Published in Image Processing On Line on 2017–11–26.
Submitted on 2017–05–08, accepted on 2017–10–23.
ISSN 2105–1232 © 2017 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2017.209

Comparison of Motion Smoothing Strategies for Video Stabilization using Parametric Models

Javier Sánchez

Centro de Tecnologías de la imagen (CTIM), Department of Computer Science,
University of Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, Spain
jsanchez@ulpgc.es

Communicated by Laurent Oudre *Demo edited by* Javier Sánchez

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:
Javier Sánchez and Jean-Michel Morel, "Motion Smoothing Strategies for Video Stabilization", *SIAM Journal on Imaging Sciences*, in press, 2017.

Abstract

This paper is devoted to a rigorous implementation and to an exhaustive comparison of video stabilization techniques. These techniques aim at removing the undesirable effects of camera shake. They first estimate a global transform from frame to frame, which can be a translation, a similarity, an affine map or a homography. This generates a signal that can be smoothed and used to compensate the noisy transform signal. This paper compares all classic smoothing methods and their boundary conditions. It also analyzes two algorithms to crop the video after stabilization. The stabilization results are displayed in a scale-space form permitting to extract valuable information about ego-motion such as its frequencies and its general tendencies.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of this article[1]. Compilation and usage instructions are included in the README.txt file of the archive.

Keywords: video stabilization; motion compensation; motion smoothing; homography; scale-space

[1] https://doi.org/10.5201/ipol.2017.209

Javier Sánchez, *Comparison of Motion Smoothing Strategies for Video Stabilization using Parametric Models*, Image Processing On Line, 7 (2017), pp. 309–346. https://doi.org/10.5201/ipol.2017.209

1 Introduction

Video stabilization is the process of estimating the undesired camera motion and warping the images to compensate for it. This is useful, for example, for videos taken with hand–held cameras, which are often jittery.

Video stabilization has four main steps: motion estimation, motion compensation or smoothing, image warping and, finally, cropping to remove the empty regions that appear at the border of the frames. A good homographic stabilization also delivers a multi–scale eight parameter temporal stabilization signal, which we call a *stabilization scale–space*. This scale–space yields a reliable estimation of the main intrinsic properties of the video, including a multi–scale understanding of camera ego-motion parameters such as pitch, roll, yaw and zoom, and of their main frequencies in a sliding window.

We shall first review briefly the main techniques that have been proposed for this complex problem. 3D stabilization techniques are the most sophisticated ones as they reconstruct the camera trajectory through external camera calibration and an estimation of the 3D structure of the scene [7], [30], [52], [36], [15], or [61]. Several methods also combine 2D and 3D strategies [35], [15], for instance, by estimating only the epipolar geometry or involving extra information by depth cameras [38]. Here we shall only deal with simpler 2D strategies that assume no external information about the 3D camera path. An estimate of the motion between successive frames is the only used information. These can involve optical flow [40] or instead estimate a parametric planar transform from frame to frame, such as a translation, a similarity, an affinity, or a homography.

Why homographies? Homographies are the adequate model for the effect on the image of a moving pinhole camera motion filming remote and steady objects. They also are the right model for the effect on the image of a rotation of the camera around its optical center. This model is widely used for creating panoramas.

Motion estimation: Classic methods [20, 45, 46, 60] either use correlation techniques or track salient points such as Harris corners [21] using, e.g., a KLT tracker [42, 51]. Pyramidal structures are typically used for estimating large displacements [43, 18, 48].

When estimating a parametric deformation model, affinities are often preferred to homographies [20, 34, 43], as they avoid introducing important deformations. Even more restrictive parameterizations such as translations [27, 60] or similarities [45] are sometimes recommended. An alternative is estimating a 3D camera rotation model [13, 48], which nevertheless requires the estimation of a focal distance. When dealing for example with strong camera jitter combined with a rolling shutter, it may be necessary to estimate a set of transformations for each frame instead of a single parametric model [17, 35, 39, 23]. Parametric motion estimation can also be based on smoothed optical flows [40]. This is, however, computationally expensive. The use of optical flow has been alleviated in [37] using a mesh of motion vector candidates instead of the full optical flow. These motion candidates are obtained through FAST features [58] and a KLT tracker. In any case, these methods rely on parametric models for initializing the motion estimates.

In this work, our focus is not on the image matching method. In the code and online demo, we use a classic feature–based technique [44], which relies on SIFT features [41], and a direct method [49], which implements the *inverse compositional algorithm* [4, 3]. Direct methods are usually faster but more sensitive to brightness changes and outliers. Feature–based methods depend on the type of selected features. They are typically more sensitive to noise and motion blur, but allow to estimate stronger deformations [56].

Motion compensation. Motion compensation estimates a transformation for each frame that reduces the effect of camera shake. The *compositional approach* [45, 20] compensates directly the images with respect to a reference frame, which makes sense if the camera is static. For smoothing arbitrary camera motions Gaussian smoothing is recommended [46, 43, 25] but robust L^1 regularization strategies have also been proposed [18, 40] to maximize the size of the cropping window. Another way is to approximate the smooth camera path by fitting a polynomial curve [8] or by segmenting the camera path to define an adaptive smoothing.

Image warping and post–processing tasks. In the final step, the images are warped according to the calculated smoothed motion. This produces empty regions at the border of the images. *Crop&Zoom* is the most widely used strategy for eliminating these empty regions [17]. A more sophisticated alternative is *video completion*, which fills these regions by inpainting from other frames [34, 62, 9, 2], sometimes guided by optical flow [43]. These methods are obviously risky and impose delicate dynamic blending strategies [57, 6]. On the other hand, *motion blur* is revealed by stabilization [29] and requires again sophisticated strategies to transfer sharp information from other frames [43].

Scale-space and the frequency analysis. Scale-space theory [63, 28, 31, 32, 32] amounts to analyze a signal or an image by convolving it with Gaussians of growing standard deviations. Scale-space theory is the subject of several recent books [53, 33, 19]. It has been proposed to stabilize the video gray scale distribution and to compensate flickering effects [11, 50]. Several works have used parametric models between two successive frames in the video for the purpose of video indexing, camera motion characterization, or ego-motion classification, such as [5] or [26]. The scale-space analysis that we propose for video stabilization can also be used for other related tasks, such as activity classification and detection [24], [54], [47].

Our plan is as follows: Section 2 introduces the general video stabilization framework and the taxonomy of the motion compensation strategies, divided in the compositional and additive groups. It also describes the crop and zoom strategy to mask empty spaces in the stabilization video by cropping adaptively each frame. Section 3 compares the different compensation alternatives and deduces from this choice a stabilization scale-space. Experiments demonstrate how it can be used to compute reliable ego–motion characteristics. This section also describes the online experimental setup associated with this paper. Conclusions are presented in Section 4.

2 Video Stabilization

Algorithm 1 shows a general framework for video stabilization, comprising four main processes: The first one computes the transformations between successive frames; the second step smoothes the trajectory; the third implements some post–processing tasks, such as the crop&zoom; and the fourth warps the images to compensate the motion in each frame. Computing the frame to frame transformations, $\{\mathbf{H}_{i,i+1}\}$ is the slowest process.

The last step consists in applying the smoothing transformations as

$$\mathbf{I}'_i(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}'_i\mathbf{x}). \tag{1}$$

By abuse of notation, here $\mathbf{H}'_i\mathbf{x}$ represents the normalized point in image coordinates, i.e., $\mathbf{H}'_i\mathbf{x} = (x'_1/x'_3, x'_2/x'_3, 1)$, and $\mathbf{I}_i(\mathbf{H}'_i\mathbf{x}) = \mathbf{I}_i(x'_1/x'_3, x'_2/x'_3)$.

