions to get a pixel's overall uncertainty. Once the algorithm has calculated a form of the posteriors, it draws from them through a Markov chain Monte Carlo. As stated in section 3.3, this algorithm utilizes a Bayesian form of the Horn-Schunk algorithm instead of the Lucas-Kanade algorithm. This is not ideal, but I couln't find and implementations of a Bayesian Lucas-Kanade algorithm. I still chose to compare this algorithm to Lucas-Kanade, because that is what I am using in my lab and I wanted to compare it to my best standard I had access to, but the comparison comes with the caveat that I would expect a Bayesian Luacas-Kanade algorithm to outperform this one.

5 Experimental Results

5.1 Ideal test and Qualitative Analysis

I originally wanted to run this Bayesian Algorithm on an entire video from my lab work, but that quickly proved impossible. I started with the test frames provided, two 20x20 pixel images, and ran the algorithm. Everything worked great, and the algorithm took about 20 seconds. I then ran it on two frames I used in a computer vision class for a homework problem on Lucas-Kanade. These frames were 194x292 pixels large, and took about 15 minutes to be processed by the algorithm. Assuming the algorithm was liner in pixel size, I then ran it on two frames of my video, which are originally 1280x720 pixels but I scaled them down by a factor of nine to 426x240 pixels and expected it to take about 30 minutes. It instead took three hours, showing that it clearly does not scale linearly. Unfortunately I didn't have a ground truth for the frames in the three hour run time so it was basically useless, but I did want to show the results of the second experiment in figure2 for some qualitative analysis.

I think this is a good image to show, even though I don't have a quantitative way to compare the estimate to the ground truth, because it shows a lot of promise for the usefulness of the Bayesian algorithm in my situation. Even though some of the arrows are very inaccurate, the uncertainty graph importantly shows low uncertainty at the borders of the objects. For reference, the first of the two frames used is shown in figure2 below. For the most part, the endpoint error is low along the borders as well. There are troubling areas, such as the curtains to the right or the triangular object at the bottom where uncertainty was low and error was high, but for the most part the algorithm shows potential. I would want my algorithm to be good at finding the flow along the coloration borders in my cochlear implant surgery videos. Even if the rest of the regions were uncertain, that would be fine because my videos have so many pixels that utilizing only the borders would give more than enough information to calculate the entire flow.

5.2 Adaptation and Quantitative Analysis

The run-times from section 5.1 made me realize I would need to scale down the images by a lot. I looked at the 20 x 20 pixel images that were used as examples to the code, and realized that these were generated by creating the first frame and then moving its pixels counterclockwise to generate the second image. This has the huge advantage over the cochlear implant videos of having a guaranteed accurate ground truth. I had created the ground truths for the cochlear implant surgeries by tracking the structures by hand over hundreds of frames, which is inherently inaccurate. So, I decided to scale up the generated images to 120 x120 pixels in order to get more data points, and then run my tests on those. I could now compare the Bayesian algorithm and the Lucas-Kanade algorithm directly to a 100% accurate ground truth. An example output of my code is shown below.4

One drawback was that I now did not have a simple way of calculating the projective transform between frames, because while the cochlear implant videos only had translation, these generated frames introduced rotation, which adds many difficulties when converting it to a global flow. Without the global flow, however, I did not have the ability to implement RANSAC, so I ended up using all pixels of the Lucas-Kanade algorithm. Though this isn't perfect, it really exemplifies the issue with using only maximum likelihood estimators in calculations where the maximum likelihood might not be much higher than the likelihoods around it. For the Bayesian algorithm, I decided to include only the pixels with uncertainty above a threshold. Since the value of the uncertainty only made sense when comparing it to other uncertainties, I decided to use the mean uncertainty as the threshold and only keep pixel flows above the mean. I started by comparing the two algorithms as I increases the magnitude that each pixel

