

Using Optical Flow for Stabilizing Image Sequences

Peter O'Donovan
502425

Cmpt 400
Supervisor: Dr. Mark Eramian
April 6,2005

1 Introduction

In the summer of 1999, the small independent film *The Blair Witch Project* was released to critical acclaim and massive audience turnouts. One significant feature was the handheld camera motions which helped emphasize a disturbing realism. While films had been shot with handheld cameras for decades, this recent release served to highlight one of the drawbacks of the technique. Numerous audience members complained of nausea due to incredible jitter in the movie. Even when the problem is not as acute, jitter in image sequences remains an annoying effect, diminishing aesthetic pleasure and making interpretation for both humans and computer vision systems increasingly difficult. This project aims to develop a small system to correct for jitter in movies that follow some basic assumptions. We will allow a stationary or moving camera and moving objects within the scene. However, we will limit the motions of the camera and objects to translation and make certain assumptions about using a visible stationary background to stabilize the image.

2 Basic Approach

Before we go into detail, the general approach of the system will be explained along with some complications which arise. Then we will discuss the stages in the development of the system in finer detail. As was explained in my survey of optical flow techniques and applications [1], optical flow is a field which describes the movement of pixels between two frames, often used for image sequences like movies. It often forms the lowest level for motion analysis and is used as input to higher level systems like segmentation and tracking for interpretation. This system follows the same approach. The optical flow field is determined for each frame of the movie and is used to segment the frame into separate regions. These regions are fed as input to an object tracking system which determines information about the regions in the image and matches them with regions seen in previous frames. This allows us to track various objects through the scene. In the current implementation, the sole reason we are tracking these objects is to be able to dismiss these regions as not being the background. The background is vital in image stabilization since we assume (perhaps wrongly) this reference is stationary. We then use the movement of the background region to determine the motion of the camera. We can analyze this movement and eliminate any extreme motions which are probably caused by jitter.

There are complications which arise though. We assume very simple motions for both the objects and camera. The constraints are not merely translational motion, but only translational motion parallel to the image plane. Motions going away or towards the camera are not dealt with. They probably would not affect the overall process greatly if only objects were moving in such a manner. However, if the camera itself was zooming in or out, this would seriously affect the reliability of the results.

3 Implementation Details

The implementation of the system uses the OpenCV computer vision library. The main reason for using the library was an optimized function for hierarchical optical flow calculation. A previous implementation in OpenGL failed partly due to the speed but mostly because of the necessity for a hierarchical approach. In movies with fast movements, many frames will have optical flow vectors with large displacements. Without a hierarchical approach, these large vectors are not reliably found. Part of the reason for using OpenCV was the complexity of such approaches, combined with the speed of OpenCV and also the supposed reliability of the flow vectors. The optical flow field is the lowest level input so you must trust these results to trust later interpretation. The method used to calculate the optical flow, developed by Bouquet [2] is based on Lucas and Kanade's method and uses hierarchical Gaussian pyramids. However, after the using the method for some time, certain bugs have arisen. I am not certain whether they can be found in the OpenCV function or this project's code. Further exploration is required. Also, before the optical flow is determined for each frame, a simple blurring is used to remove noise.

As was explained in the survey [1], due to the aperture problem not all optical flow vectors are fully defined. The approach taken here is to use feature detection to find regions of high two-dimensional gradient in the image and determine the optical flow vectors for those points. This allows us to trust the results of the optical flow stage without worrying about their reliability. The Moravec operator [3] finds good features in the image by taking the autocorrelation of a pixel in 4 directions (vertical, horizontal, and the two diagonals). The minimum of the 4 is the value returned. This value will be high only at a point of high gradient in two directions. If the pixel is part of a homogeneous region or an edge, the value will be low.

4 Image Stabilization

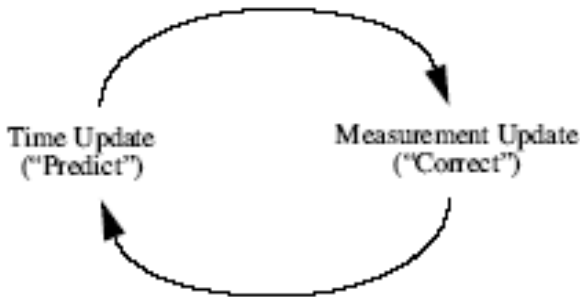
4.1 Stationary Background

Now that we have found an optical flow field for each frame, we will consider the simplest case for image stabilization: a stationary background and no moving objects. If we can assume no moving objects than the entire image is a single region corresponding to the background. If we can assume that the background should be stationary, then any motion detected is jitter which must be compensated for. The average of all the optical flow vectors will be the movement of the frame. Note that this doesn't deal with the depth disparities in images. Motion parallax will produce shorter optical flow vectors for distant objects and longer vectors for nearer objects. However, since we are taking the average, the value will be in the middle of the range and give acceptable results. The average displacement is then used to simply translate the image back to its stationary location. This will produce a black border where the image lacks information and extra pixels will be cut off. Since this will result in a smaller image, one approach is to use mosaicking so that pixels from previous frames can be used to fill in the black areas.