Algorithm 1: Video stabilization framework

```
Input  : $\{\mathbf{I}_i\}$
Output: $\{\mathbf{I}'_i\}$
Motion_estimation($\{\mathbf{I}_i\}$,$\{\mathbf{H}_{i,i+1}\}$)
Motion_smooting($\{\mathbf{H}_{i,i+1}\}$,$\{\mathbf{H}'_i\}$)
Post_processing($\{\mathbf{H}'_i\}$)
Image_warping($\{\mathbf{I}_i\}$,$\{\mathbf{I}'_i\}$,$\{\mathbf{H}'_i\}$)
```

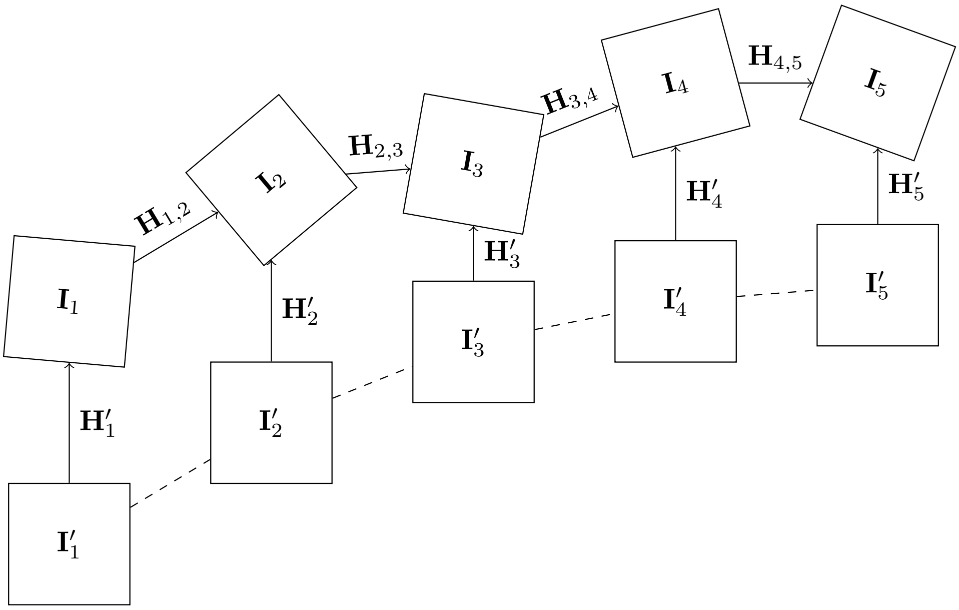


Figure 1: Illustration of the motion smoothing process. $\{\mathbf{I}_i\}_{i=1,..N}$ is the initial image sequence, $\{\mathbf{H}_{i,i+1}\}_{i=1,..N-1}$ are the computed transformations between consecutive images. $\{\mathbf{I}'_i\}_{i=1,..N}$ is the stabilized sequence and $\{\mathbf{H}'_i\}_{i=1,..N}$ are the stabilizing transformations.

2.1 Motion Estimation

The motion estimation step receives a sequence of images, $\{\mathbf{I}_i\}$, and computes a set of transformations, $\{\mathbf{H}_{i,i+1}\}$, between successive frames, i and $i+1$, as illustrated in Figure 1.

The pixels of the images are related by the photometry consistency principle $\mathbf{I}_i(\mathbf{x}) = \mathbf{I}_{i+1}(\mathbf{H}_{i,i+1}\mathbf{x})$, where $\mathbf{x} = (x, y, 1)$ is the pixel position. These matrices may be, for example, any of the transformations given in Table 1, each of them depending on a set of parameters. These parameterizations are typical of direct methods [56]. Note that the matrices and points are expressed in homogeneous coordinates, so the image positions are obtained after normalizing by the third component. In the following, we will use matrix notation or its corresponding parameterization indistinctly.

In this work, we use two motion estimation techniques. On the one hand, we use a feature–based technique [44], which is based on SIFT features and RANSAC. In its current implementation, it only estimates homographies between frames. The benefit of using SIFT and RANSAC is that it can compute stronger deformations between the images with higher accuracy. In general, the homographies are representatives of the background motion, where the number of inliers is usually bigger. On the other hand, we use a direct method [49] implementing the *inverse compositional algorithm* [4, 3], which is an improvement of the Lucas-Kanade Algorithm [42]. It can compute any of the transformations in Table 1. This is faster than the previous method but more sensitive to outliers. Their implementations are available in IPOL.

Transform	Parameters ($\mathbf{p}$)	$\mathbf{H}$
Translation	(t_x, t_y)	$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$
Euclidean	(t_x, t_y, θ)	$\begin{pmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{pmatrix}$
Similarity	(t_x, t_y, a, b)	$\begin{pmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \\ 0 & 0 & 1 \end{pmatrix}$
Affinity	$(t_x, t_y, a_{11}, a_{12}, a_{21}, a_{22})$	$\begin{pmatrix} 1+a_{11} & a_{12} & t_x \\ a_{21} & 1+a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$
Homography	$(h_{11}, h_{12}, h_{13}, \cdots, h_{32})$	$\begin{pmatrix} 1+h_{11} & h_{12} & h_{13} \\ h_{21} & 1+h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$

Table 1: Typical planar transformations with their parameters and homogeneous matrix representations.

2.2 Motion Compensation

The objective of motion compensation is to obtain a new set of transformations that effectively remove the undesired background motion. Given the transition background transforms $\{\mathbf{H}_{i,i+1}\}$, which we assume henceforth to be correct, the aim is to obtain the stabilization deformations, as in Figure 1.

The motion smoothing step obtains a new set of transformations, $\{\mathbf{H}'_i\}$, from the computed motions $\{\mathbf{H}_{i,i+1}\}$. To that purpose, it applies a Gaussian convolution to the time series of transformations. We discuss the smoothing alternatives in the following section. $\{\mathbf{H}'_i\}$ are the transformations that must be applied to $\{\mathbf{I}_i\}$ to eventually obtain the stabilized images $\{\mathbf{I}'_i\}$.

The strategies for computing $\{\mathbf{H}'_i\}$ can be classified in *Compositional* and *Additive* methods, depending on whether they are based on the composition of transformations or the integration of their coefficients, respectively.

2.2.1 Compositional Methods

Pure compositional approach. Compositional approaches are relevant for fixed cameras, for which a reference background frame can be established.

Definition 1. *We define the* ***compositional*** *approach as the process of calculating the transformations that compensate the images with respect to a reference frame. The transformations are obtained by composing the relative motions between successive frames by*

$$\mathbf{H}'_i = \mathbf{H}_{1,i} := \prod_{j=2}^{N} \mathbf{H}_{j-1,j} = \mathbf{H}_{i-1,i}\mathbf{H}_{i-2,i-1}\cdots\mathbf{H}_{2,3}\mathbf{H}_{1,2}. \tag{2}$$

This is the technique used in the first works [20, 45] and it was usually combined with the construction of mosaics. It is compatible with real–time processing, since each single frame can be directly compensated. A simple iterative procedure is given in Algorithm 2.

Algorithm 2: Compositional approach

```
Input  : $\{\mathbf{H}_{i,i+1}\}$
Output: $\{\mathbf{H}'_i\}$
$\mathbf{H}'_1 \leftarrow \mathbf{Id}$ // identity matrix
for $i \leftarrow 2$ to $N$ do
    $\mathbf{H}'_i \leftarrow \mathbf{H}_{i-1,i}\mathbf{H}'_{i-1}$
```

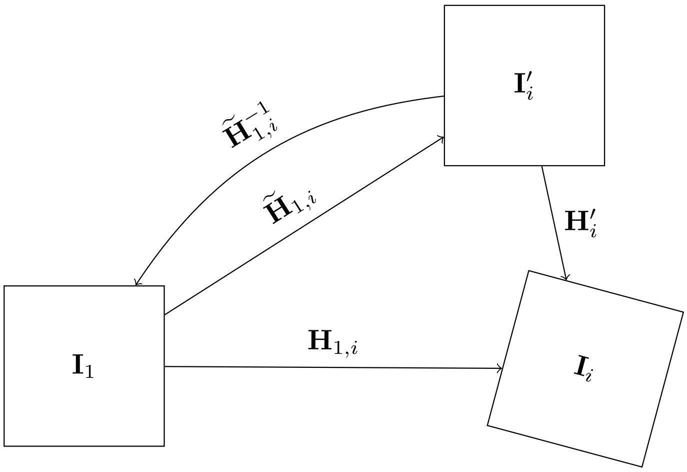


Figure 2: Relation between the reference frame, $\mathbf{I}_1$, the actual frame, $\mathbf{I}_i$, and the compensated frame, $\mathbf{I}'_i$.