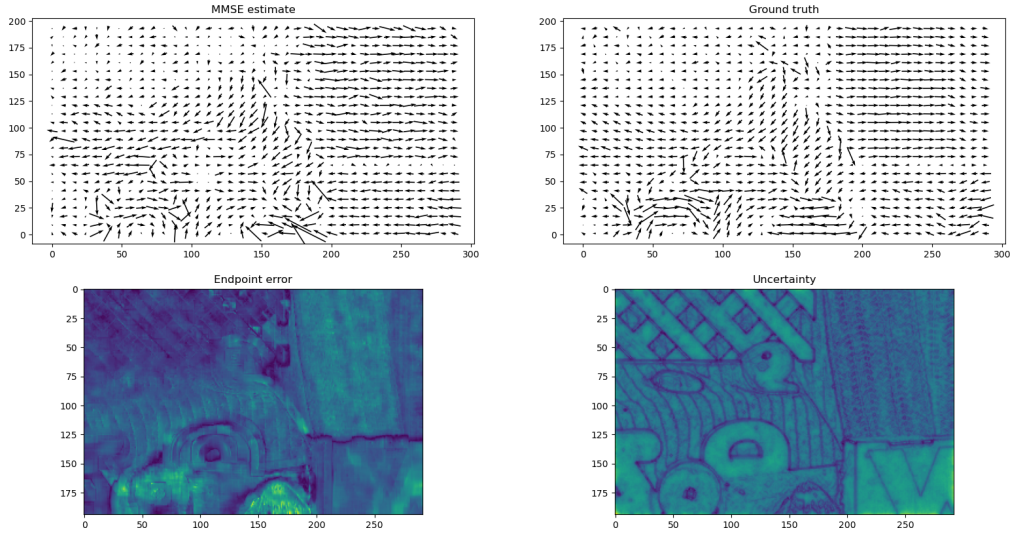

Figure 2: Four graphs showing the estimated flow calculated with the Bayesian algorithm (MMSE estimate), the ground truth flow, the error of each pixel, and the uncertainty of each pixel (brighter yellow correlate with higher errors/uncertainties and darker blues correlate with lower errors/uncertainties).

Figure 3: The first frame used in the flow calculations of figure 2 above.

moved between frames. Note: for all subsequent charts, error is calculated in units of number of pixels and was obtained with the L2-norm.

Magnitude	Lucas-Kanade	Bayesian
1	0.072	0.005
2	0.175	0.010
3	0.304	0.013
4	0.479	0.017
5	0.696	0.024
6	0.957	0.028
7	1.259	0.033
8	1.599	0.045
9	1.97	0.048
10	2.383	0.054

The above chart shows that as the distance a pixel moves between frames increases, the error in the Lucas-Kanade flow increases much more drastically than the error in the Bayesian algorithm. This illustrates the difference between the Lucas-Kanade and Horn-Schunck algorithms discussed in section 3.3. It is important to note that Lucas-Kanade uses a parameter called window_size to decide how large the windows are that it analyzes for each pixel. I kept the window_size at 11 for this experiment because that is the window size I use in on the cochlear implant videos, but if I had increased the window size between iterations the algorithm would have been much better at calculating flow, especially for the higher magnitudes.

I then wanted to test how pixel intensities would affect the algorithms, because this is the crucial variable when it comes to the algorithms effectiveness in the cochlear implant videos. To do this, I scaled the intensities of all pixel values down by a certain percent. For instance, at 80% a pixel with intensity 1 would be converted to a pixel with intensity 0.8.

