# Video Stabilization using Block Matching Algorithm

Liu Yang

Fudan University
13307130167@fudan.edu.cn

May 28, 2016

***Abstract***

**This report presents a video stabilization method based on block matching algorithm. It discusses a typical block matching algorithm – EBMA, and compares EBMA with some faster block matching algorithm, along with video stabilization implement result in MATLAB code.**

## I. Introduction

Video stabilization is a key problem in producing high quality video sequence, especially when we are in self-media age and much more videos are shot with smart phones, which means video stabilization is in great demand.
Typically, there are three steps in video stabilization workflow:

- motion estimation
- shaking recognition
- motion compensation

Among all three steps, the most computationally expensive and resource consumed one is motion estimation. Some mature models are discussed in [1]: based on optical flow, based on pixel, based on block, based on mesh, etc.
This report and video stabilization implement focus on block-based algorithm, starting with naive and slow EBMA, to some other improved and faster algorithm. The algorithms that have been implemented are Exhaustive Block Matching Algorithm (EBMA), 2-D Log Search Method (a.k.a Diamond Search, DS), Three-Step Search Method (TSS). Multi-Resolution Motion Estimation with Hierarchical Block Matching Algorithm (HBMA) is also introduced. Section II explains how these algorithms work. Section III makes a comparison between the first three algorithms. Section IV presents a way to recognize shaking and shows the video stabilization result with motion compensation.

## II. Block Matching Algorithm

Considering motion estimation between two given frames, $\psi(x, y, t_1)$ and $\psi(x, y, t_2)$, we define motion vector (MV) at $\mathbf{x}$ between time $t_1$ and $t_2$ is the displacement of this point from $t_1$ to $t_2$. We will call the frame at time $t_1$ the anchor frame, and the frame at time $t_2$ the tracked frame. The anchor frame can be either before or after the tracked frame, as illustrated in Figure 1. It is forward motion estimation when $t_1 < t_2$ while backward motion estimation when $t_1 > t_2$. We use $\psi_1(\mathbf{x})$ to denote anchor frame and $\psi_2(\mathbf{x})$ to denote tracked frame.
In this report and the implement, we use backward motion estimation.

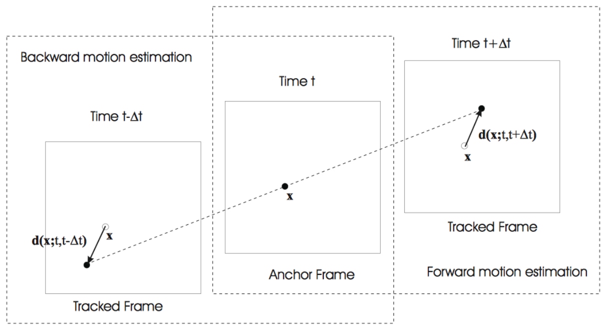

**Figure 1:** *Forward and backward motion estimation*

The idea behind motion estimation is to find the most similar part of two continuous frame, use the displacement of this two matched part as the motion vector. Block Matching Algorithm (BMA) is a collection of block-based motion estimation algo-

rithms. BMA divide the whole video frame into non-overlapping small regions, usually called blocks, and assume that the motion within each block can be characterized by a simple parametric model, e.g., constant, affine or bilinear. If the block is sufficiently small, then we can get a quite accurate motion estimation.
In the discussion below, $\Lambda$ will denote the whole video frame. And we divide a frame into $M$ blocks, we use $\beta_{ij}$ to represent the $i$-th row, $j$-th column block. So the block partition should satisfy:

$$\cup\beta_{ij} = \Lambda, \text{and} \quad \beta_{ij} \cap \beta_{i'j'} = \emptyset, i \neq i', j \neq j'.$$