If the camera is moving, then the compositional approach is very restrictive as it considerably reduces the field of view of the video. It is necessary to smooth the transformations in order to respect the main camera motion.

Compositional smoothing. The idea of compositional smoothing is illustrated in Figure 2. The objective is to find the transformations $\{\mathbf{H}'_i\}$ that relate $\{\mathbf{I}_i\}$ to $\{\mathbf{I}'_i\}$, so that the smooth transformations $\{\widetilde{\mathbf{H}}_{1,i}\}$ do not include the high frequency motion. The compositional smoothing approach is the process of calculating the transformations that simulate a smooth registration of the current frame with respect to a given reference frame.

Definition 2. *We start from the compositional transition homographies $\mathbf{H}_{1,i}$ defined by (2) and, more precisely, from their eight matrix elements $\mathbf{H}_{1,i}(p,q)$, where p and q are the row and column index respectively. Consider the smooth homographies $\widetilde{\mathbf{H}}_{1,i}$, whose elements $\widetilde{\mathbf{H}}_{1,i}(p,q)$ are obtained through a convolution with a discrete Gaussian function of the series $\mathbf{H}_{1,i}(p,q)$ as*

$$\widetilde{\mathbf{H}}_{1,i}(p,q) := (G_\sigma * \mathbf{H}_{1,\cdot})_i(p,q) = \sum_{j\in\mathcal{N}_i} G_\sigma(j-i)\mathbf{H}_{1,j}(p,q), \tag{3}$$

with $G_\sigma(x) := We^{-\frac{x^2}{2\sigma^2}}$, $\mathcal{N}_i = \{j : i-k \leq j \leq i+k\}$, and $W := 1/\sum_{i=-k}^{i=k} e^{-\frac{i^2}{2\sigma^2}}$ a normalizing coefficient. The ***compositional smoothing*** *approach is defined by the following rectifying transformations and image stabilized sequence*

$$\mathbf{H}'_i := \mathbf{H}_{1,i}\widetilde{\mathbf{H}}^{-1}_{1,i}\ ; \quad \mathbf{I}'_i(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}'_i\mathbf{x}\}. \tag{4}$$

Note that we are using the compositions with respect to the first frame and each element is obtained as a weighted average of the elements of the composed homographies.

Alternatively to Definition 2, the smoothing can be realized on the parameters of the transformations, $\mathbf{p}$, as detailed in Table 1. For instance, if we use an Euclidean transformation, the parameters to be smoothed are the translation, (t_x, t_y), and the angle of rotation, θ. Since they are treated separately, the smoothing has a geometrical meaning.

However, for affinities and homographies, the smoothing is less intuitive and can be formalized with the transformation matrices. In the case of similarities, it is possible to use another parametrization based on the translation (t_x, t_y), the scale factor λ, and the angle of rotation θ. These can be easily obtained from the parameters in Table 1, and, again, the smoothing has a geometrical meaning.

Algorithms 3 and 4 implement this method and the corresponding discrete Gaussian convolution.

Algorithm 3: Compositional smoothing

```
Input  : ${\mathbf{H}}, \sigma, bc$
Output: ${\mathbf{H}'}$
// compute transformations with respect to the initial reference frame
$\mathbf{H}^c_{1,1} \leftarrow \mathbf{H}_{1,1}$
for $i \leftarrow 2$ to $N$ do
    $\mathbf{H}^c_{1,i} \leftarrow \mathbf{H}_{i-1,i}\mathbf{H}^c_{1,i-1}$
// convolve with a Gaussian kernel; Eq. (3)
for $i \leftarrow 1$ to $N$ do
    call Gaussian convolution(${\mathbf{H}^c}$, $\widetilde{\mathbf{H}}_{1,i}$, $i$, $\sigma$, $bc$)
for $i \leftarrow 1$ to $N$ do
    $\mathbf{H}'_i \leftarrow \mathbf{H}^c_{1,i}\widetilde{\mathbf{H}}^{-1}_{1,i}$
```

Note that the reference frame must also be stabilized. For this reason, we must introduce the identity matrix at the beginning ($\mathbf{H}_{1,1} = \mathbf{Id}$) and also convolve it with a Gaussian function. The method explained in [45] follows this scheme, although it combines both the motion estimation and smoothing steps.

Boundary conditions for compositional smoothing. The Gaussian filtering of the sequence of homographies requires specifying how to deal with the temporal boundary conditions at the beginning and end of the time interval. When the smoothing radius goes beyond the limits of the image sequence, several strategies can be envisaged.

Definition 3. *Let $\{\mathbf{H}_{1,i}\}$ be a set of transformations from each frame i to the reference frame 1 in a time interval between 1 and N. We define **constant boundary conditions** by*

$$\mathbf{H}_{1,j} := \begin{cases} \mathbf{H}_{1,1} & \text{if } j < 1 \\ \mathbf{H}_{1,N} & \text{if } j > N \end{cases}. \tag{5}$$

*We define **Neumann boundary conditions** in the range $[-N+1, 2N-1]$ by*

$$\mathbf{H}_{1,j} := \begin{cases} \mathbf{H}_{1,-j+1} & \text{if } j < 1 \\ \mathbf{H}_{1,2N-j} & \text{if } j > N \end{cases}. \tag{6}$$

Algorithm 4: Gaussian convolution

```
Input  : $\{\mathbf{H}\}, i, \sigma, bc$
Output: $\widetilde{\mathbf{H}}_i$
$radius \leftarrow 3\sigma$
if $radius > N$ then
    $radius \leftarrow N$
// Gaussian convolution in each parameter separately
forall the $(p, q)$ do
    $average \leftarrow 0$
    $sum \leftarrow 0$
    for $j \leftarrow i - radius$ to $i + radius$ do
        $value \leftarrow 0$
        if $j < 1$ then
            switch $bc$ do
                case $CONSTANT\_BC$
                    $value \leftarrow \mathbf{H}_{1,1}(p, q)$
                case $NEUMANN\_BC$
                    $value \leftarrow \mathbf{H}_{1,-j+2}(p, q)$
                case $DIRICHLET\_BC$
                    $value \leftarrow 2\mathbf{H}_{1,1}(p, q) - \mathbf{H}_{1,-j+2}(p, q)$
        else if $j > N$ then
            switch $bc$ do
                case $CONSTANT\_BC$
                    $value \leftarrow \mathbf{H}_{1,N}(p, q)$
                case $NEUMANN\_BC$
                    $value \leftarrow \mathbf{H}_{1,2N-j}(p, q)$
                case $DIRICHLET\_BC$
                    $value \leftarrow 2\mathbf{H}_{1,N}(p, q) - \mathbf{H}_{1,2N-j}(p, q)$
        else
            $value \leftarrow \mathbf{H}_{1,j}(p, q)$
        $gauss \leftarrow e^{-\frac{(j-i)^2}{2\sigma^2}}$
        $average \leftarrow average + gauss \times value$
        $sum \leftarrow sum + gauss$
    $\widetilde{\mathbf{H}}_i(p, q) \leftarrow average/sum$
```

We define ***Dirichlet boundary conditions*** *in the range* $[-2N+2, 3N-1]$ *by*

$$\mathbf{H}_{1,j} := \begin{cases} \mathbf{H}_{1,-j+2} + 2\mathbf{H}_{1,1} - 2\mathbf{H}_{1,N} & \text{if } -2N+2 \leq j < -N+1 \\ 2\mathbf{H}_{1,1} - \mathbf{H}_{1,-j+1} & \text{if } -N+1 \leq j < 1 \\ 2\mathbf{H}_{1,N} - \mathbf{H}_{1,2N-j} & \text{if } N \leq j < 2N \\ \mathbf{H}_{1,j-2N+1} + 2\mathbf{H}_{1,N} - 2\mathbf{H}_{1,1} & \text{if } 2N \leq j < 3N \end{cases}. \tag{7}$$

Constant boundary conditions replicate the first and last transformations beyond the scope of the video. A consequence of this boundary condition is that, for large values of σ, the initial and final frames of the video are allowed to move from their original positions. In the Neumann boundary conditions, the derivative of the homographies is constant in the boundaries. These conditions are accomplished by reflecting the values on both ends. Again, this allows the initial and final frames to move from their original positions. The goal of Dirichlet boundary conditions is, by an odd reflection across the temporal boundaries, to ensure that the initial and final frames do not move, namely to obtain at the boundaries the equivalent to the original matrices. With these boundary conditions, the first and last frames will coincide with the original video. Note that, in practice, it is sufficient to take into account the two inner conditions because they already contain the video sequence.