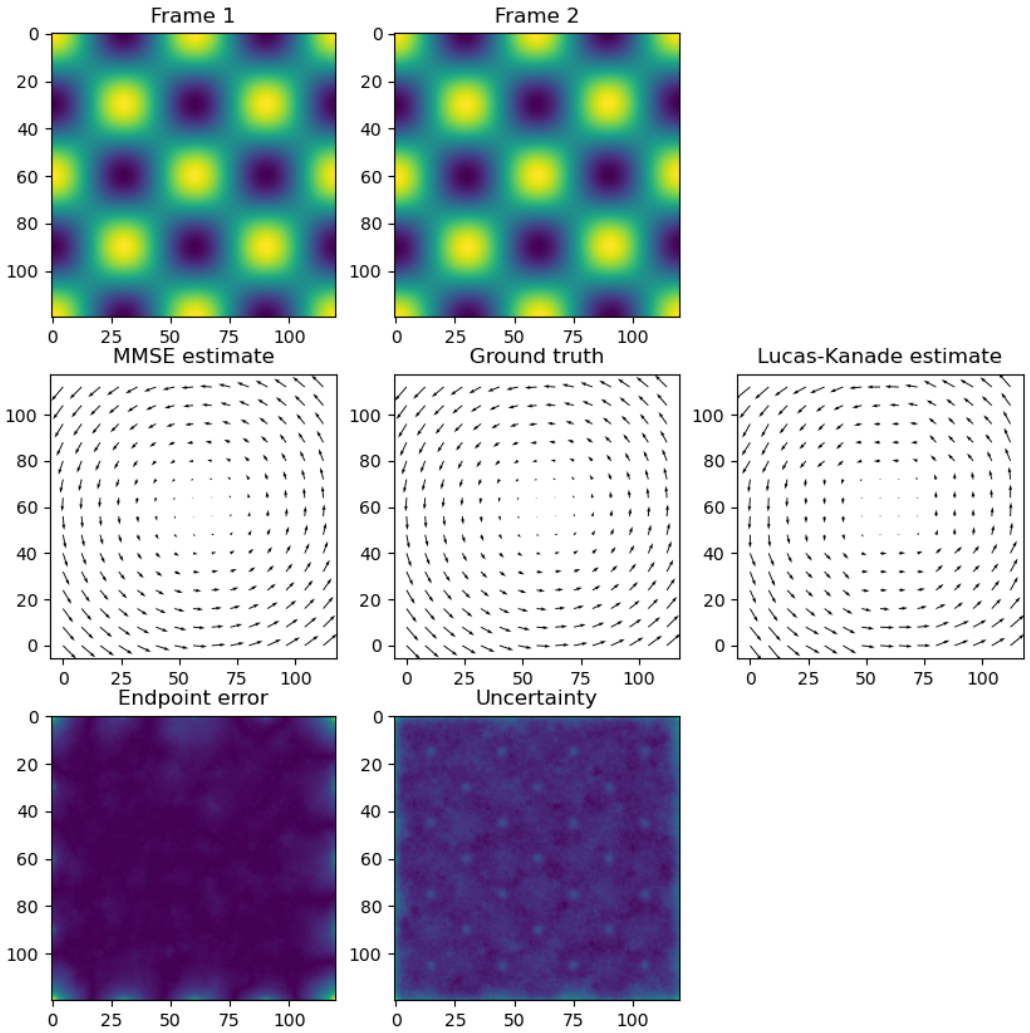

Figure 4: A very simple rotation of 1 pixel between two frames. Both algorithms are very accurate because of the small movement and high differentiation of pixel intensities.

Scale	Lucas-Kanade	Bayesian
100%	0.072	0.005
80%	0.072	0.011
60%	0.073	0.015
40%	0.074	0.014
20%	0.076	0.032
10%	0.078	0.033
1%	0.077	0.529

The above chart shows that as I scaled the pixels, Lucas-Kanade was essentially unaffected while the Bayesian Algorithm performed worse. This makes sense for Lucas-Kanade, since it is only looking for flow in small windows. As long as the relative change between pixels in those windows is the same, Lucas-Kanade will give the same output despite the lower scale. The Bayesian algorithm doesn't have this property, but still performs pretty well at 1% scale, especially considering the figure 6 in

Appendix A below. So many pixels have a very high uncertainty, but the Bayesian algorithm uses the ones with low certainty to great effect and maintains an error value of less than a pixel. If I redid this experiment, I would have tried some image preprocessing techniques such as adaptive thresholding in order to make the shapes borders more apparent to the Bayesian algorithm. I also would have added a small amount of noise into this experiment, in order to make the results realistic for real world applications and to make the problem different every iteration for the Lucas-Kanade algorithm.
Finally, I wanted to run some tests with random noise added. To do this, I changed random pixel values, increasing the number of pixels changed each iteration.