In this report, the block is always square shape. And we assume the motion in each block is constant, i.e., the entire block is in same motion, which is called *block-wise translational model*.
The task of BMA is to find a single motion vector for each block. Given a block $\beta$ in the anchor frame, motion estimation is to find a most matched block $\beta'$ in the tracked frame so that the error between this two blocks is minimized. And the displacement vector $\mathbf{d}$ between this two blocks is the MV of the block in tracked frame. Since we simply adapt the block-wise translational model, estimated block can be represented as $\beta_{es} = \beta_{an} + \mathbf{d}$. So the error can be written as:

$$E(\mathbf{d}_m, \forall 1 \leq m \leq M) = \sum_m \sum_{\beta \in \beta_{ij}} |\psi_2(\beta + \mathbf{d}_m) - \psi_1(\beta)|^p$$

where $\psi$ represents the gray value, $\psi_2$ for tracked frame, $\psi_1$ for anchor frame.
Because all blocks in anchor frame is estimated independently, so the error minimizing problem is to minimize estimation error for each block, which is:

$$E_m(\mathbf{d}_m) = \sum_{\beta \in \beta_{ij}} |\psi_2(\beta + \mathbf{d}_m) - \psi_1(\beta)|^p \qquad (1)$$

To reduce computational load, the Mean Absolute Difference (MAD) error ( $p = 1$) is used in this report and the implement.

## i. Exhaustive Block Matching Algorithm

Exhaustive Block Matching Algorithm (EBMA) is also called Full Search. It is the simplest but most computational expensive block matching algorithm of all.
For a given block $\beta$ in the anchor frame, EBMA searches surrounding area of $\beta$, compares $\beta$ with all candidate blocks $\beta_{can}$ in the tracked frame and find best matched candidate block $\beta'$ with minimum error. Then the displacement between anchor block $\beta$ and best matched block $\beta'$ is the estimated motion vector.
In EBMA, search area can be parameterized by $R_x$ pixels horizontally and $R_y$ pixels vertically. Both $R_x$ and $R_y$ is symmetric with respect to the center of the anchor block $\beta$. Usually the shaking is random both horizontally and vertically, so search area is a square area with $R = R_x = R_y$.
EBMA can be illustrated in Figure 2.

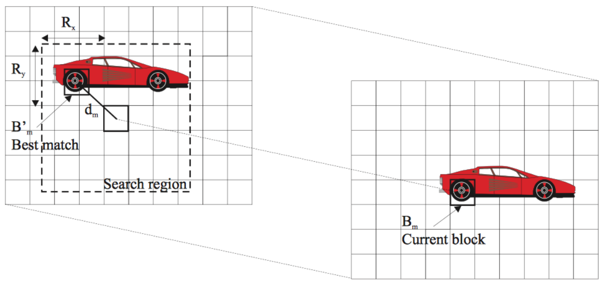

**Figure 2:** *The search procedure of EBMA for one anchor block*

For EBMA, the estimation accuracy can be specified by search step size, which is the distance between two nearby candidate blocks in the horizontal or vertical direction. In this report and the implement, the same step size is used in both directions. In most case, step size is one pixel and is called *integer-pel accuracy search*. On the other hand, for those low resolution frame, a more accurate motion estimation is needed. So in this case, we will use fractional-pel accuracy search. But there exists a problem that there may not be corresponding candidate points in the tracked frame for certain sample anchor points. To realize a step size of $1/K$ pixel search, the tracked frame firstly needs to be resized by factor $K$. Typically the $K = 2$, which is know as *half-pel accuracy search*. Half-pel accuracy search is shown in Figure 3.

Although EBMA can get a global optimal motion estimation and with half-pel accuracy search it can get more accurate, one obvious disadvantage of EBMA is that it requires intense computation when goes on with full search regardless of the possibility of candidate block to be the most matched block.
Because of the computation problem, several faster algorithms have been developed to get a trade-off between the estimation accuracy and computation efficiency.

