Camera Shake Reduction from Video

By:

Houman Kamran

Semester Project for:

EE7700 – Digital Video Processing

Electrical and Computer Engineering

Louisiana State University and A&M College

1. Introduction

As a very common problem, while taking a video clip using hand-held cameras, it is hard to keep the hand steady. As a result, the final video will be shaky most of the time and does not have the quality that the user expected. There have been efforts by a lot of programmers to process the video and get rid of the shake caused by unsteady hand and it has been successful to the extent that a lot of devices nowadays provide real time camera shake reduction services.

In this project, the goal is to be able to do the same thing, which is reducing shake after processing the raw video, taken by user.

In this work, it is assumed that objects in the scene have movements of their own as is expected. Also, the person who takes the video clip might as well want to pan the camera to be able to capture images of different objects in the scene. This movement is different from the shake and is called the desired movement of the camera as opposed to the shake that is called the undesired movement of the camera. So, it is important to remember that camera moves in different directions as well.

As a result of this assumption, then, the algorithm should be able to separate the camera motion from the action in the scene. This camera motion, separated from the action in the scene, consists of both the panning movement of the camera or desired movement and also the shake, known as the undesired movement of the camera, as well. Next, the algorithm should be able to separate these two, the desired camera movement from the undesired camera movement.

Another important assumption is that the shake is the result of small unwanted translation motions and small unwanted rotation motions of the camera as the user takes the video. So the goal is to detect these unwanted movements of the camera and reverse them.

2. Algorithm

The purpose of the project is to reduce the effect of camera shake from the input video and produce a steady video as the output. In order to be able to do so, the first thing is to separate the action in the scene from the camera motion. As is expected, there might be objects in the scene that have the movements of their own and this needs to be separated from camera motion before trying to reverse the effect of shaking.