Percent Pixels Changed	Lucas-Kanade	Bayesian
0%	0.072	0.005
0.5%	0.132	0.090
1%	0.213	0.164
5%	0.363	0.468
20%	0.661	0.740
50%	0.791	0.780

This experiment surprised me by far the most. Based on the output shown in Appendix 6, I expected error to be higher. I think this puts into context what the error values mean. Sub-pixel error may seem very small, but the distances being estimated are very small themselves. Since each pixel in the original image only moves one pixel between frames, an error value of .8 means the pixels only moved about 20% of the way towards their correct location. This paints the above two graphs in a new light, because I had previously thought an error of 0.5 wasn't so bad for the Bayesian Algorithm, but it is actually quite bad. Another interesting aspect of this experiment is that the algorithms scale very similarly with large noise, but the Bayesian algorithm handles small noise much better. I have included three of the outputs from this experiment because I found it the most interesting. The 50% noise is figure 9 is essentially random, and the 5% noise figure 8 is only slightly better, but the 0.5% figure 7 is beautiful. There is high endpoint error in those specific noisy pixels and the uncertainty chart shows little points of high uncertainty as well. There are also dark rings of low uncertainty which I think occurred where the noise happened to be placed in locations in both frames where it caused the algorithm to perceive a border. Notice that there are less rings than noisy pixels, because only some of the noisy pixels made meaningful borders.

6 Conclusion

The differences in Horn-Schunck and Lucas-Kanade aside, the Bayesian algorithm shows a huge amount of promise in these experiments. It is able to handle small noise better that the non-Bayesian method. It appears that it does not perform very well with similar intensity values, but this could be fixed by using a Bayesian Lucas-Kanade algorithm or even with some image preprocessing techniques. Choosing a technique would have to be decided on a case by case basis, but I believe the results of this paper have certainly shown that there are situations where a Bayesian optical flow algorithm will have significant upsides over a frequentist one.

References

[1] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185–203, 1981.

[2] J. Sun, F. J. Quevedo, and E. Bollt. Bayesian optical flow with uncertainty quantification. *Inverse Problems*, 34(10):105008, Aug 2018.

7 Appendix A

I wanted to include the final diagrams for each experiment, for instance the estimated flow when the magnitude was 10, to show some examples visually. As my paper was already getting pretty long I decided to add them in the appendix.