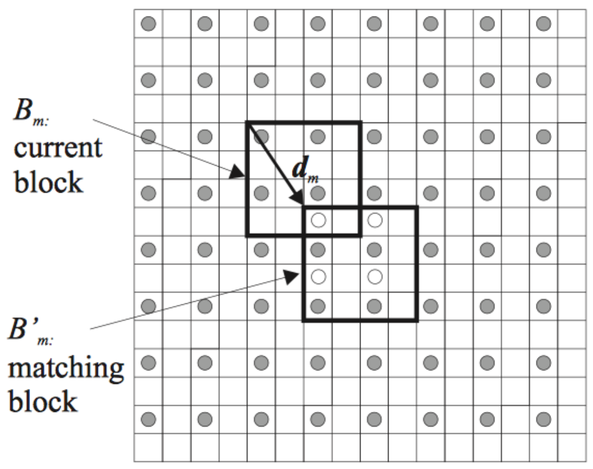

**Figure 3:** *Half-pel accuracy block matching. Filled circles are samples existing in the original tracked frame, open circles are samples to be interpolated for calculating the matching error.*

## ii. 2-D Log Search Method

2-D Log Search Method is also known as Diamond Search. It is illustrated in Figure 4

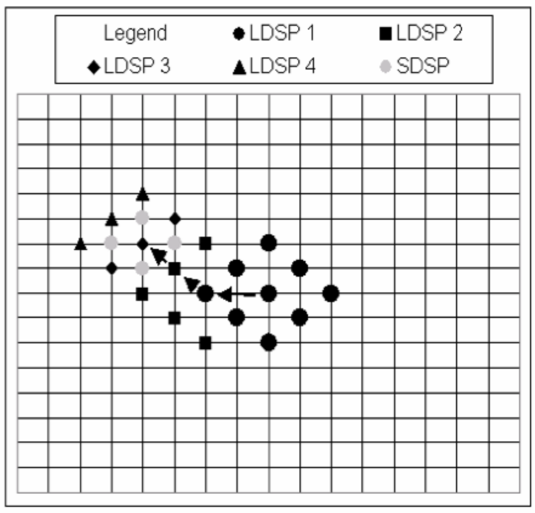

**Figure 4:** *2-D Log Search Method. Search points start with black circles, end with gray circles.*

It starts from the position corresponding to zero-displacement of the anchor block. In each step, nine nearby points within diamond shape will be tested to get a best matched point. Then the center of search area will move to the best matched point resulting from the previous step and repeat the step. In 2-D log search method, the search step size, also represents the radius of the search diamond shape, is reduced by one if the best matched point is the center point of the diamond, or on the border of the search area. Otherwise, the step size will remain same in next search step. The final step is reached when the step size reaches 1 pixel range. Usually the initial step size is set to be half of the max search range.
As the fact that the number of steps and the total number of search points depends on the actual motion vectors, there is no limit to the number of 2-D log search steps, so it could find global minimum accurately while the computation expense goes down quickly.

## iii. Three-Step Search Method

As illustrated in Figure 5, the search starts with a step size equal to or slightly larger than half of the maximum search range. In each step, nine search points, which are in a square shape around the center of the previous search step, are compared. The step size is reduced by half after each step, and the search ends with step size reaching 1 pixel range. At each new step, the search center is moved to the best matched point resulting from the previous step.