Compositional local smoothing. The compositional local smoothing approach is the process of calculating the transformations that create a smooth motion path of the video with respect to a given reference frame. The main difference with respect to the previous approach is that the transformations are smoothed locally and then composed with the original transformations.

Definition 4. *Given the following convolution with a Gaussian function,*

$$\widetilde{\mathbf{H}}_{i,i+1}(p,q) := (G_\sigma * \{\mathbf{H}\})_i(p,q) = \sum_{j\in\mathcal{N}_i} G_\sigma(i-j)\mathbf{H}_{j,j+1}(p,q), \tag{8}$$

the ***compositional local smoothing*** *is defined by the rectifying transformations*

$$\begin{aligned}\mathbf{H}'_i := \prod_{j=1}^{i}\left(\mathbf{H}_{j,j+1}\widetilde{\mathbf{H}}^{-1}_{j,j+1}\right) = \left(\mathbf{H}_{i,i+1}\widetilde{\mathbf{H}}^{-1}_{i,i+1}\right)\left(\mathbf{H}_{i-1,i}\widetilde{\mathbf{H}}^{-1}_{i-1,i}\right)\cdots \\ \cdots\left(\mathbf{H}_{2,3}\widetilde{\mathbf{H}}^{-1}_{2,3}\right)\left(\mathbf{H}_{1,2}\widetilde{\mathbf{H}}^{-1}_{1,2}\right).\end{aligned} \tag{9}$$

The stabilized image sequence is defined by (1), $\mathbf{I}'_i(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}'_i\mathbf{x}\}$.

This relation is similar to the compositional smoothing approach in Figure 2, where the new matrix represents the desired smooth local increment. This method is mentioned in [43]. The expression $\mathbf{H}_{i,i+1}\widetilde{\mathbf{H}}^{-1}_{i,i+1}$ removes the original local shake and introduces a smooth increment.

The steps are explained in Algorithm 5 and the Gaussian filtering is given in Algorithm 6. The first matrix must be initialized to the identity matrix: $\mathbf{H}'_0 \leftarrow \mathbf{Id}$.

```
Algorithm 5: Compositional local smoothing
  Input  : $\{\mathbf{H}\}, \sigma, bc$
  Output: $\{\mathbf{H}'\}$
  // convolve with a Gaussian kernel; Eq. (3)
  for $i \leftarrow 1$ to $N$ do
      call Local Gaussian convolution($\{\mathbf{H}\}$, $\widetilde{\mathbf{H}}_i$, $i$, $\sigma$, $bc$)
  for $i \leftarrow 1$ to $N-1$ do
      $\mathbf{H}'_i \leftarrow (\mathbf{H}_{i-1,i}\widetilde{\mathbf{H}}^{-1}_{i-1,i})\mathbf{H}'_{i-1}$
```

Boundary conditions for compositional local smoothing. The boundary conditions for the compositional local smoothing approach are slightly different from the previous scheme, because the transformations are increments between consecutive images. In this case, we have the following conditions:

Algorithm 6: Local Gaussian convolution

```
Input  : $\{\mathbf{H}\}, i, \sigma, bc$
Output: $\widetilde{\mathbf{H}}_i$
$radius \leftarrow 3\sigma$
if $radius > N$ then
    $radius \leftarrow N$
// Gaussian convolution in each parameter separately
forall the $(p, q)$ do
    $average \leftarrow 0$
    $sum \leftarrow 0$
    for $j \leftarrow i - radius$ to $i + radius$ do
        $value \leftarrow 0$
        if $j < 1$ then
            switch $bc$ do
                case $CONSTANT\_BC$
                    $value \leftarrow 0$
                case $NEUMANN\_BC$
                    $value \leftarrow \mathbf{H}^{-1}_{-j+1,-j+2}(p, q)$
                case $DIRICHLET\_BC$
                    $value \leftarrow \mathbf{H}_{-j,-j+1}(p, q)$
        else if $j > N$ then
            switch $bc$ do
                case $CONSTANT\_BC$
                    $value \leftarrow 0$
                case $NEUMANN\_BC$
                    $value \leftarrow \mathbf{H}^{-1}_{2N-j,2N-j+1}(p, q)$
                case $DIRICHLET\_BC$
                    $value \leftarrow \mathbf{H}_{2N-j,2N-j+1}(p, q)$
        else
            $value \leftarrow \mathbf{H}_{j,j+1}(p, q)$
        $gauss \leftarrow e^{-\frac{(j-i)^2}{2\sigma^2}}$
        $average \leftarrow average + gauss \times value$
        $sum \leftarrow sum + gauss$
    $\widetilde{\mathbf{H}}_i(p, q) \leftarrow average/sum$
```

Definition 5. *Let* **Id** *be the identity matrix.*

Constant boundary conditions *are defined as* $\mathbf{H}_{j,j+1} := \mathbf{Id}$ *if* $j < 1$ *or* $j > N$*.*

Neumann boundary conditions *are defined in the range* $[-N + 2, 2N - 1]$ *as*

$$\mathbf{H}_{j,j+1} := \begin{cases} \mathbf{H}^{-1}_{-j+1,-j+2} & \textit{if } j < 1 \\ \mathbf{H}^{-1}_{2N-j,2N-j+1} & \textit{if } j > N \end{cases}. \tag{10}$$

Dirichlet boundary conditions *in the range* $[-N+1, 2N-1]$ *are defined by*

$$\mathbf{H}_{j,j+1} := \begin{cases} \mathbf{H}_{-j,-j+1} & \text{if } j < 1 \\ \mathbf{H}_{2N-j,2N-j+1} & \text{if } j > N \end{cases}. \tag{11}$$

The main problem of the previous approaches is that the composition of transformations from a given frame successively accumulates errors. One way to reduce this effect is to set the coordinate system in each frame and perform the convolution in a local neighborhood. We envisage two alternatives: in the first strategy, the matrices are referenced to the current frame and the components of the matrices are smoothed similarly to the previous approaches; in the second strategy, a set of points is selected in the current frame and their positions are tracked in the neighboring frames by applying the transition transforms. In that case, the trajectories of these points are smoothed and the resulting points are used to calculate the rectifying transformation.

Local matrix–based smoothing. This method was proposed in [43] and is presented in Figure 3. The reference system is centered in the current frame, and the smoothing is carried out with a set of transformations around a temporal neighborhood. These transformations need to be related to this central frame, so the composition is given in both temporal directions.

Definition 6. *Consider the following compositions with the previous transformations from the current frame*

$$\mathbf{H}_{i,j:j<i} = \prod_{l=j}^{i-1} \mathbf{H}_{l+1,l} = \mathbf{H}_{j+1,j}\mathbf{H}_{j+2,j+1}\cdots\mathbf{H}_{i-1,i-2}\mathbf{H}_{i,i-1}, \tag{12}$$

with $\mathbf{H}_{l+1,l} = \mathbf{H}_{l,l+1}^{-1}$, *and the composition with the following frames as*

$$\mathbf{H}_{i,j:j>i} = \prod_{l=i}^{j} \mathbf{H}_{l,l+1} = \mathbf{H}_{j-1,j}\mathbf{H}_{j-2,j-1}\cdots\mathbf{H}_{i-1,i}\mathbf{H}_{i,i+1}. \tag{13}$$

in a neighborhood, $\mathcal{N}_i = \{j : i-k \leq j \leq i+k\}$, *around frame* i. *Define the convolution with a Gaussian function in this interval by*

$$\widehat{\mathbf{H}}_i(p,q) := \sum_{j=i-k}^{i+k} G_\sigma(i-j)\mathbf{H}_{i,j}(p,q). \tag{14}$$

with $\mathbf{H}_{i,i} := \mathbf{Id}$. *Then the* ***local matrix–based*** *stabilization is defined by the rectifying transformations* $\mathbf{H}'_i := \widehat{\mathbf{H}}_i^{-1}$ *and the stabilized image sequence by* (1), $\mathbf{I}'_i(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}'_i\mathbf{x}\}$.