4.2 Moving Background

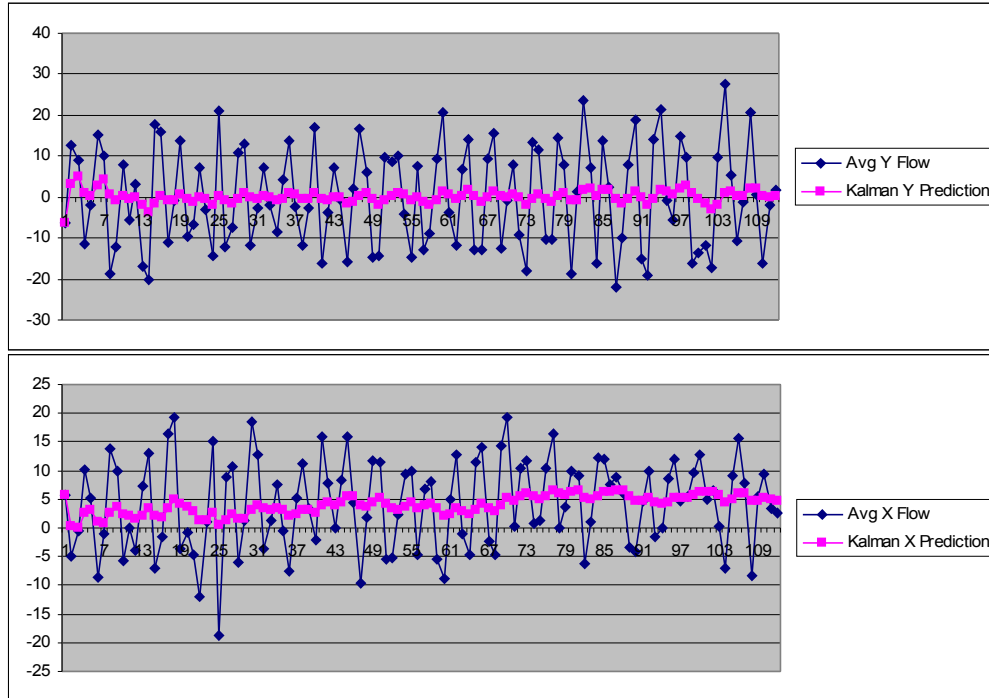
The second case we consider is a moving background and no moving objects. This corresponds to a camera moving in the environment, although only with our simple motion assumptions. We use the same average displacement as the first case but now we must perform further analysis. We must isolate intentional motions from jitter. We use Kalman filters to help us analyze the information. For a detailed explanation of Kalman filters see Welsh and Bishop [1]. However, for this paper we will limit ourselves to a brief explanation.

Welsh and Bishop define the Kalman filter as “a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error.” The Kalman filter provides predictions of the state of a model for the past, present and future, based on the previous states of the model. The equations for the Kalman filter are grouped into two sections: prediction and update. The prediction equations allow you to guess the model for future states. The update equations are used to correct the Kalman prediction once a measurement of the system has been made.



Kalman Prediction/Update Cycle

This Kalman cycle applies perfectly to the measurements and stabilization we make for our image sequences. The Kalman filter keeps a model of the movement in the X and Y dimensions. We use the equations to predict how much displacement will occur in either of the dimensions. Then we take the measurement of the displacement and update the Kalman filter.



Kalman predictions and actual measurements for a movie. The X-axis is time and the Y-axis is displacement.

As the graphs show, we use two one-dimensional Kalman filters to describe the movement in the X and Y directions. For each position in time, the Kalman filter predicts the current displacement in that direction. This prediction is compared to the actual displacement measured from the image. The difference between the two is considered the error or jitter. We will shift the image back according to this difference. As we can see in the graph, because the camera is panning left, the values oscillate around +5. After a few frames, the Kalman filter begins predicting this higher displacement. The values greater or lower than +5 are treated as jitter. There is little intentional movement in the Y-dimension so the Kalman filter stays around 0, so all the displacements are seen as jitter.