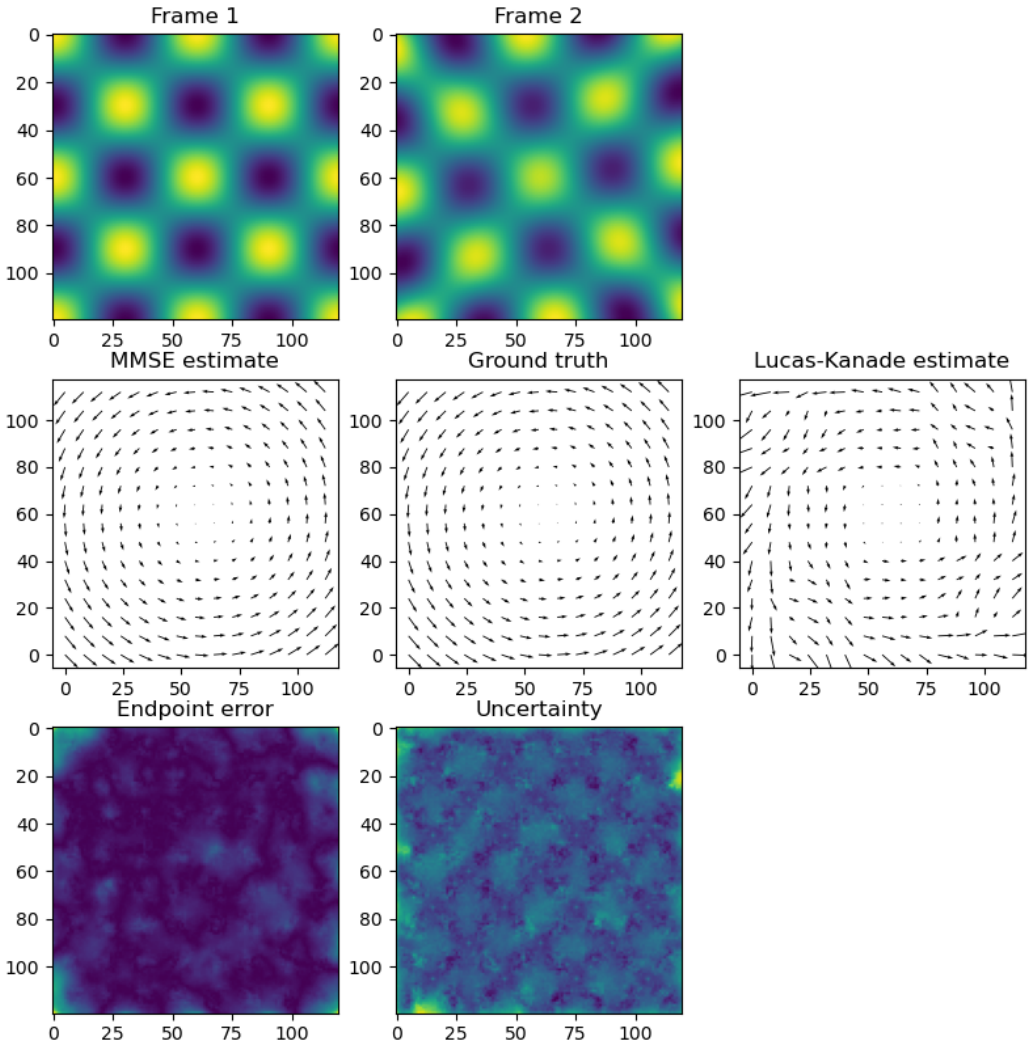

Figure 5: Output when the magnitude that each pixel moved is 10 pixels. The rotation in the frames is much more apparent than in figure 4, and while the Bayesian algorithm still functions almost perfectly, Lucas-Kanade is much worse at calculating the flow.

