Video Stabilization

Bohdan Mahometa, Yaroslav Revera, Maksym Tsapiv

May 2022

1 Problem Formulation

Video stabilization is one of the classic **Computer Vision** problems. Its importance is hard to argue with while the demand for video cameras on moving platforms is on the rise. Smartphones, drones, cars, etc – all the captured video tends to suffer from shaky global motion and rolling shutter distortion, making stabilization a necessity. In the vast majority of cases physical stabilizers are of little use, not to mention the cost of such devices. Instead, integrating an embedded video stabilization solution into the imaging pipeline of a product adds significant value to the customers. So, a computer-vision-based video stabilization approach definitely is the way to go.

We want to develop a computer-vision-based video stabilization system featuring image inpainting. The final result of our project is an application, capable of stabilizing videos.

2 Literature Review

"Full-frame Video Stabilization" paper [1] from Microsoft is one of the main sources for the topic. This paper contains a lot of general information as well as specific method for video stabilization. More specifically, it describes Video Completion with Motion Inpainting with step by step solutions, such as Mosaicing with consistency constraint, Local motion computation, Motion Inpainting, Mosaicing with local warping.

Georgios Evangelidis and Emmanouil Psarakis provide an algorithm for aligning images in their "Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization"[2] . The paper proposes a way to find the parameters of transformation that aligns two images, namely through maximizing the ECC coefficient. The paper justifies why the proposed objective function is a good metric for evaluation of alignment as well as provides a way to optimize the metric.

3 Video Stabilization

We are given a sequence of RGB images of the same size. We propose a video stabilization algorithm that operates on the sequence as follows:

1. Perform image alignment algorithm between each pair of consecutive images assuming some type of transformation between them.
2. Compute the composition of transformations between the first image of the sequence and other images.
3. Apply inverses of transformations computed in **Step 2** to each of the images.
4. Calculate the trajectory of the center of the image and rotation of each of the transformed images relative to position and rotation of the first image.
5. Smooth trajectory and rotation functions.
6. Calculate new images taking into account smoothed trajectory and rotation functions.
7. Perform video inpainting.
8. Check results with PSNR (Peak signal-to-noise ratio).

Considering the relative simplicity of **Steps 2-6**, the problem amounts to **Step 1** and **Step 7**. They are the most difficult, both in terms of implementation and mathematical justification. The following two subsections are to explain the methods used in **Step 1** and **Step 7**.

3.1 Image Alignment

In order to find the transformation aligning the first image to match the second we utilize the methodology proposed by Georgios Evangelidis and Emmanouil Psarakis in their "Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization".

Consider two consecutive video frames. Let $I_1(\mathbf{x})$ be the intensity function mapping the coordinates of the point of the first of the two frames to intensity of the image at that point. Analogously, the intensity of the second image at point $\mathbf{y}$ is $I_2(\mathbf{y})$. Pay attention that we assume that the intensities are defined at each point of $\mathbb{R}^2$.

Consider the transformation $\phi(\mathbf{x}, \mathbf{p})$ with a vector of parameters $\mathbf{p}$ mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$. Our aim is to find parameters $\mathbf{p}$ such that $I_2(\phi(\mathbf{x}, \mathbf{p}))$ is approximately $I_1(\mathbf{x})$ for any point $\mathbf{x} \in S$, where $S \subset \mathbb{R}^2$ and S is of cardinality K:

$I_1(\mathbf{x}) \approx I_2(\phi(\mathbf{x}, \mathbf{p}))$

$$S = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(K)}\}$$

Let $\mathbf{i}_1$ be a vector of intensities at points from S. Analogously, $\mathbf{i}_2(\mathbf{p})$ is a vector of intensities at transformed points:

$$\mathbf{i}_1 = \begin{bmatrix} I_1(\mathbf{x}^{(1)}) \\ I_1(\mathbf{x}^{(2)}) \\ \vdots \\ I_1(\mathbf{x}^{(K)}) \end{bmatrix}, \mathbf{i}_2(\mathbf{p}) = \begin{bmatrix} I_2(\phi(\mathbf{x}^{(1)}, \mathbf{p})) \\ I_2(\phi(\mathbf{x}^{(2)}, \mathbf{p})) \\ \vdots \\ I_2(\phi(\mathbf{x}^{(K)}, \mathbf{p})) \end{bmatrix}$$