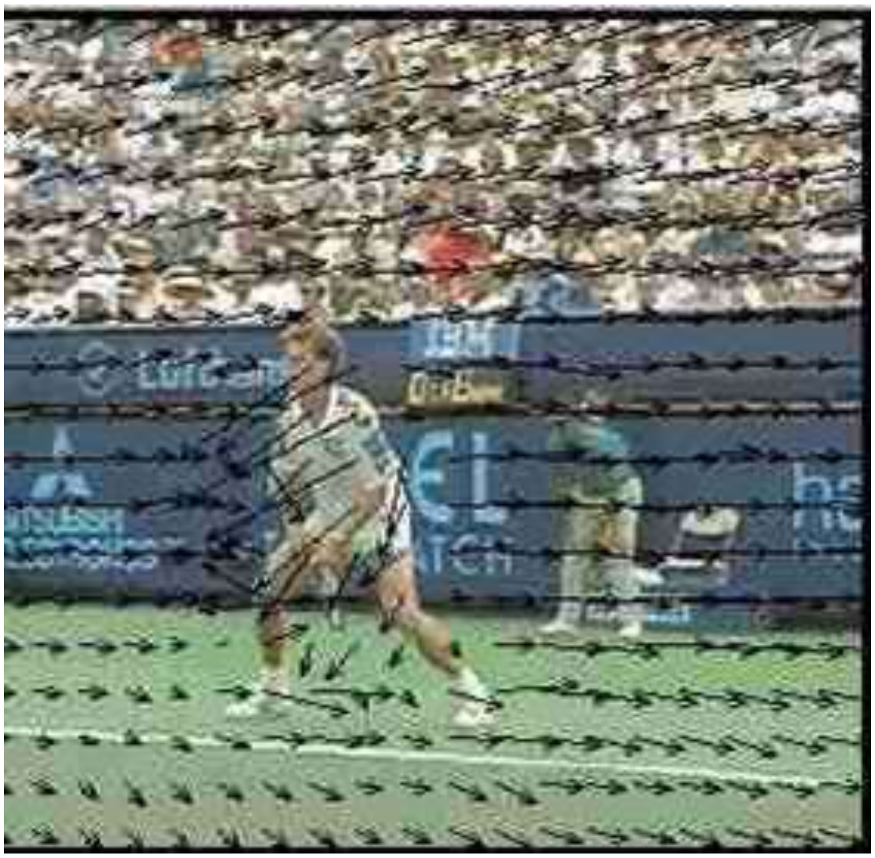


Figure 1

To make this a little bit clearer, consider Figure 1. In this image, which is an image from a sequence of images, the tennis player is moving down and to the left and that is while the camera is moving to the right. First step, then, is to separate the movement of the tennis player from the movement of the camera.

After separating the camera motion from the motion of other objects in the scene, this fact needs to be taken into consideration that not all the time the camera motion is caused by the shake. Sometimes, the big part of camera motion is because the user is panning the camera to be able to capture other objects in the scene and in fact, most of the time, the shake is more like noise sitting on top of the desired camera motion. So, the next step is to separate the shake motion from the actual desired camera motion and at the end reverse the effect of shake only.

In order to be able to remove the camera shake, this assumption is made that the shake is the result of small undesired translation movements and small undesired rotation movements of the camera, as the user is taking the video clip. The goal is to detect these movements and reverse them.

To start, it is assumed that the shake is only the result of translation movements. This means that it is assumed that frame at time t+1 is translated t_x units in x-direction and t_y units in y-direction, as a result of shake, compared to frame at time t. So in order to reverse the effect of shake and remove it, all that is needed to be done is to calculate the new position for the frame at time t+1 based on the location of the frame at time t. This new location is equal to the location of the frame at time t, shifted $-t_x$ units in x-direction and $–t_y$ units in y-direction. This needs to be repeated for all frames.

Next, it is assumed that the shake is because of the small rotation movements, as the user takes the video. In this case, let's assume α is the amount of rotation caused by shake, from frame at time t to the frame at time t+1. In order to remove the effect of shake all that is needed to be done is to rotate the frame at time t+1 to the amount of $–\alpha$ with respect to the center of the image and then replace the frame at time t+1 in the image sequence with the new rotated one.

Considering all this, the algorithm can be divided into 5 main steps as follows:

1. Find all motion in the scene
2. Separate camera motion from action within the scene
3. Separate the panning of the camera (desired camera movement) from the shake
4. Reverse the effect of the shake
5. Clean up the edges

In the next subsections every step is explained briefly.

2.1 Find all motion in the scene

The main function used for this step is Luca-Kanade. One of the important assumptions for Lucas-Kanade algorithm to work is that the motion vectors for pixels need to be very small. Since, in the case of this problem this is not guaranteed, it is preferred to use a modified version of Lucas-Kanade. This new version is called Hierarchical Lucas-Kanade and it works better for the cases that sometimes the motion vectors for some pixels are not small enough for basic Lucas-Kanade to be effective, as in this case.

Since Lucas-Kanade is an exhaustive algorithm and performs the calculation for every pixel in the image and finds the motion vector for every pixel in the image, it is very slow. This causes this step of the algorithm to be a bottleneck for the speed of the whole algorithm. On the upside, this algorithm is accurate and the motion vectors found for pixels in the image are very accurate and this improves the quality of the final output.