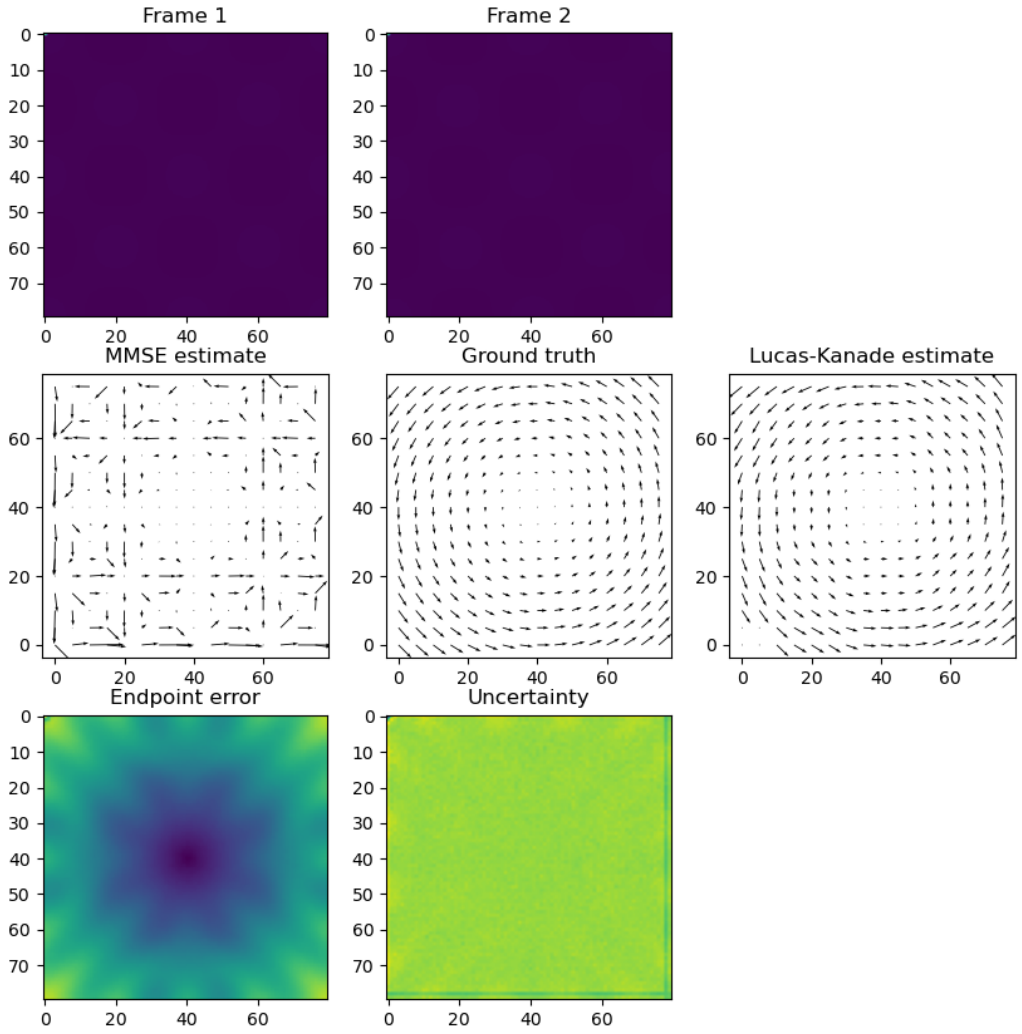

Figure 6: Output when each pixel is scaled down to 1% of its original intensity. While the Bayesian algorithm's flow graph looks very bad, when the error is calculated only using the uncertainties above the mean its error remains low.