Let $\bar{\mathbf{i}}_1$ be a zero-mean version of $\mathbf{i}_1$ which is obtained by subtracting from $\mathbf{i}_1$ the mean of elements of the vector $\mathbf{i}_1$. Analogously, $\bar{\mathbf{i}}_2(\mathbf{p})$ is a zero-mean version of $\mathbf{i}_2(\mathbf{p})$. $\hat{\mathbf{i}}_1$ and $\hat{\mathbf{i}}_2(\mathbf{p})$ are normalized versions of intensities $\mathbf{i}_1$ and $\mathbf{i}_2(\mathbf{p})$ and are insensitive to bias and gain in intensities:

$$\hat{\mathbf{i}}_1 = \frac{\bar{\mathbf{i}}_1}{\|\bar{\mathbf{i}}_1\|}, \ \hat{\mathbf{i}}_2(\mathbf{p}) = \frac{\bar{\mathbf{i}}_2(\mathbf{p})}{\|\bar{\mathbf{i}}_2(\mathbf{p})\|}$$

That makes using $\hat{\mathbf{i}}_1$ and $\hat{\mathbf{i}}_2(\mathbf{p})$ instead of $\mathbf{i}_1$ and $\mathbf{i}_2(\mathbf{p})$ particularly advantageous. Ideally, we would want to find parameters $\mathbf{p}$ that minimize the squared norm of difference between $\hat{\mathbf{i}}_1$ and $\hat{\mathbf{i}}_2(\mathbf{p})$:

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \left\| \frac{\bar{\mathbf{i}}_1}{\|\bar{\mathbf{i}}_1\|} - \frac{\bar{\mathbf{i}}_2(\mathbf{p})}{\|\bar{\mathbf{i}}_2(\mathbf{p})\|} \right\|^2 \tag{1}$$

Manipulating the last expression:

$$\|\hat{\mathbf{i}}_1 - \hat{\mathbf{i}}_2(\mathbf{p})\|^2 = \|\hat{\mathbf{i}}_1\|^2 - 2\hat{\mathbf{i}}_1^T\hat{\mathbf{i}}_2(\mathbf{p}) + \|\hat{\mathbf{i}}_2(\mathbf{p})\|^2 =$$

$$= 2 - \|\hat{\mathbf{i}}_1\|^2 - 2\hat{\mathbf{i}}_1^T\hat{\mathbf{i}}_2(\mathbf{p})$$

So minimizing (1) amounts to maximizing the ECC (Enhanced Correlation Coefficient):

$$E_{ECC}(\mathbf{p}) = \langle \hat{\mathbf{i}}_1 \ , \ \hat{\mathbf{i}}_2(\mathbf{p}) \rangle$$

It is still hard to maximize $E_{ECC}(\mathbf{p})$, so we approximate $f(\mathbf{x}, \mathbf{p}) = I_2(\phi(\mathbf{x}, \mathbf{p}))$ at point $\mathbf{p}$ with its first-order Taylor polynomial:

$$f(\mathbf{x}, \mathbf{p} + \Delta\mathbf{p}) \approx f(\mathbf{x}, \mathbf{p}) + \frac{\partial f(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p} =$$

$$= I_2(\phi(\mathbf{x}, \mathbf{p})) + \frac{\partial I_2(\phi(\mathbf{x}, \mathbf{p}))}{\partial \mathbf{y}} \frac{\partial \phi(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p}$$

Then we approximate $\mathbf{i}_2(\mathbf{p})$:

$$\mathbf{i}_2(\mathbf{p} + \Delta\mathbf{p}) = \begin{bmatrix} f(\mathbf{x}^{(1)}, \mathbf{p} + \Delta\mathbf{p}) \\ f(\mathbf{x}^{(2)}, \mathbf{p} + \Delta\mathbf{p}) \\ \vdots \\ f(\mathbf{x}^{(K)}, \mathbf{p} + \Delta\mathbf{p}) \end{bmatrix} \approx$$

$$\approx \begin{bmatrix} I_2(\phi(\mathbf{x^{(1)}},\mathbf{p})) + \frac{\partial I_2(\phi(\mathbf{x^{(1)}},\mathbf{p}))}{\partial \mathbf{y}} \frac{\partial \phi(\mathbf{x^{(1)}},\mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p} \\ I_2(\phi(\mathbf{x^{(2)}},\mathbf{p})) + \frac{\partial I_2(\phi(\mathbf{x^{(2)}},\mathbf{p}))}{\partial \mathbf{y}} \frac{\partial \phi(\mathbf{x^{(2)}},\mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p} \\ \vdots \\ I_2(\phi(\mathbf{x^{(K)}},\mathbf{p})) + \frac{\partial I_2(\phi(\mathbf{x^{(K)}},\mathbf{p}))}{\partial \mathbf{y}} \frac{\partial \phi(\mathbf{x^{(K)}},\mathbf{p})}{\partial \mathbf{p}} \Delta\mathbf{p} \end{bmatrix} =$$