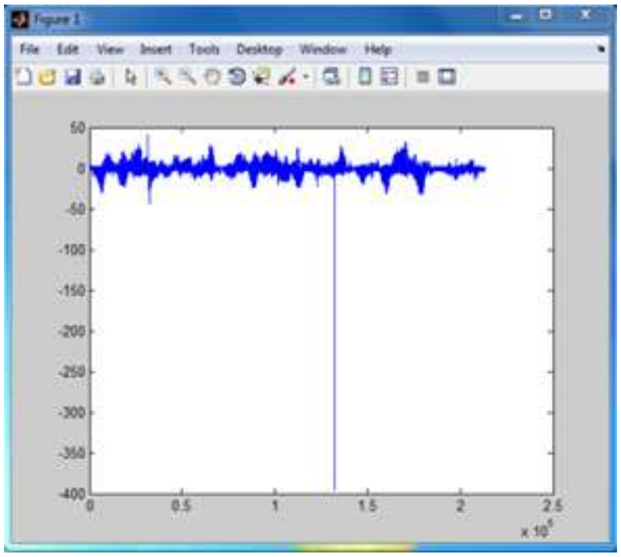
So in this step simply the motion vectors for all pixels from the frame at time t to the frame at time t+1 is calculated and is placed in matrix u1(:,:,t) and v1(:,:,t) for all t. Note that u1 is the matrix that keeps the motion vectors for pixels in x-direction and v1 is the matrix that keeps the motion vectors for pixels in y-direction.

2.2 Separate camera motion from action within the scene

A Statistical approach is being used for this part of the algorithm. As the camera moves, those pixels that do not move in the scene, like the pixels corresponding to the background, appear to be moving uniformly from one frame to another, in the opposite direction of the camera motion.

On the other hand, those pixels representing an object that moves in the scene will have different motion vectors. The motion vector for any one of this kind of pixels is the summation of the motion vector caused by the camera movement and the motion vector caused by the object movement.

The former pixels are the dominant type among all pixels in the image. In most of the cases, when dealing with natural situations, most of the pixels in the image represent background and therefore their motion vectors represent only the camera motion. Figure 2 – a below plots the amount of movement in x-direction for every single pixel from an arbitrary frame at time t to the next frame at time t+1.



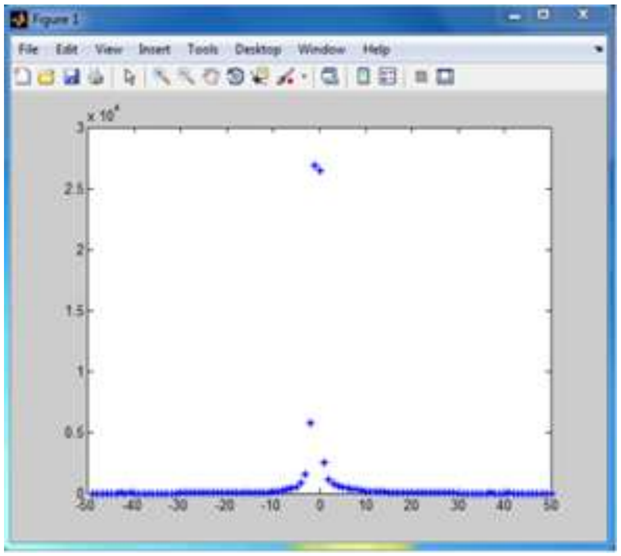


Figure 2 – a

Figure 2 - b

For some pixels the amount of motion vector seems to be out of order and can be the result of miscalculation. For some the amount of motion in x-direction, calculated in step 1, is equal to zero. These pixels are also ignored because they do not provide any information about the motion of the camera or the objects in the scene.

Figure 2 – b is the histogram of the plot in figure 2 – a, with bins of length 1. Other than the pixels with motion vector equal to zero, figure 2 – b reveals that most of the pixels have a motion vector equal or close to -1. Considering the fact that these pixels are representing the background or in other words the camera motion, this means that t_x is equal to -1 for this particular frame.

In order to calculate the value of t_x from frame at time t to frame at time t+1, a function called “ransac” is used. This function helps getting rid of the outliers. After applying ransac and finding the inliers, a translation model is fit to all the motion vectors and the values of t_x and $t_{y,}$ as part of the translation model that best fits the data, are found.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Above is the formula for a translation model. For the frame mentioned above, t_x for example will be -1.