The boundary conditions for this approach are identical to those of the compositional smoothing approach, so that Definition 3 is valid.

This approach is more stable and works better for more complex transformations, such as affinities and homographies. The steps of the local matrix-based smoothing are detailed in Algorithm 7.

One of the clear benefits of this approach is that the influence of errors, that may have been produced during the motion estimation process, is limited to a region given by the temporal window in (14). As a consequence, the method is more stable and may recover from errors during the compensation. In the previous techniques, once an error is introduced in the motion of an image, it is propagated to the rest of frames.

This idea is also used in [48] for the smoothing of 3D camera rotation matrices. In that case, it also compensates for the rolling shutter effects.

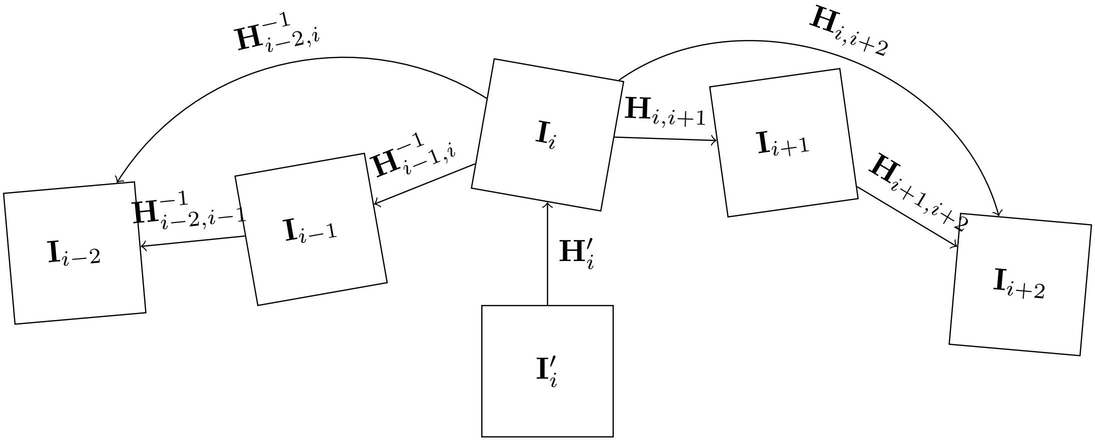


Figure 3: Local matrix-based smoothing.

Algorithm 7: Local matrix-based smoothing

```
Input  : ${\mathbf{H}}, \sigma, bc$
Output: ${\mathbf{H}'}$
$radius \leftarrow 3\sigma$
if $radius > N$ then
    $radius \leftarrow N$
for $i \leftarrow 1$ to $N$ do
    $t_1 \leftarrow i - radius$
    $t_2 \leftarrow i + radius$
    if $t_1 < 1$ then
        $t_1 \leftarrow 1$
    if $t_2 > N$ then
        $t_2 \leftarrow N$
    // compute backward transformations
    if $i > 1$ then
        $\mathbf{H}^c_{i-1} \leftarrow \mathbf{H}^{-1}_{i-1,i}$
        for $j \leftarrow i - 2$ to $t_1$ do
            $\mathbf{H}^c_j \leftarrow \mathbf{H}^{-1}_{j,j+1}\mathbf{H}^c_{j+1}$
    // introduce the identity matrix
    $\mathbf{H}^c_i \leftarrow \mathbf{Id}$
    // compute forward transformations
    if $i < N$ then
        $\mathbf{H}^c_{i+1} \leftarrow \mathbf{H}^{-1}_{i,i+1}$
        for $j \leftarrow i + 2$ to $t_2$ do
            $\mathbf{H}^c_j \leftarrow \mathbf{H}_{j-1,j}\mathbf{H}^c_{j-1}$
    // convolve with a Gaussian kernel; Eq. (3)
    call Gaussian convolution(${\mathbf{H}^c}$, $\widetilde{\mathbf{H}}_i$, $i$, $\sigma$, $bc$)
    $\mathbf{H}'_i \leftarrow \widetilde{\mathbf{H}}^{-1}_i$
```

Local point–based smoothing. This variant proposes a more intuitive way to process the video by smoothing point trajectories instead of matrices. The idea is to select several points in the current frame and track them in its neighborhood using the estimated homographies. A Gaussian convolution in each path will provide an average set of points. Then, a homography can be computed from the averaged points. The local point–based smoothing approach is the smoothing process based on the trajectory of a set of points in a neighborhood around each frame.

If we want to compute a homography —eight parameters— we need to select, at least, four points in each frame, $\{\mathbf{x}_i^p\}^{p=1,\dots,4}$. These points are then projected forwards and backwards as $\mathbf{x}_{i+1}^p := \mathbf{H}_{i,i+1}\mathbf{x}_i^p$ and $\mathbf{x}_{i-1}^p := \mathbf{H}_{i-1,i}^{-1}\mathbf{x}_i^p$, respectively.

Definition 7. *Consider a set of points in the current frame i, $\{\mathbf{x}_i^p\}^{p=1,\dots,N}$, and a trajectory for each point in a temporal neighborhood, $\mathcal{N}_i = \{j : i-k \leq j \leq i+k\}$, given by the following points*

$$\text{for } \ j = i-k,\dots,i-1, \ \ \mathbf{x}_j^p := \left(\prod_{l=j}^{i-1}\mathbf{H}_{l+1,l}\right)\mathbf{x}_i^p = \mathbf{H}_{j+1,j}\mathbf{H}_{j+2,j+1}\cdots \cdots\mathbf{H}_{i-1,i-2}\mathbf{H}_{i,i-1}\mathbf{x}_i^p, \tag{15}$$

on the left and

$$\text{for } \ j = i+1,\dots,i+k, \ \ \mathbf{x}_j^p := \left(\prod_{l=i}^{j-1}\mathbf{H}_{l,l+1}\right)\mathbf{x}_i^p = \mathbf{H}_{j-1,j}\mathbf{H}_{j-2,j-1}\cdots \cdots\mathbf{H}_{i+1,i+2}\mathbf{H}_{i,i+1}\mathbf{x}_i^p, \tag{16}$$

on the right. We define the convolution of each trajectory with a Gaussian function by

$$\text{for } \ p = 1,\dots,N, \ \ \widetilde{\mathbf{x}}_i^p := \left(G_\sigma * \{\mathbf{x}_j^p\}_{j=i-k,\dots,i+k}\right)_i. \tag{17}$$

*The **local point–based smoothing** approach is then obtained by the rectifying transformations $\mathbf{H}_i' := \widehat{\mathbf{H}}_i^{-1}$, where $\widehat{\mathbf{H}}_i$ is calculated from points $\{\mathbf{x}_i^p\}$ to the smoothed set $\{\widetilde{\mathbf{x}}_i^p\}$. Finally the stabilized image sequence is obtained by* (1) *as $\mathbf{I}_i'(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}_i'\mathbf{x}\}$.*

The steps of the local point–based smoothing are illustrated in Algorithm 8 and the corresponding Gaussian filtering in Algorithm 9.

In order to compute the homography from the points, we may follow the strategies proposed in [22]. The inhomogeneous system, where one element of the matrix is set constant, is the simplest one: we obtain a closed form solution from a system of eight linear equations with eight unknowns.

In the case of degenerate configurations (e.g. when three or four points are aligned), the system becomes underdetermined and it is not possible to find a unique solution. It is important to choose the four tracking points well distributed in a rectangle to try to avoid this situation. Nevertheless, depending on the motion of the camera and the amount of smoothness, we may come across with this problem.