$$= \begin{bmatrix} I_2(\phi(\mathbf{x^{(1)}},\mathbf{p})) \\ I_2(\phi(\mathbf{x^{(2)}},\mathbf{p})) \\ \vdots \\ I_2(\phi(\mathbf{x^{(K)}},\mathbf{p})) \end{bmatrix} + \begin{bmatrix} \frac{\partial I_2(\phi(\mathbf{x^{(1)}},\mathbf{p}))}{\partial \mathbf{y}} \frac{\partial \phi(\mathbf{x^{(1)}},\mathbf{p})}{\partial \mathbf{p}} \\ \frac{\partial I_2(\phi(\mathbf{x^{(2)}},\mathbf{p}))}{\partial \mathbf{y}} \frac{\partial \phi(\mathbf{x^{(2)}},\mathbf{p})}{\partial \mathbf{p}} \\ \vdots \\ \frac{\partial I_2(\phi(\mathbf{x^{(K)}},\mathbf{p}))}{\partial \mathbf{y}} \frac{\partial \phi(\mathbf{x^{(K)}},\mathbf{p})}{\partial \mathbf{p}} \end{bmatrix} \Delta\mathbf{p} =$$

$$= \mathbf{i}_2(\mathbf{p}) + G(\mathbf{p})\Delta\mathbf{p} \tag{2}$$

The last approximation is useful because the algorithm proposed in "Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization" on each iteration solves for $\Delta\mathbf{p}$ that maximizes $E_{ECC}(\tilde{\mathbf{p}} + \Delta\mathbf{p})$. With approximation (2) the solution has closed form.

We took $\phi(\mathbf{x}, \mathbf{p})$ to be affine. If the transformation corresponds to matrix $\begin{bmatrix} p_0 & p_2 & p_4 \\ p_1 & p_3 & p_5 \end{bmatrix}$, then $\frac{\partial \phi(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}} = \begin{bmatrix} x_1 & 0 & x_2 & 0 & 1 & 0 \\ 0 & x_1 & 0 & x_2 & 0 & 1 \end{bmatrix}$

3.2 Video Inpainting

After image alignment, video inpainting is performed. It consists of two steps.

First step is to perform mosaicing, which will cover all missing static parts of the frames in the video. We join intersecting frames (the frames that have common region) by the parts that overlap. Like that, they will complement each other. To moscaic a pixel, we compute warp matrix from different neighbouring frames to the frame where we are trying to mosaic a unknown pixel. As we have a lot of frames in our video, we will have multiple variants to complete the same region. Ideally, mosaics from different frames should perfectly coincide or be at most some distance one from another. Distance between two pixels was computed based on their colours. However, some incorrect transformation as well as other stability related issues may have occurred during video recording. Also, as was already noted, this method do not help inpaint dynamic moving objects. On the other hand, it is easy to derive a measure to determine how valid a mosaic is – if the variance of a pixel is large enough among frames, we can claim that the mosaic is not valid, usually because the region may not be static.

$$\begin{cases} median(I_{t'}(T_t^{t'} p_t)) & : \quad var(p) \leq T \\ keep\ unpainted & : otherwise \end{cases}$$

where

p_t is missing pixel which we try to inpaint using mosaicing
$T_t^{t'}$ is transform from frame t to frame t'
T – predefined threshold for acceptance

When we accept a pixel to be inpainted we compute its value – color – by taking mean. So we try to perform mosaicing and inpaint a certain pixel of the frame if the variance is small enough. Otherwise we leave the pixel unknown/unpainted and try to inpaint it in further steps.

