Notes for Video Stabilization using L1 optimal camera paths

Ishank Juneja

June 13, 2020

1 Procedure for offline video stabilization

The general pipeline for offline video stabilization consists of the following broad steps-

1. Motion estimation - Find $C(t)$
To stabilize a video stream, we first need to extract the raw noisy and jerky camera motion $C(t)$. Only then can we come up with a stabilized version $P(t)$. For processing on a computer these trajectories will have to be discretized to C_t and P_t respectively.
In general the raw camera trajectory C_t can be a time indexed sequence of parameter vectors. The number of parameters in each vector depend on the choice of motion model for the camera trajectory. In this paper and in most other literature, C_t is taken from a family of 2D motion models.
C_t is computed iteratively using the relation,

$$C_{t+1} = C_t F_{t+1} \implies C_t = F_1 F_2 \dots F_t. \tag{1}$$

Where the 2D linear motion model F_t is obtained from the sequence of images comprising the video $(I_1, I_2, \dots, I_t)$, and the collection of feature points tracked (using some suitable algorithm) $\mathbf{x}$.
Every frame pair (I_{t-1}, I_t) is associated with a linear motion model $F_t(\mathbf{x})$. A certain minimum number of points will need to be present in $\mathbf{x}$ depending on how complex our choice of motion model is. The availability of more than the required number of points will be better since then, a robust set of parameters can then be obtained using regression.
To be precise, here F_t is a time dependent update operation that has been cast as a matrix. For instance, for a 2D affine model, with 6 DOF will be

$$F_t = \begin{pmatrix} a_t & b_t \\ c_t & d_t \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x_t \\ \Delta y_t \end{pmatrix}. \tag{2}$$

For this example, the frame update transforms F_t could equivalently be characterized by a parameter vector $p_t = (a_t, b_t, c_t, d_t, \Delta x, \Delta y)^T$

2. Obtain a stabilized camera trajectory $P(t)$
Once the raw camera trajectory $C(t)$ is obtained, we can process it, using an algorithm of our choice, to obtain a stabilized camera trajectory $P(t)$.
For instance a simplistic method of obtaining such a trajectory could be to replace each data point of the parameter sequence C_t by an average over a suitably sized window w. With $w = 1$, the stabilized trajectory P_t would look like,

$$P_t = \frac{C_{t-1} + C_t + C_{t+1}}{3}. \tag{3}$$

In this paper the authors look at a Linear Programming formulation to obtain P_t. This has been explained in the Section 2.

3. Synthesising stabilized video
Once the stabilized camera path has been obtained, we can move on to synthesizing the stabilized video. This final step is achieved by extracting a crop window from the existing frames (Using a method described later) and then applying a stabilization and retargeting transform B_t. The stabilized camera path P_t can be written as

$$P_t = C_t B_t. \tag{4}$$

Where B_t is the stabilization transform given by $B_t = C_t^{-1} P_t$. As mentioned, B_t serves two purposes - stabilization and retargeting. When B_t is applied to a centered crop window, the output is a stabilized version of the original video with a slightly diminished field of view.

2 Linear Programming Formulation

Taking inspiration from the stable camera trajectories employed by professional videographers, the algorithm finds a best fit ('optimal') camera path composed of a combination of constant, linear and parabolic segments.
The authors justify this choice by stating that professional cinematographic qualities are the supposed aim of video stabilization.
The *best fit* is under the L1 norm. As a result of this choice, the optimization problem can be cast as a linear programming problem.
The *fit* is achieved by minimizing a combination of first, second and third derivatives of the stabilized camera path P_t.

Objective

$$\min_{\{B_t\}_1^n} w_1 |D(P)|_1 + w_2 |D^2(P)|_1 + w_3 |D^3(P)|_1. \tag{5}$$

Where D is the derivative operator and w_1, w_2 and w_3 are scalar weights chosen by the programmer and n, is the number of time steps.