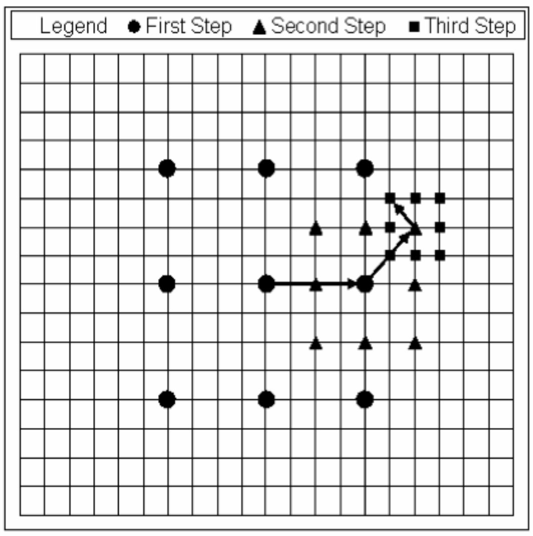

**Figure 5:** *Three-Step Search Method. Search points start with black circle, end with black square.*

Let $R_0$ represent the initial search step size, there are at most $L = \lfloor \log_2 R_0 + 1 \rfloor$ search step. At each search step, eight points are searched, except in the very first nine points searched. So the total number of search points is $8L + 1$. For example, for a search range of $R = 32$, with EBMA, the total number of search points is $(2 * R + 1)^2 = (2 * 32 + 1)^2 = 4225$, whereas with three-step search method, the number is 41, ($L = \lfloor \log_2 R_0 + 1 \rfloor = \lfloor \log_2 \frac{32}{2} + 1 \rfloor = 5, 8 * L + 1 = 41$ ). Three-step search method is over 100 times efficient than EBMA.

## iv. Hierarchical Block Matching Algorithm

Among above motion estimation methods, there two major difficulties when obtaining the correct solution:

1. the minimization function usually has many local minimum and it is not easy to reach the global minimum.
2. the expense of computation is very high.

To cope with above two problems, hierarchical block matching algorithm (HBMA) with multi-resolution approach is developed. It searches the optimal solution in successively finer resolutions, as illustrated in Figure 6.

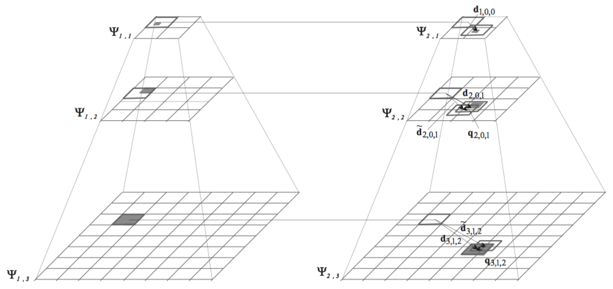

**Figure 6:** *3-level hierarchical block matching algorithm*

First, we get pyramid-like representations of anchor frame and tracked frame. Each level is a resized frame as a reduced resolution representation, where bottom level is the original one. The higher the level is, the lower its resolution is. Typical resolution reduction factor between two successive levels is 2, both horizontally and vertically. Then the motion vectors between corresponding levels of the two pyramids is estimated, starting from the top coarsest level and processing to the lower finer level repeatedly.
Here we assume the same block size is used at different levels, so that the number of blocks is reduced by half in each dimension as well. Let the motion vector for block $(i, j)$ as level $l$ be denoted by $\mathbf{d}_{l,i,j}$. Starting from top level, we first find the motion vectors for all blocks in this level. At each lower level $l > 1$, for each block, its initial motion vector $\widetilde{\mathbf{d}}_{l,i,j}$ is interpolated from a corresponding block in level $l - 1$ by

$$\widetilde{\mathbf{d}}_{l,i,j} = 2 * \mathbf{d}_{l-1,\lfloor i/2 \rfloor,\lfloor j/2 \rfloor}$$

Then within the search area, we search and get a correction vector $\mathbf{q}_{l,i,j}$, then make the correction to get the final estimated motion vector

$$\mathbf{d}_{l,i,j} = \widetilde{\mathbf{d}}_{l,i,j} + \mathbf{q}_{l,i,j}$$

We could notice that using a block with $N$ size at level $l$ corresponds to a block size of $2^{L-l} * N$ at the full resolution. Therefore, by using the same block size, search range and step size at different levels, we actually use a larger block size, a larger search range and a larger step size in the beginning of the search, and then gradually half-reduce these three parameters in successive level steps.
The benefit of HBMA is that by first searching the solution in a coarse resolution, one can usually obtain a solution that is close to the true motion vector. In addition, by limiting the search area to a small neighborhood of the solution obtained in the previous resolution, the total number of searches can be reduced, compared to those required by directly searching in the finest resolution over a large range.

## III. Algorithm Comparison

This comparison part mainly focuses on three BMA: EBMA, 2-D Log Search Method and Three-Step Search Method.

## i. Computation Expense

As we already discussed in section II, EBMA is a most computation expensive algorithm, while 2-D log search method and three-step search method could drop the expense greatly.
As shown in Figure 7, in our comparison experiment, video has 32 frames, block is divided as $16 * 16$ *pel* square and search area is set to be 7 *pel*.