4.3 Moving Background and Objects

The last case we consider is a moving camera combined with moving objects. Now, certain regions in the image will correspond to objects moving independently of the camera, although again only with our simple translational motion assumptions. The first step is to segment the image into separate regions. To do this we use a very simple segmentation algorithm:

1. Get optical flow vector list
2. Normalize all vectors
3. Discretize all vectors
4. Get max of the number of vectors for each discretized area
5. If max > threshold
 - Create new region x

- For each optical flow vector
 - If within error threshold of max then assign to region x and remove vector from list
- Goto 4

While this algorithm may be simple to implement, it has drawbacks. It only captures translational motion parallel to the image plane. Objects or a camera moving towards or away from the camera is unconsidered. Rotation is also not captured. The choice of thresholds is also a sensitive issue.



Normalized and discretized flow vectors

As we can see in the previous image, there appear to be 2 or possibly 3 regions in the image which are moving with distinct motions. To segment this, we would take the maximum point and, if within a threshold, assign all vectors within some error to that region. We would continue doing this until the maximum point was lower than another threshold.

Once we have segmented the image into regions, we must track them over a sequence of frames. To do this we begin by finding information about each region. For this project, only a few pieces of information were used:

1. The center of the region
2. The average distance to the center point, i.e. density
3. The direction of motion
4. The last frame seen
5. The total frames seen

We keep a list of regions seen over the course of the movie. For each new frame, we use the information for each region to try and match with a previous region. If a match is

found within a threshold, we assign this region to previous one and update the information with a simple update equation.

$$X = \alpha(X_{\text{new}}) + (1-\alpha)(X_{\text{old}})$$

If no matched was found, the region is added to the list.

Now that we have a list of tracked regions, the next task is background determination. Remember, for image stabilization, tracking the background is of primary importance. Tracking other regions can be useful but is secondary. There is a problem however with initially finding the background of the image. For this project, we make the assumption that for the first second of video, there are no moving regions and the background takes up the entire image. This initial time is used to lock onto the background for later tracking. Once regions begin moving independently of the background, we will compare each region with our special background region from the first second.

5 Results

The results using this technique are quite promising. The stabilized movie produced using the first two cases contain significantly less jitter. There are three videos which should accompany this document. The first contains an example of jitter correction in the second case where the camera is panning left. This movie, 1.m1v, shows the original image on the left and the stabilized image on the right. Again, the large black borders are caused by shifting the frames to line up correctly.

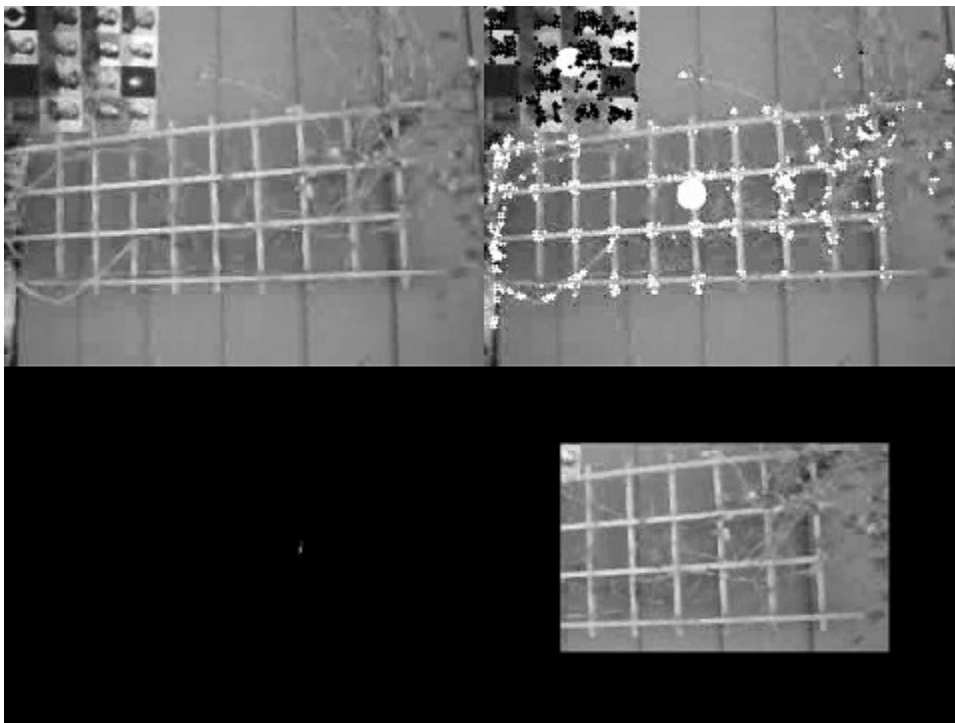


Output from a stabilized movie with intentional motion

An example of the first case was not included since the previous video obviously shows the system works well with our basic assumptions of a single background region for the entire image. The times where the background isn't moving is a subset of the second case and would be treated similarly. During testing, stationary background movies were used and the jitter was compensated for nicely.