Using the relation 4 and the definition $|D(P)| = \sum_t |P_{t+1} - P_t|$, and assuming that we wish to minimize the sum of the derivatives of every component of P_t, we can derive the following relations -

$$|D(P)| = \sum_t |R_t| \tag{6}$$

$$|D^2(P)| = \sum_t |R_{t+1} - R_t| \tag{7}$$

$$|D^3(P)| = \sum_t |R_{t+2} - 2R_{t+1} + R_t|. \tag{8}$$

Where R_t is the residual defined by

$$R_t = F_{t+1}B_{t+1} - B_t. \tag{9}$$

The objective 5 requires a search over the space of possible stabilization transforms. B_t can be reduced to a parameter vector p_t similar to how F_t was reduced to one in an earlier example.
Under this new framework, the residual R_t becomes,

$$|R_t(p)| = |M(F_{t+1}, p_{t+1}) - p_t|, \tag{10}$$

Where $M(F_{t+1}, p_{t+1})$ represents an operator giving a result equivalent to the matrix product $F_{t+1}B_{t+1}$. For the proceeding discussion, consider p_t to be an N dimensional parameter vector.

To formulate the problem as a linear programming problem (LPP), the non linear objective specified in 5 must be reformulated. Conversion from an objective containing an absolute value to a linear objective is described in [1].
For example the conversion of a residual of $|D(P)|_1$ at time instant t would be,

$$-e_t^1 \leq |R_t(p)| = |M(F_{t+1}, p_{t+1}) - p_t| \leq e_t^1, \tag{11}$$

where e_t^1 is an N dimensional vector, with all entries as positive in line with the construction illustrated in [1], and the inequalities are component wise. Similarly there would exist slack vectors e_t^2, e_t^3 for residuals associated with the terms $|D^2(P)|_1$ and $|D^3(P)_1|$ of the objective 5.
On introducing the above family of inequalities such as the one in 11, the objective 5 becomes,

$$\min_p c^T e. \tag{12}$$

Where p is the entire N dimensional parameter space over the span of n frames - $(p_1, \ldots, p_N)$. e is the collection (e^1, e^2, e^3) with each e^i representing the collection of slack vectors for one of the three terms $(|D^1(P)|_1, |D^2(P)|_1, |D^3(P)_1|)$ across all time steps n - $(e_1^i, \ldots, e_n^i)$. c is a suitably chosen vector/matrix of repeated entries of the weights w_1, w_2, w_3 such that the objective remains consistent with the one in 5.

Constraints in the LPP

In the absence of any constraints on the objective 12, the optimal transform would simply be $P_t = I$, with all residuals as 0. To get a useful result, we must introduce meaningful constraints.

Proximity Constraint

Path P_t should preserve the intent of path C_t. If the original path C_t contained a zoom in/out, then P_t should follow the same smoothly. Assuming the 6 DOF affine transform model assumed in 2 is being used for both C_t and P_t, the following linear constraints can enforce this,

$$\begin{aligned} 0.9 \leq a_t, d_t &\leq 1.1 && (13)\\ -0.1 \leq b_t, c_t &\leq 0.1 && (14)\\ -0.05 \leq b_t + c_t &\leq 0.05 && (15)\\ -0.1 \leq a_t - d_t &\leq 0.1. && (16)\end{aligned}$$

Here the constants have been chosen depending on the role of a certain component in the 6 DOF model.
Since all these constants are a part of the parameterization p_t, this can be represented in a more compact form as

$$\text{lb} \leq U p_t \leq \text{ub}, \tag{17}$$

where U specifies the linear combinations by picking out parameters from p_t.

Inclusion Constraint

As mentioned earlier, a crop window must be selected out of every image frame I_t, so that the transform B_t can be applied to it to obtained the stabilized frame. The inclusion constraint says that even after the transform B_t (or its proxy p_t) is applied, the resulting corner coordinates should be contained in the original frame's w, h rectangle.
Let the 4 corners of this rectangular crop window be specified by $c_i = (c_i^x, c_i^y)$ for $i = 1, \ldots, 4$. Then the constraint becomes

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 & c_i^x & c_i^y & 0 & 0 \\ 0 & 1 & 0 & 0 & c_i^x & c_i^y \end{pmatrix} p_t \leq \begin{pmatrix} w \\ h \end{pmatrix} \tag{18}$$

2.0.1 Saliency Constraint

Certain important (salient) points should ideally never be excluded from the choice of crop window. These could include a person's face for instance. Video stabilization could be achieved without this constraint, but adding it would enable *directed* video stabilization. This has been deferred to the next update so that initial code can get up and running.

3 References

[1] LPP with absolute value in the objective