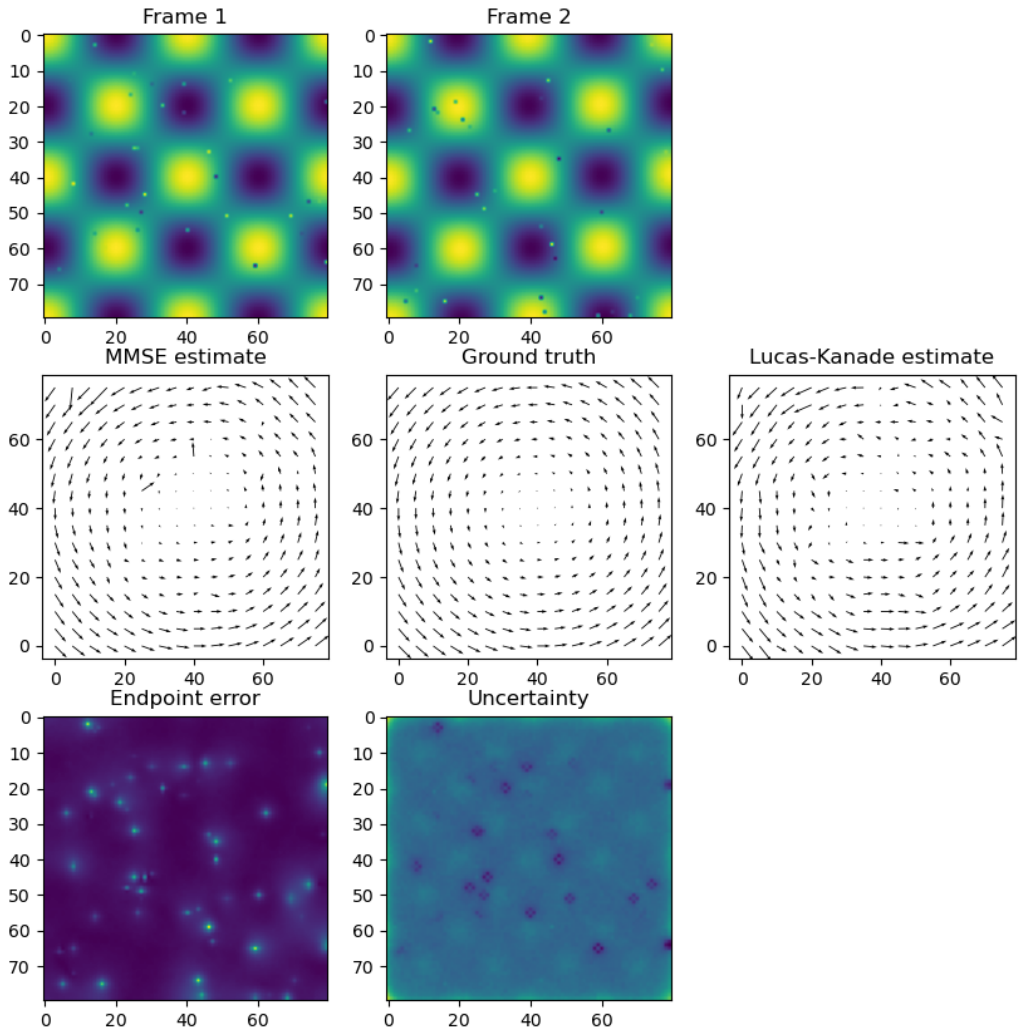

Figure 7: Output when random noise is introduced in 0.5 percent of the pixels.

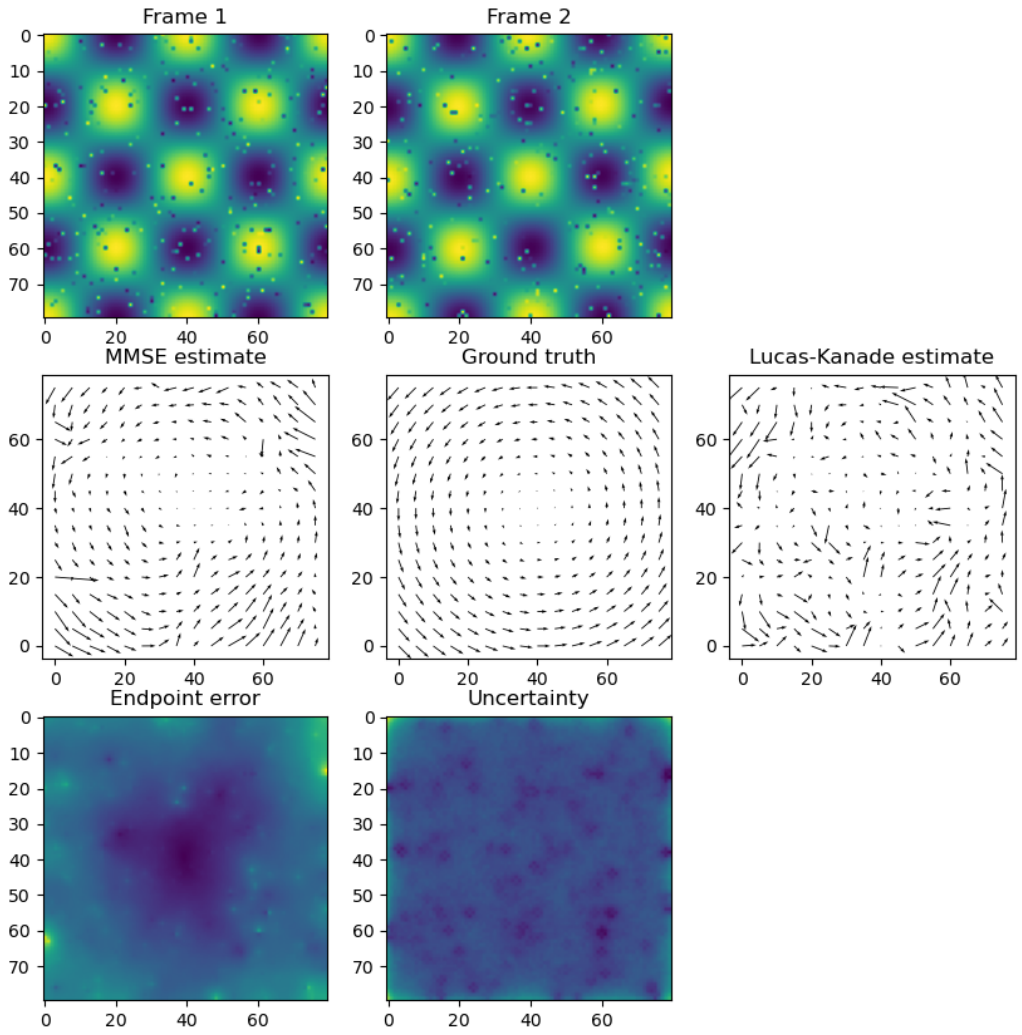

Figure 8: Output when random noise is introduced in 5 percent of the pixels.