The number of points to track depends on the type of transformation we choose. Since every point establishes two equations, we need one point for translations, two for Euclidean and similarity transforms, three for affinities and four for homographies. For Euclidean transformations there is one spare equation, and we have an overdetermined system. Another alternative is to always track four points and then adapt the resulting homography to the chosen parametrization.

Here, logically, the boundary conditions are applied to the points themselves with a definition similar to the one applied on the local matrix–based smoothing.

Definition 8. *Given a point $\mathbf{x}_i^p$ in the reference frame,*

Algorithm 8: Local point-based smoothing

```
Input  : $\{\mathbf{H}\}, \sigma, bc$
Output: $\{\mathbf{H}'\}$
$radius \leftarrow 3\sigma$
for $i \leftarrow 1$ to $N$ do
    // choose four points
    $\{\mathbf{x}_i^p\} \leftarrow \{(0,0),(0,500),(500,0),(500,500)\}$
    for $j \leftarrow 1$ to $radius$ do
        // forward projection
        if $i+j<N$ then
            $\mathbf{x}_{i+j}^1 \leftarrow \mathbf{H}_{i+j-1,i+j}\mathbf{x}_{i+j-1}^1$
            $\mathbf{x}_{i+j}^2 \leftarrow \mathbf{H}_{i+j-1,i+j}\mathbf{x}_{i+j-1}^2$
            $\mathbf{x}_{i+j}^3 \leftarrow \mathbf{H}_{i+j-1,i+j}\mathbf{x}_{i+j-1}^3$
            $\mathbf{x}_{i+j}^4 \leftarrow \mathbf{H}_{i+j-1,i+j}\mathbf{x}_{i+j-1}^4$
        // backward projection
        if $i-j>1$ then
            $\mathbf{x}_{i-j}^1 \leftarrow \mathbf{H}_{i-j,i-j+1}^{-1}\mathbf{x}_{i-j+1}^1$
            $\mathbf{x}_{i-j}^2 \leftarrow \mathbf{H}_{i-j,i-j+1}^{-1}\mathbf{x}_{i-j+1}^2$
            $\mathbf{x}_{i-j}^3 \leftarrow \mathbf{H}_{i-j,i-j+1}^{-1}\mathbf{x}_{i-j+1}^3$
            $\mathbf{x}_{i-j}^4 \leftarrow \mathbf{H}_{i-j,i-j+1}^{-1}\mathbf{x}_{i-j+1}^4$
    // convolve trajectories with a Gaussian kernel; Eq. (17)
    call Point Gaussian convolution($\{\mathbf{x}^1\}, \widetilde{\mathbf{x}}_i^1, i, \sigma, bc$)
    call Point Gaussian convolution($\{\mathbf{x}^2\}, \widetilde{\mathbf{x}}_i^2, i, \sigma, bc$)
    call Point Gaussian convolution($\{\mathbf{x}^3\}, \widetilde{\mathbf{x}}_i^3, i, \sigma, bc$)
    call Point Gaussian convolution($\{\mathbf{x}^4\}, \widetilde{\mathbf{x}}_i^4, i, \sigma, bc$)
    // calculate the smoothed homography from the four smoothed points
    Compute homography, $\widetilde{\mathbf{H}}_i$, from $\{\widetilde{\mathbf{x}}_i^p\} \leftrightarrow \{\mathbf{x}_i^p\}$
    $\mathbf{H}_i' \leftarrow \widetilde{\mathbf{H}}_i$
```

***Constant boundary conditions** are defined by*

$$\mathbf{x}_j^p := \begin{cases} \mathbf{x}_1^p & \text{if } j<1 \\ \mathbf{x}_N^p & \text{if } j>N \end{cases}. \tag{18}$$

***Neumann boundary conditions** are defined in the range $[-N+1, 2N-1]$ by*

$$\mathbf{x}_j^p := \begin{cases} \mathbf{x}_{-j+1}^p & \text{if } j<1 \\ \mathbf{x}_{2N-j}^p & \text{if } j>N \end{cases}. \tag{19}$$

***Dirichlet boundary conditions** are defined in the range $[-2N+2, 3N-1]$ by*

$$\mathbf{x}_j^p := \begin{cases} \mathbf{x}_{-j+2}^p + 2\mathbf{x}_1^p - 2\mathbf{x}_N^p & \text{if } -2N+2 \le j < -N+1 \\ 2\mathbf{x}_1^p - \mathbf{x}_{-j+1}^p & \text{if } -N+1 \le j < 1 \\ 2\mathbf{x}_N^p - \mathbf{x}_{2N-j}^p & \text{if } N \le j < 2N \\ \mathbf{x}_{j-2N}^p + 2\mathbf{x}_N^p - 2\mathbf{x}_1^p & \text{if } 2N \le j < 3N \end{cases}. \tag{20}$$

Algorithm 9: Point Gaussian convolution

```
Input  : ${\mathbf{x}}, i, \sigma, bc$
Output: $\widetilde{\mathbf{x}}_i$
$average \leftarrow 0$
$sum \leftarrow 0$
$radius \leftarrow 3\sigma$
if $radius > N$ then
    $radius \leftarrow N$
// Gaussian convolution
for $j \leftarrow i - radius$ to $i + radius$ do
    $value \leftarrow 0$
    if $j < 1$ then
        switch $bc$ do
            case $CONSTANT\_BC$
                $value \leftarrow \mathbf{x}_1$
            case $NEUMANN\_BC$
                $value \leftarrow \mathbf{x}_{-j+1}$
            case $DIRICHLET\_BC$
                $value \leftarrow 2\mathbf{x}_1 - \mathbf{x}_{-j+1}$
    else if $j > N$ then
        switch $bc$ do
            case $CONSTANT\_BC$
                $value \leftarrow \mathbf{x}_N$
            case $NEUMANN\_BC$
                $value \leftarrow \mathbf{x}_{2N-j}$
            case $DIRICHLET\_BC$
                $value \leftarrow 2\mathbf{x}_N - \mathbf{x}_{2N-j}$
    else
        $value \leftarrow \mathbf{x}_j$
    $gauss \leftarrow e^{-\frac{(j-i)^2}{2\sigma^2}}$
    $average \leftarrow average + gauss \times value$
    $sum \leftarrow sum + gauss$
$\widetilde{\mathbf{x}}_i \leftarrow average/sum$
```

2.2.2 Additive Methods

In this section, we describe two further simple strategies that avoid the composition of matrices. The benefit of these schemes is that the errors produced by the compositions are not accumulated. We call these approaches *additive methods* because the information is computed in an incremental way through the addition of the transformations. This relies on what we call *virtual trajectories*, where the information used for estimating the set of regularized transformations is not based on real trajectories, but on the integral of the velocity of, e.g., the central pixel.

Local linear matrix–based smoothing. This technique proposes a linear variant of the local matrix–based smoothing.

Definition 9. *We define the virtual trajectory of a homography as*

$$\bar{\mathbf{H}}_i := \mathbf{Id} + \sum_{l=1}^{i-1} \left(\mathbf{H}_{l,l+1} - \mathbf{Id}\right). \tag{21}$$

The smoothed coefficients of the matrix trajectory for $i = 1, \ldots, N$ are defined by

$$\widetilde{\mathbf{H}}_i(p,q) := \left(G_\sigma * \{\bar{\mathbf{H}}_j(p,q)\}\right)_i = \sum_{j=-k}^{j=k} G_\sigma(j)\bar{\mathbf{H}}_{i-j}(p,q). \tag{22}$$

and

$$\widehat{\mathbf{H}}_i = \widetilde{\mathbf{H}}_i - \bar{\mathbf{H}}_i + \mathbf{Id}. \tag{23}$$

The ***local linear matrix–based smoothing*** *approach is then obtained by the rectifying transformations $\mathbf{H}'_i := \widehat{\mathbf{H}}_i^{-1}$. Finally, the stabilized image sequence is obtained by (1) as $\mathbf{I}'_i(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}'_i\mathbf{x}\}$.*

Virtual trajectories differ from actual trajectories in that the points do not follow a real path in the images. This concept is clearer in the case of points (see the next method), where the virtual trajectory is not given by the true displacement of the points from frame to frame but by the apparent motion of the points in each position.