By following the exact same steps, the amount of rotation α, from any arbitrary frame at time t to the next frame at time t+1, can be calculated. It is assumed that undesired rotation is the only reason of shake. Then, after applying ransac and getting rid of outliers, a rotation transformation model is fit to the motion vectors from frame at time t to frame at time t+1. The calculated rotation transformation will be the best rotation transformation that fits the data. Among the parameters returned in calculated rotation transformation, we have sin(α). Simply by taking the reverse sin function, arcsin, of this value we can find α, the amount of rotation from frame at time t to the next frame at time t+1. Note that α consists of both the amount of rotation caused by the desired rotation of the camera by user and also the amount of rotation caused by shake.

In next step, it is explained how to separate the desired component of movement in x-direction from the component of the movement in x-direction that is caused by shake. The same principles apply for motion in y-direction and rotation.

Below is a mention of rotation transformation formula used.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2.3 Separate the panning of the camera (desired camera movement) from the shake

As explained in 2.2 we can find the amount of camera motion in x-direction and in y-direction from any arbitrary frame at time t to the next frame at time t+1. As mentioned before, this camera motion is not the desired movement of the camera only nor is the undesired movement of the camera only. The camera motion obtained in previous step from frame at time t to the next frame at time t+1 in fact includes both the motion of the camera due to panning by user and also motion of the camera causes by shake which is the undesired part of camera motion.

The same argument also applies for rotation. The α, calculated in previous step, is the accumulated amount of rotation caused by the desired rotation of the camera by the user and the undesired rotation of the camera due to shake, both.

The purpose of this step of algorithm is to separate the desired movement of the camera from the undesired movement of the camera due to shake. In order to do so, motion in x-direction and motion in y-direction and rotation are all dealt with separately. First, it is assumed that the only effect of shake is translation in x-direction. Then the motion of the camera in x-direction, calculated in previous step, is decomposed into two components, the desired movement of the camera in x-direction and the translation in x-direction due to shake only. The same steps are followed for motion in y-direction and for rotation as well. Here the steps are explained for the motion in x-direction only.

The motion of the camera due to shake is like noise sitting on top of desired movement of the camera. This statement also holds for the motion of the camera in x-direction only. So, the amount of movement in x-direction only, calculated in previous step is plotted in a new graph versus time. For any possible value of time t, the movement of the camera from frame at time t to the next frame at time t+1 in x-direction is calculated, as explained in previous step and all the values calculated are plotted in a new graph versus time. This is done for all possible values of t in the whole video sequence. Figure 3 - a is an example of such graph for an arbitrary video sequence.