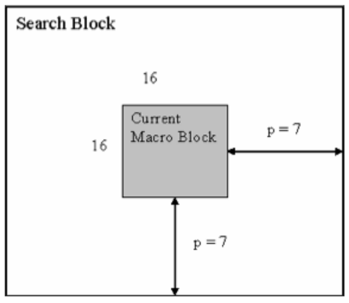

**Figure 7:** *EBMA search block with 16 pel size and 7 pel range.*

Search points for a certain anchor block in EBMA is fixed once search range is fixed, in this case is $(2 * R + 1)^2 = (2 * 7 + 1)^2 = 255$ searches per block. Since in each search of BMA, it is necessary to compute on the entire block to get MAD error value, we only need to compare the average search times per block of these three algorithms. And because the average search times of EBMA is one magnitude larger than the other two, we do not plot it in the result figure. So the computation expense comparison between 2-D log search method and three-step search method is plot in Figure 8.

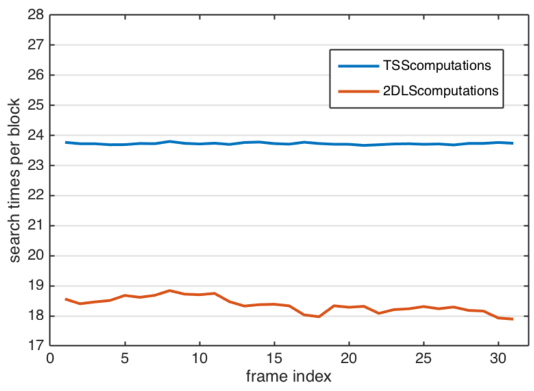

**Figure 8:** *Computation expense comparison between TSS and 2DLS*

In Figure 8, we can see that 2-D log search method and three-step search method is actually one magnitude faster than EBMA. Meanwhile 2-D log search method is slightly faster than three-step search method, which means our experiment video undergoes small shaking.

## ii. Accuracy

In equation 1, when $q = 2$, then equation 1 goes to be Mean Square Error (MSE), which is also an error evaluation method.
Instead of the MSE, the *peak signal to noise ratio* (PSNR) in decibel (dB) is more often used as a quality measure in video coding. The PSNR is defined as

$$PSNR = 10 \log_{10} \frac{\psi_{max}^2}{MSE} \tag{2}$$

where $\psi_{max}$ is the peak (maximum) intensity value of the anchor frame.
PSNR value is used in our algorithm accuracy comparison. PSNR evaluates the quality of compensated frame resulting from estimated motion vectors.

In Figure 9 , we plot PSNR values of these three algorithms.

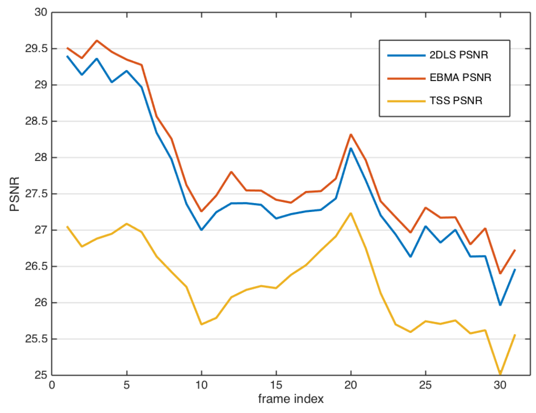

**Figure 9:** *Accuracy comparison between EBMA, TSS and 2DLS*

In Figure 9, we can see EBMA gets the highest PSNR value, which means it gets the most accurate compensated result. 2-D log search method gets a result close to the best one, which is caused by the unlimited search steps by this algorithm to able to get the global optimal motion vector. Three-step search method gets the worst compensated results in all frames because it only allows limited search steps, usually 3∼4 times, and easily to get a local optimal motion vector.
So although EBMA is most computationally expensive and resource consumed, it could get best result. While three-step search method drop the search times dramatically, it might get a not so good result. 2-D log search method is a good trade-off between computational expense and accuracy.