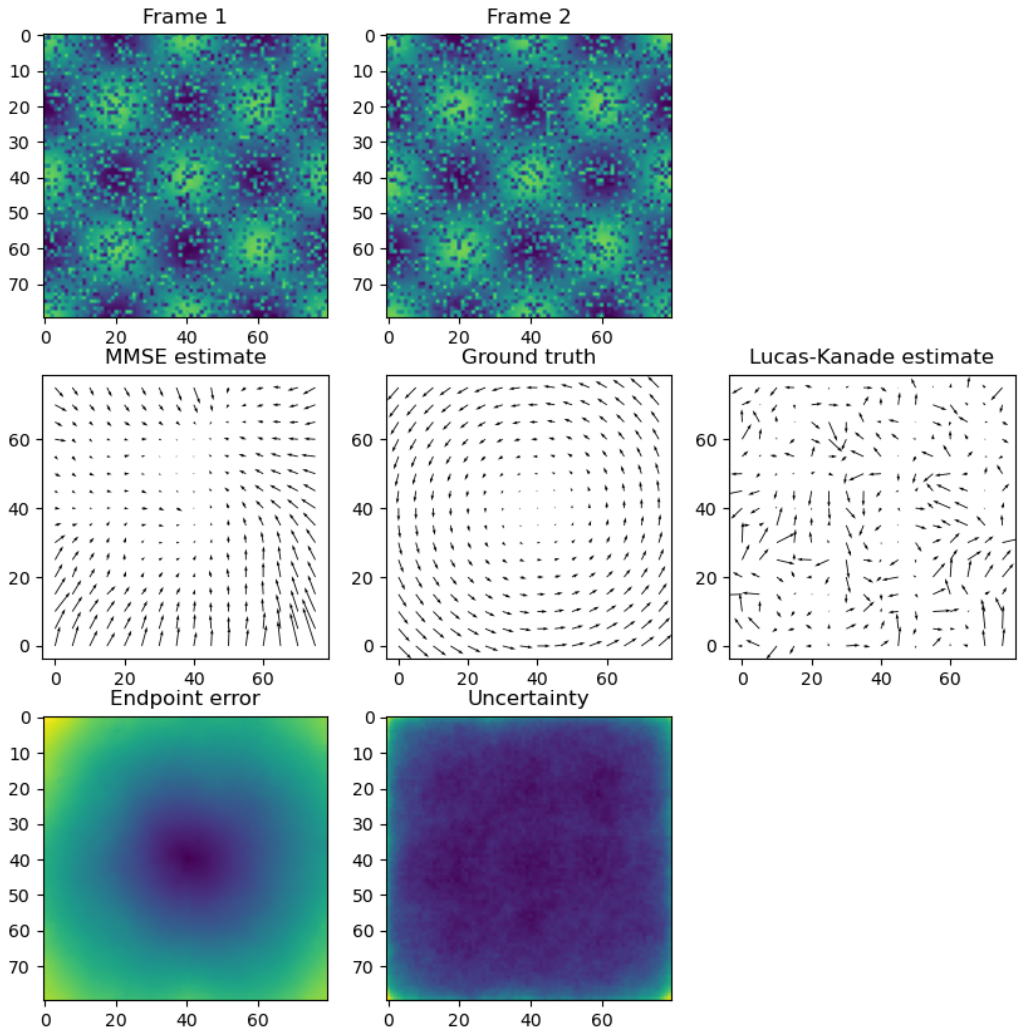

Figure 9: Output when random noise is introduced in 50 percent of the pixels.