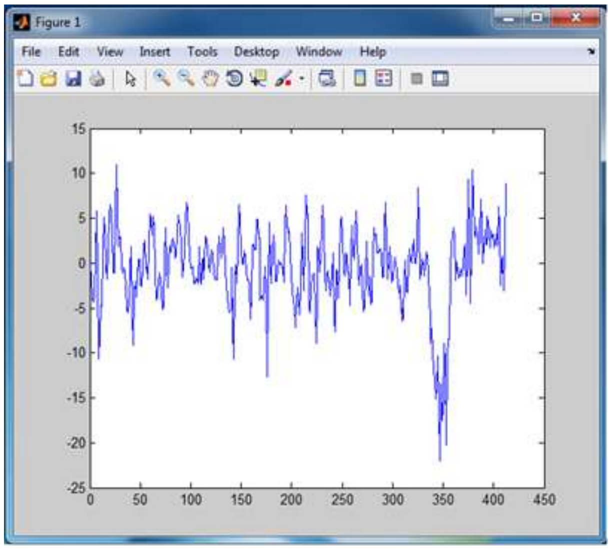


Figure 3 – a

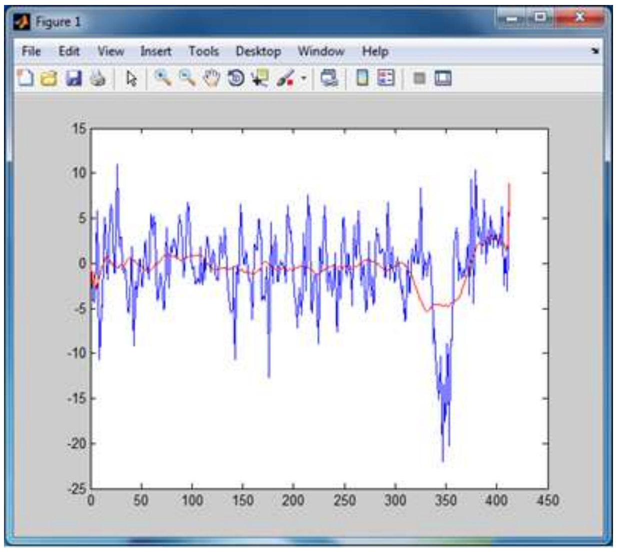


Figure 3 – b

If we assume that the shake is like noise on top of desired camera motion, to separate shake from desired camera movement, what is needed to be done is to smooth the graph shown in figure 3 - a.

In figure 3 – b the smooth version of graph in figure 3 – a is shown using red color. The value of the red colored graph at t = t_1 represents the desired motion of the camera from frame at time t_1 to the next frame at time t_1+1. The difference between the value of the blue graph and the value of the red graph at

$t = t_1$ shows the amount of motion due to shake in x-direction from frame at time t_1 to the next frame at time t_1+1. To smooth the graph shown in figure 3 – a, a function called "smooth" is being used.

Following the same principles the amount of shake in y-direction from frame at time t to the next frame at time t+1 cane be calculated for any possible value for t. Also the amount of rotation due to shake only, from frame at time t to the next frame at time t+1 will be calculated using the same method.

At the end of this step, what we have is the amount of movement in x-direction due to shake only, the amount of movement in y-direction due to shake only, and the amount of rotation with respect to the center of the image due to shake only from frame at time t to the next frame at time t+1 for any possible value for t. In the next step, we try to reverse the effect of shake from any frame to the next frame.

2.4 Reverse the effect of the shake

In this step, we go frame by frame and reverse the effect of shake from frame at time t to the next frame at time t+1, for all values of t.

First is to try to reverse the effect of translation due to shake. In order to reverse the effect of translation due to shake new position for frame at time t+1 needs to be calculated based on the location of the frame at time t. the new location for the frame at time t+1 is equal to the location of the frame at time t, shifted $-t_x$ units in x-direction and $-t_y$ units in y-direction. After the new location for frame at time t+1 is calculated, then, the frame at time t+1 should be placed in its new position in the sequence of frames.

Next is to remove the effect of rotation due to shake. New frame at time t+1 is calculated by rotation the frame at time t+1 to the amount of $-\alpha$ with respect to the center of the image. The new rotated frame then needs to replace the old frame at time t+1 in the sequence of frame.

2.5 Clean up the edges

In order to reverse the effect of shake in previous step, some frames had to be shifted and some had to be rotated as well. Shifting the frames does not create any artifact in the new calculated frame. On the other hand rotating a frame will create some artifacts in the newly calculated frame. After rotating a frame some parallel zigzag lines will appear in the image, degrading the quality of the newly calculated image. In order to have a better quality output, some process the newly calculated images even on further step, trying to get rid of the parallel zigzag artifacts.

In this project due to time limitations this step is left alone. This step could be considered as one of the main chapters in further studies.

3. Final remarks

Speed can be mentioned as the most important limitation for this implementation. The bottleneck regarding speed is Hierarchical Lucas Kanade. In order to improve speed other different algorithms to detect motion vectors can be tried out. Hierarchical Lucas Kanade is too slow and too accurate for this special purpose. So, by trying a less accurate and faster motion detection algorithm we might be able to speed up the run time a little bit.

After rotating the images, in step 4, in order to reverse the effect of shake, some jagged edges will appear as the effect of rotation. By further processing the output of this step we might be able to reduce this artifact in the output. At the moment there has not been any effort regarding reducing this kind of artifacts and it can be considered as something to work on in future work.