Second part of video inpainting is motion inpainting. In this part we try to inpaint missing dynamic objects of video frames. To that mean, we compute optical flow using, for example, the Lucas Kanade algorithm. The algorithm is based on the assumption that optical flow is relatively small and constant between two neighboring frames in the neighborhood of a pixel, for which we estimate optical flow. It tries to determine the best motion estimation using least squares. Least squares is useful in such a case because our system is well determined – there might be dozens of pixels which we want to consider in the neighborhood of the current pixel, and the system we construct is over-determined as it has more equations than unknowns. So if we consider a pixel, for which we want to estimate optical flow (find velocity vector $V = \{V_x, V_y\}$) and we have a window around the pixel (pixels in the window are denoted as q_i), we construct the following optical flow equation:

$$Av = b$$

where

$$\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ ... & ... \\ I_x(q_n) & I_y(q_n) \end{bmatrix}$$

$$v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}$$

$$b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ ... \\ -I_t(q_n) \end{bmatrix}$$

I_x – partial derivative of the image I with respect to position x
I_y – partial derivative of the image I with respect to position y
I_t - partial derivative of the image I with respect to time t
So using least squares, we solve the system of equations:

$$A^T Av = A^T b \quad \equiv \quad v = (A^T A)^{-1} A^T$$

Although not the entire image is known – we have missing pixels in some image areas that appeared when we fixed the background during motion smoothing. For the missing pixels, we compute optical flow from its neighbors, gradually moving from the frame regions where we know pixels to the image areas where the pixels are missing. For a missing pixel, we compute its optical flow by weighted averaging of the optical flow of its neighbors:

$$\frac{\sum_{q_t \in H(p_t)} w(p_t, q_t) F_t(q_t))}{\sum_{q_t \in H(p_t)} w(p_t, q_t)}$$

$w(p_t, q_t)$ – contribution of the motion value of neighboring pixel q_t to the motion value of current unknown pixel p_t

$F_{t'}(q_t)$ – motion value of neighboring pixel q_t

$H(p_t)$ – pixels in the window around unknown pixel p_t (in other words, neighboring pixels)

3.3 Peak signal-to-noise ratio

Even though there are no well-known methods to evaluate how good video is stabilized, there are some experimental ways to do that. The current evaluating methods of video stabilization are mainly based on the **PSNR (peek signal-to-noise ratio)**. So, we implemented **PSNR** to compare our results with some other stabilizers.

The term **peak signal-to-noise ratio** (**PSNR**) is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. Because many signals have a very wide dynamic range, (ratio between the largest and smallest possible values of a changeable quantity) the **PSNR** is usually expressed in terms of the logarithmic decibel scale.

Mathematics

For the following implementation, let us assume we are dealing with a standard 2D array of data or matrix. The mathematical representation of the **PSNR** is as follows:

$$ITF = \frac{1}{N_f - 1} \sum_{i=1}^{N_f - 1} PSNR(t)$$

$$PSNR = 20 \log_{10}(\frac{MAX_f}{\sqrt{MSE}})$$

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|f(i,j) - g(i,j))\|^2$$

where

f represents the matrix data of our original image

g represents the matrix data of our degraded image in question

m represents the numbers of rows of pixels of the images and i represents the index of that row

n represents the number of columns of pixels of the image and j represents the index of that column

$\mathbf{MAX_f}$ is the maximum signal value that exists in our original "known to be good" image

ITF is just a mean of PSNRs. We obtained the following value of ITF index:

4 Implementation Pipeline

We decided to implement our video-stabilization system using **OpenCV** methods in **C++** programming language. You can find source code for our project here: https://github.com/MaksymTsapiv/VideoStabilization.git.

We implemented the whole pipeline of video stabilization with the help of OpenCV TwoPassStabilizer with few minor differences from the pipeline that is described in this paper.

Also we implemented Steps 1-3 of the pipeline described in this paper with the help of function findTransformECC.

Additinally, we implemented our own ECC maximization algorithm.

References

[1] Matsushita Y. et al. "Full-frame Video Stabilization". Microsoft Research Asia. (https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/videostabilization$_c vpr$05.*pdf*).

[2] Evangelidis G., Psarakis E. "Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization". IEEE Transactions on Pattern Analysis and Machine Intelligence, Institute of Electrical and Electronics Engineers, 2008. (https://hal.inria.fr/hal-00864385/document).

[3] Bin Chen, Jianjun Zhao and Yi Wang Weapon Science and Technology Dept, NAAU, Yantai, China "Research on Evaluation Method of Video Stabilization" (https://www.atlantis-press.com/article/25858149.pdf)

[4] W. Guilluy, A. Beghdadi and L. Oudre, "A performance evaluation framework for video stabilization methods," 2018 7th European Workshop on Visual Information Processing (EUVIP), 2018, pp. 1-6, doi: 10.1109/EUVIP.2018.8611729. (http://www.laurentoudre.fr/publis/GBO-EUVIP-18.pdf)