## IV. Experiment

### i. Block Motion Vectors

With the implemented MATLAB code for EBMA, 2-D log search method, three-step search method and HBMA, we could get the following results showing the estimated motion vectors. We use 'red-cyan' color channel to distinguish anchor frame and tracked frame, red represents tracked frame and cyan represents anchor frame.

1. motion vectors from EBMA is in Figure 10
2. motion vectors from 2-D Log Search Method is in Figure 11

3. motion vectors from Three-Step Search Method is in Figure 12

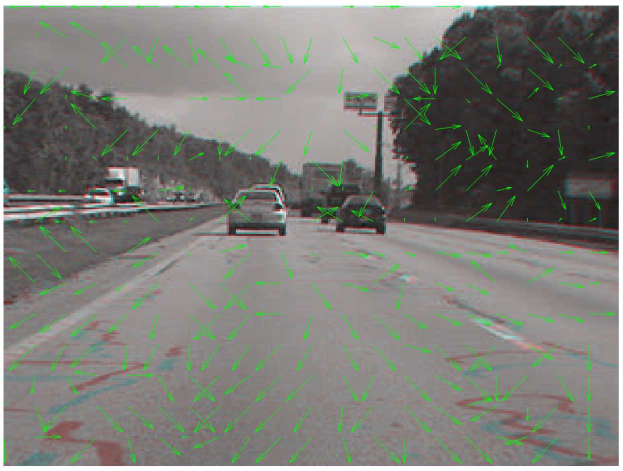

**Figure 10:** *Motion vector resulting from EBMA*

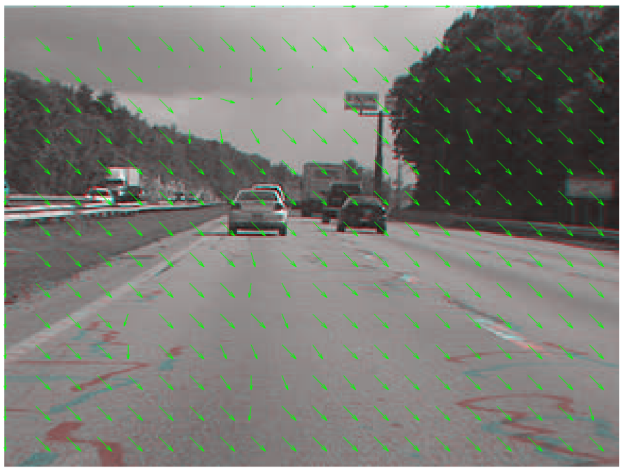

**Figure 11:** *Motion vector resulting from 2-D log search method*

## ii. Global Motion Vector

With the estimated motion vector of each block done by presented BMA, we could carry on with shaking recognition part. From block motion vectors of a frame, we can evaluate a vector identifying unwanted movements. This vector summarizes the motion of the frame and is called Global Motion Vector (GMV). One way to get GMV is simply letting the most frequent vector presented in the block motion vectors be the GMV. This way assumes that most part of the frame undergoes the exactly same displacement.
But the drawback of BMA is that not all block motion vectors are reliable. For example, considering Figure

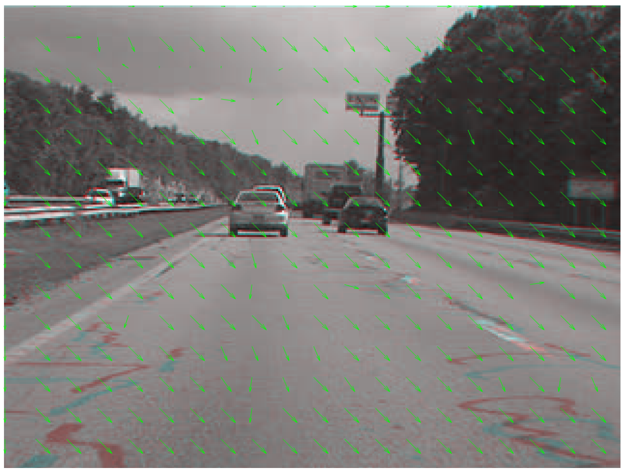

**Figure 12:** *Motion vector resulting from three-step search method*

13 below, a homogeneous block in a homogeneous area can give a wrong motion vector as a poor match is found. In Figure 13, block MAD is calculated by

$$MAD = \frac{1}{N^2} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} |Y_{i,j} - \bar{Y}| \qquad (3)$$

where $\bar{Y}$ is the average gray value in the block.
So the lower the MAD of block $\beta$ is, the more likely $\beta$ is homogeneous.