The boundary conditions for this method are equivalent to the local matrix–based smoothing strategy, as given in Definition 3.

The steps of Local matrix–based smoothing are illustrated in Algorithm 10. The inputs of the algorithm are given by the set of the transformations and the smoothing parameter, σ. In this case, the parametric transformations are previously converted to matrices.

```
Algorithm 10: Local linear matrix-based smoothing
Input  : {$\mathbf{H}$}, $\sigma$, $bc$
Output: {$\mathbf{H}'$}
$\bar{\mathbf{H}}_1 \leftarrow \mathbf{Id}$
// Compute the virtual matrix trajectories
for $i \leftarrow 2$ to $N$ do
    $\bar{\mathbf{H}}_i \leftarrow \bar{\mathbf{H}}^{i-1} + \mathbf{H}_{i-1,i} - \mathbf{Id}$
// convolve the virtual trajectories with a Gaussian kernel (DCT)
call Matrix DCT Gaussian convolution($\{\bar{\mathbf{H}}\}, \{\widetilde{\mathbf{H}}\}, \sigma, bc$)
for $i \leftarrow 1$ to $N$ do
    // compute the correction matrix
    $\widehat{\mathbf{H}}_i \leftarrow \widetilde{\mathbf{H}}_i - \bar{\mathbf{H}}_i + \mathbf{Id}$
    // compute its inverse
    $\mathbf{H}'_i \leftarrow \widehat{\mathbf{H}}_i^{-1}$
```

Since we obtain a unique virtual trajectory for the whole sequence, it is easy to apply a DCT based Gaussian filtering. This method is preferred to discrete filters because it is more precise [14]. The DCT based Gaussian filtering is detailed in Algorithm 11.

In the case of a pure translation, linear matrix based smoothing is equivalent to local linear point–based smoothing, that we shall examine next. However, for general transformations, the matrix–based scheme is less sensitive to abrupt camera changes and numerical inaccuracies.

Algorithm 11: Matrix DCT Gaussian convolution

```
Input  : $\{\mathbf{H}\}, \sigma, bc$
Output: $\{\widetilde{\mathbf{H}}\}$
// convolution in each matrix position
forall the $(p,q)$ do
    // copy the original image
    for $i \leftarrow N$ to $2N-1$ do
        $src(i) \leftarrow \mathbf{H}_{i-N+1}(p,q)$
    // boundary conditions
    switch $bc$ do
        case CONSTANT_BC
            for $i \leftarrow 1$ to $N-1$ do
                $src(i) \leftarrow \mathbf{H}_{1,1}(p,q)$
            for $i \leftarrow 2N$ to $3N-2$ do
                $src(i) \leftarrow \mathbf{H}_{N-1,N}(p,q)$
        case NEUMANN_BC
            for $i \leftarrow 1$ to $N-1$ do
                $src(i) \leftarrow \mathbf{H}_{N-i-1,N-i}(p,q)$
            for $i \leftarrow 2N$ to $3N-2$ do
                $src(i) \leftarrow \mathbf{H}_{3N-i-1,3N-i}(p,q)$
        case DIRICHLET_BC
            for $i \leftarrow 1$ to $N-1$ do
                $src(i) \leftarrow 2\mathbf{H}_{1,1}(p,q) - \mathbf{H}_{N-i-1,N-i}(p,q)$
            for $i \leftarrow 2N$ to $3N-2$ do
                $src(i) \leftarrow \mathbf{H}_{N-1,N}(p,q) - \mathbf{H}_{3N-i-1,3N-i}(p,q)$
    // apply DCT Gaussian convolution using the method in [14]
    call dct_gaussian_conv($dest, src, 3N-2, \sigma$)
    // copy the signal in the domain
    for $i \leftarrow 1$ to $N$ do
        $\widetilde{\mathbf{H}}_i(p,q) \leftarrow dest(N+i-1)$
```

Local linear point–based smoothing. This variant proposes a linear system to process the video by smoothing point trajectories. Instead of composing the transition homographies $H_{j+1,j}$ to compute the trajectories of the virtual points $\{\mathbf{x}_i^p\}^{p=1,\dots,4}$, the *virtual trajectories* of the reference points are computed by integrating their temporal discrete derivative. Then, as in the preceding section, they are smoothed out to compute a new position at time i. The new position of these points yields the stabilization homography.

Definition 10. *Consider a fixed set of points in frame coordinates,* $\{\mathbf{x}^p\}^{p=1,\dots,P}$*, and define their virtual trajectory by* $\mathbf{x}_1^p = \mathbf{x}^p$ *and*

$$\text{for } \ i = 1, \dots, N-1, \ \ \mathbf{x}_i^p := \mathbf{x}^p + \sum_{l=1}^{i-1} \left(\mathbf{H}_{l,l+1}\mathbf{x}^p - \mathbf{x}^p \right). \tag{24}$$

Algorithm 12: Local linear point–based smoothing

```
Input  : $\{\mathbf{H}_{i,i+1}\}, \sigma, bc$
Output: $\{\mathbf{H}'_i\}$
// choose four fixed points for all frames
$\{\mathbf{x}^p\} \leftarrow \{(0,0),(0,500),(500,0),(500,500)\}$
$\mathbf{x}^1_1 \leftarrow \mathbf{x}^1$, $\mathbf{x}^2_1 \leftarrow \mathbf{x}^2$, $\mathbf{x}^3_1 \leftarrow \mathbf{x}^3$, $\mathbf{x}^4_1 \leftarrow \mathbf{x}^4$
// compute the virtual trajectories
for $i \leftarrow 2$ to $N$ do
    $\mathbf{x}^1_i \leftarrow \mathbf{x}^1_{i-1} + (\mathbf{H}_{i-1,i}\mathbf{x}^1 - \mathbf{x}^1)$
    $\mathbf{x}^2_i \leftarrow \mathbf{x}^2_{i-1} + (\mathbf{H}_{i-1,i}\mathbf{x}^2 - \mathbf{x}^2)$
    $\mathbf{x}^3_i \leftarrow \mathbf{x}^3_{i-1} + (\mathbf{H}_{i-1,i}\mathbf{x}^3 - \mathbf{x}^3)$
    $\mathbf{x}^4_i \leftarrow \mathbf{x}^4_{i-1} + (\mathbf{H}_{i-1,i}\mathbf{x}^4 - \mathbf{x}^4)$
// DCT Gaussian convolution of each virtual trajectory (using [14])
call Point DCT Gaussian convolution($\{\mathbf{x}^1\}, \{\widetilde{\mathbf{x}}^1\}, \sigma, bc$)
call Point DCT Gaussian convolution($\{\mathbf{x}^2\}, \{\widetilde{\mathbf{x}}^2\}, \sigma, bc$)
call Point DCT Gaussian convolution($\{\mathbf{x}^3\}, \{\widetilde{\mathbf{x}}^3\}, \sigma, bc$)
call Point DCT Gaussian convolution($\{\mathbf{x}^4\}, \{\widetilde{\mathbf{x}}^4\}, \sigma, bc$)
for $i \leftarrow 1$ to $N$ do
    $\widehat{\mathbf{x}}^1_i := \mathbf{x}^1 - \mathbf{x}^1_i + \widetilde{\mathbf{x}}^1_i$
    $\widehat{\mathbf{x}}^2_i := \mathbf{x}^2 - \mathbf{x}^1_i + \widetilde{\mathbf{x}}^2_i$
    $\widehat{\mathbf{x}}^3_i := \mathbf{x}^3 - \mathbf{x}^1_i + \widetilde{\mathbf{x}}^3_i$
    $\widehat{\mathbf{x}}^4_i := \mathbf{x}^4 - \mathbf{x}^1_i + \widetilde{\mathbf{x}}^4_i$
    Compute homography, $\widetilde{\mathbf{H}}_i$, from $(\widehat{\mathbf{x}}^1_i, \widehat{\mathbf{x}}^2_i, \widehat{\mathbf{x}}^3_i, \widehat{\mathbf{x}}^4_i) \leftrightarrow (\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4)$
    $\mathbf{H}'_i \leftarrow \widetilde{\mathbf{H}}_i$
```