The third case of moving regions is also perceptively more stable, but less so than the previous two cases. The additional complexity of region segmentation, tracking,

background determination, and recovery all diminish the quality of the resulting stabilized image. There are two included videos for this case, 2-1.m1v and 2-2.m1v. The upper left quadrant shows the original image. The lower left image shows the normalized and discretized flow vectors for the frame. This circular counter is used to segment the image into the various regions. The upper right image shows the regions of the image. It was difficult to represent the regions in these test movies. I decided eventually to mark the features from the background region as white and all features from other regions as black. I also included a large white dot to show the center of the region. From watching the movies, one can see how the background is often lost, which corresponds to only seeing small black dots. The lower right image is the stabilized output image.



Output from a stabilized movie with intentional motion and moving objects

6 Problems and Further Work

One major problem with the system so far is background recovery. If the background is lost because of occlusion from another object there are no methods to recover it. As long as during the loss, the background region doesn't change drastically, the system can recover when the background reappears. It merely feeds the Kalman filter's prediction back into itself as the measurement, keeping the movement of the camera along the same path as previously estimated. If the background moves significantly when lost, when it reattaches there will be a significant jerk to compensate for the updating of the Kalman filter. Also, if the background reappears in a different part of the screen, the system is not intelligent enough to recover. For example, consider a large van moving right in front of the camera. The background will be pushed to the right

hand side of the image while the van moves by the camera. As the van leaves, the background will reappear on the left side of the image. The current system cannot handle this difficulty.

There are further limitations and problems with the current system. Firstly, the tracking of regions is quite poor. Often objects moving are not properly tracked and many extraneous regions are created during an object's motion. Our only salvation is that we don't really use the information from these tracked regions other than the background. These other moving regions are simply dismissed. Their flow vectors are ignored for the final displacement measurement given to the Kalman filter. Only the background region is used for this measurement and often the background is large enough to compensate for the errors in the tracking. However, if more robust intelligence was put into the system, i.e., background recovery or analysis of object's motion, then a more robust tracking system would also need to be developed.

As was previously mentioned, a significant drawback of the system is the simple translational motion assumptions for both camera and objects. This is a fairly complex problem to deal with. While many systems utilize more complex motion parameters, in general, this is a difficult problem since there are many ambiguous motions. The optical flow calculation should also be improved upon. There are many instances where odd bugs seemed to occur in the optical flow field, perhaps cause by the OpenCV function. With more time, this function should be re-implemented.

Another area that could be improved upon is using mosaicking to fill in the region of the image with black bars. Since we are shifting the image over, these areas are unavoidable. However, we have information for this area from previous frames. By combining this information from separate frames, we can remove the black bars. The last area which could be explored is using the previous segmentation and region information to make more informed segmentation of the following frames. Though a much more general issue than this project entails, the problem is definitely an interesting area worth further exploration.

Conclusions

This project has developed a system for compensation of jittery camera motion. The problem was divided into 3 general cases: stationary background, intentional camera motion with no moving regions, and intentional camera motion with moving regions. Kalman filters are used to model the movement of the camera and decide which motions are intentional and which are jitter. Moving regions are segmented and then tracked between frames. There are many complications and assumptions that have been made in this implementation. We assume a subset of translational motion, no moving regions in the first second and background regions which do not jump between areas of the image. Within these assumptions though, the results are quite promising. Movies are significantly less jittery and with further work could be improved upon immensely.

References

- [1] Peter O'Donovan. Optical Flow: Techniques and Applications. 2005.
- [2] Jean-Yves Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm", Intel Corporation, Microprocessor Research Labs, OpenCV Documents, 1999.
- [3] H.P. Moravec, *Towards automatic visual obstacle avoidance*, Proc. 5th Int. Joint Conf. ArtificialIntell. (1977), 584.
- [3] Greg Welch, Gary Bishop. An Introduction to the Kalman Filter. Technical Report TR95-041, University of North Carolina at Chapel Hill, 1995. Online version is available at http://www.cs.unc.edu/~welch/kalman/kalman_filter/kalman.html