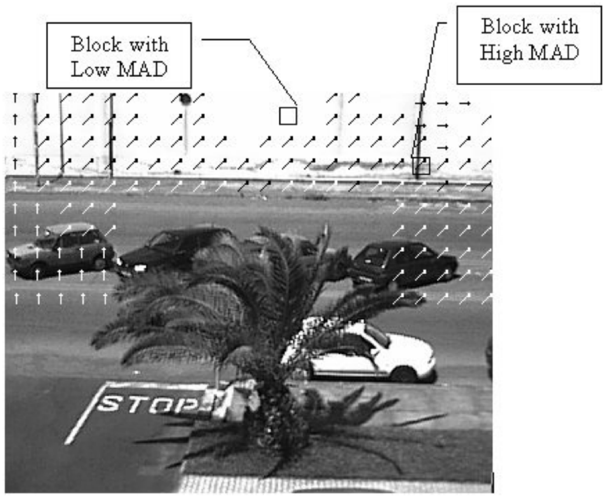

**Figure 13:** *Different MAD to present homogeneousness*

With simple counting method to get GMV, a number of unreliable motion vectors might be used and will lead to wrong GMV result.
An improved way to get GMV is weighted counting. Different weights for the motion vector counting are

considered. Homogeneous blocks will have a low weight while block with high MAD value will have a higher weight. In this way a most reliable block will be counted with more strength than a normal block. Weight values are assigned considering that higher the MAD value is, the bigger must be the weight associated with its motion vector. In this report and its implement, we use a weight table 1.

**Table 1:** *Weights for BMV*

| MAD | Weight |
|---|---|
| MAD < 5 | 0 |
| $5 \leq$ MAD < 40 | 1 |
| $40 \leq$ MAD < 60 | 2 |
| $60 \leq$ MAD < 80 | 4 |
| $80 \leq$ MAD < 100 | 8 |
| MAD $\geq$100 | 16 |

Now the most weighted motion vector will be chosen as the global motion vector.

## iii. Motion Compensation

The above global motion vector of tracked frame corresponds to the anchor frame. To have a stabilization of the whole series of frames, the compensation must be done corresponding to a common reference frame, which in our case is the first frame of the video. So each tracked frame's global motion vector will accumulate to get the absolute global motion vector. Absolute GMV is considered as the vector referred to the absolute coordinate system. So the compensation will be done relying on this absolute GMV.
The final big problem in motion compensation is to distinguish between jiggling and panning. Jiggling is the shaking that lowers the video quality and need to be stabilized. For most case, jiggling is in small scale and in random direction. Panning is intended movement by the video shooter and we want to keep panning movement in stabilized video. Unlike jiggling, panning is in coherent direction and in a bigger movement than jiggling.
In [2], it uses absolute GMV and set a hard threshold to discriminate jiggling from panning.
Instead, in this report and its implement, we use another way to distinguish between jiggling and panning. When jiggling appears, the frame is changed by a little, which means the MAD values of anchor frame and tracked frame is almost the same. While in the panning, most of the frame will be moved out by a certain amount of pixels, leading to a quite large change in the MAD values of anchor frame and tracked frame. So we use the following judgment (Figure 14) to judge whether the frame undergoes jiggling or panning. Here when encountering panning, we reset absolute GMV to switch to a new video context and consider the frame after panning as a new common reference frame.

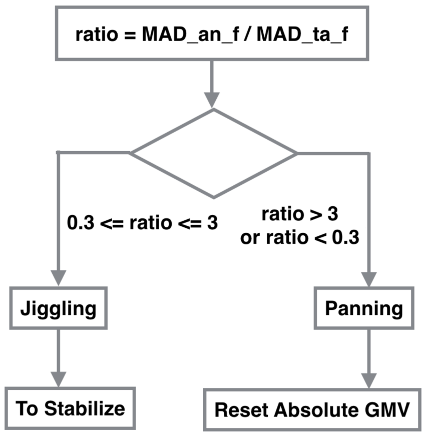

**Figure 14:** *Jiggling and Panning Judgment*

## iv. Final Result

Here lists some captures of the stabilized video 15, 16, 17. Left is the original and right is the compensated.

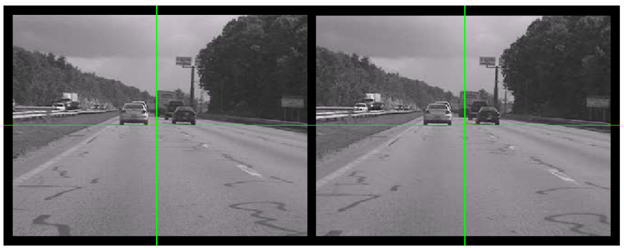

**Figure 15:** *Original frame and compensated frame at the beginning of video*

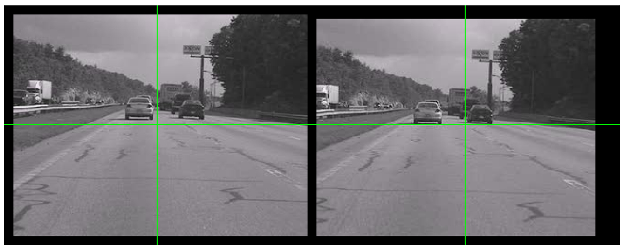

**Figure 16:** *Original frame and compensated frame in the middle of the video*

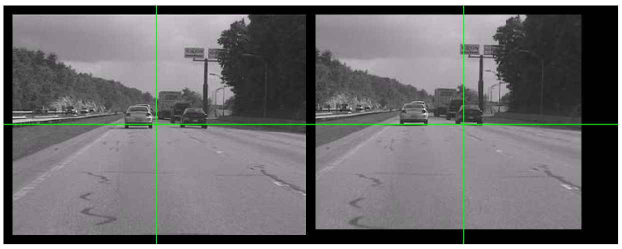

**Figure 17:** *Original frame and compensated frame at the end of the video*

The final stabilized video is available here: Youtube or BaiduYun:

## V. Conclusion

Block Matching Algorithm is a widely used algorithm in video stabilization. Among three BMAs that we compare, EBMA is the most computationally expensive but most accurate one. Three-Step Search Method requires much less computational resource but decreases the quality of estimated motion vectors. 2-D Log Search Method gains a balance between computational expense and accuracy. HBMA develops a new idea about search blocks.
Global motion vector is generated by counting block motion vectors with different weights. Discrimination between jiggling and panning is considered by applying MAD value comparison.
One obvious drawback of block matching algorithm in video stabilization is that it could hardly stabilize rotation in the frame.

## VI. Acknowledgement

MATLAB code used to generate the comparison results in section III is provided by *Aroh Barjatya* on *MathWorks Community*.

## References

[1] Wang Y, Ostermann J, Zhang Y Q. *Video processing and communications[M]. Chap. 6: Two Dimensional Motion Estimation.* Upper Saddle River: Prentice Hall, 2002.

[2] Vella F, Castorina A, Mancuso M, et al. *Digital image stabilization by adaptive block motion vectors filtering[J].* IEEE Transactions on Consumer Electronics, 2002, 48(3): 796-801.

[3] Bierling M. *Displacement estimation by hierarchical blockmatching[C].* Visual Communications and Image Processing'88: Third in a Series. International Society for Optics and Photonics, 1988: 942-953.

[4] Engelsberg A, Schmidt G. *A comparative review of digital image stabilising algorithms for mobile video communications[J].* Consumer Electronics, IEEE Transactions on, 1999, 45(3): 591-597.