We define the stabilized position of $\mathbf{x}^p_i$ *as*

$$\widetilde{\mathbf{x}}^p_i := \left(G_\sigma * \{\mathbf{x}^p_j\}\right)_i = \sum_{j=-k}^{j=k} G_\sigma(j)\mathbf{x}^p_{i-j}, \tag{25}$$

$$\widehat{\mathbf{x}}^p_i := \mathbf{x}^p - \mathbf{x}^p_i + \widetilde{\mathbf{x}}^p_i. \tag{26}$$

The ***local linear point–based smoothing*** *approach is then obtained by the rectifying transformations* $\mathbf{H}'_i := \widehat{\mathbf{H}}^{-1}_i$*, where the homography* $\widehat{\mathbf{H}}_i$ *is calculated from the fixed points* $\{\mathbf{x}^p\}$ *to the stabilized set at frame i,* $\{\widehat{\mathbf{x}}^p_i\}$*. Finally, the stabilized image sequence is obtained by* (1) *as* $\mathbf{I}'_i(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}'_i\mathbf{x}\}$*.*

The boundary conditions for this method are equivalent to the local point–based smoothing strategy, as given in Definition 8.

The steps of the point–based smoothing are detailed in Algorithm 12. The inputs of the algorithm are given by the set of transformations and the smoothing parameter, σ. The convolution of the point trajectories with the Gaussian kernel is given in Algorithm 13, which is based on the DCT.

A clear benefit of the additive approaches is that, unlike the previous techniques, no composition between homographies is performed. This reduces the effect of numerical errors or the influence of errors during the motion estimation step. Another advantage is that it is easy to use a Gaussian convolution based on the DCT, since we obtain a unique one–dimensional signal for each point trajectory or matrix coefficient.

Algorithm 13: Point DCT Gaussian convolution

```
Input  : $\{\mathbf{x}\}, \sigma, bc$
Output: $\{\widetilde{\mathbf{x}}\}$
// copy the original signal
for $i \leftarrow N$ to $2N-1$ do
    $src(i) \leftarrow \mathbf{x}_{i-N+1}$
// boundary conditions
switch $bc$ do
    case CONSTANT_BC
        for $i \leftarrow 1$ to $N-1$ do
            $src(i) \leftarrow \mathbf{x}_1$
        for $i \leftarrow 2N$ to $3N-2$ do
            $src(i) \leftarrow \mathbf{x}_N$
    case NEUMANN_BC
        for $i \leftarrow 1$ to $N-1$ do
            $src(i) \leftarrow \mathbf{x}_{N-i-1}$
        for $i \leftarrow 2N$ to $3N-2$ do
            $src(i) \leftarrow \mathbf{x}_{3N-i-1}$
    case DIRICHLET_BC
        for $i \leftarrow 1$ to $N-1$ do
            $src(i) \leftarrow 2\mathbf{x}_1 - \mathbf{x}_{N-i-1}$
        for $i \leftarrow 2N$ to $3N-2$ do
            $src(i) \leftarrow \mathbf{x}_N - \mathbf{x}_{3N-i-1}$
// apply DCT Gaussian convolution using method in [14]
call dct_gaussian_conv($dest, src, 3N-2, \sigma$)
// copy the signal in the domain
for $i \leftarrow 1$ to $N$ do
    $\widetilde{\mathbf{x}}_i \leftarrow dest(N+i-1)$
```

In the experimental results, we will use a linear scale–space analysis which is based on a formulation similar to (24). Furthermore, we will see that these two approaches behave much better if we analyze the empty regions created by the stabilization.

2.3 Post-Processing: Crop and Zoom Strategies

A post–processing step is necessary to remove the empty regions that appear at the border of the images. The simplest approach, and probably the most used, is Crop & Zoom. The idea is to find the largest area axis–parallel rectangle [10] that does not contain empty regions, and apply the adequate crop to all frames to remove them. This can be realized by translating and scaling the rectangle in order to fit the image dimensions. This process can be executed before the warping step in the smoothed homographies to account for this similarity transformation.

A fast and simple method for computing the rectangle is given in Algorithm 14. It iteratively projects the corners of the images using the homographies of each frame and updates the limits of the rectangle, given by (x_1, y_1) and (x_2, y_2), if necessary. This method is very fast, although it does not necessarily obtain the largest area rectangle.

Algorithm 14: Fast computation of inscribed rectangle

```
Input   : $\{\mathbf{H}'_i\}, n_x, n_y$
Output: $x_1, y_1, x_2, y_2$

// variables to store the two corners of the rectangle
$x_1 \leftarrow 0, y_1 \leftarrow 0, x_2 \leftarrow n_x - 1, y_2 \leftarrow n_y - 1$

// select the maximum rectangle without empty regions
for $i \leftarrow 1$ to $N$ do

    // project the top-left corner
    $(x, y, z)^T \leftarrow (\mathbf{H}'_i)^{-1} \cdot (0, 0, 1)^T$
    if $x^z > x_1$ then $x_1 \leftarrow x/z$
    if $y/z > y_1$ then $y_1 \leftarrow y/z$

    // project the top-right corner
    $(x, y, z)^T \leftarrow (\mathbf{H}'_i)^{-1} \cdot (n_x - 1, 0, 1)^T$
    if $x/z < x_2$ then $x_2 \leftarrow x/z$
    if $y/z > y_1$ then $y_1 \leftarrow y/z$

    // project the bottom-left corner
    $(x, y, z)^T \leftarrow (\mathbf{H}'_i)^{-1} \cdot (0, n_y - 1, 1)^T$
    if $x/z > x_1$ then $x_1 \leftarrow x/z$
    if $y/z < y_2$ then $y_2 \leftarrow y/z$

    // project the bottom-right corner
    $(x, y, z)^T \leftarrow (\mathbf{H}'_i)^{-1} \cdot (n_x - 1, n_y - 1, 1)^T$
    if $x/z < x_2$ then $x_2 \leftarrow x/z$
    if $y/z < y_2$ then $y_2 \leftarrow y/z$
```

If we want to compute the largest rectangle, then the process is more complex. One alternative is given by the following steps:

- Find the convex polygon given by the intersection of all the quadrilaterals produced by the homographies.
- Create a binary image with the same size as the video frame, with the pixels inside the polygon equal to 1 and the pixels outside equal to 0, for instance.
- Compute the rectangle using the algorithm of the maximum rectangle inscribed in a histogram[2].

In fact, the last two steps can be substituted by a direct method, such as [12] or [1], which yields the coordinates of the rectangle directly from the intersecting polygon. However, we prefer these steps because they are simpler to implement.

[2] The method of the maximum area axis–parallel rectangle inscribed in a histogram is explained in http://stackoverflow.com/questions/4311694/maximize-the-rectangular-area-under-histogram

In the first step, the objective is to find the common area not including the empty regions. This can be easily obtained by projecting the four corners with the homographies as in the previous algorithm. We get a set of quadrilaterals that are intersected to form a convex polygon, which defines the region that is visible in all the frames. This problem is typical in computer graphics and there are several standard algorithms, such as [59] or [16]. Since our polygons are convex, one can use the method of Sutherland and Hodgman [55].

In the second step, a binary image is created, with zeros outside the polygon and ones inside. Since the polygon is convex, this process is simple: it computes the minimum and maximum y–coordinates of the polygon, y_{min} and y_{max}, respectively; then, the horizontal lines, y_i, from y_{min} to y_{max} produce two intersection points with the polygon, (x_1, y_i) and (x_2, y_i) for each line. The image is filled between x_1 and x_2 in this line. The algorithm needs to control two special cases: when the intersection with the polygon is just one point, which happens when a vertex of the polygon joins an ascending and descending edge, or when it intersects a whole horizontal edge.

In the last step, the rectangle is found by using an algorithm based on the histogram, detailed in Algorithm 15. This works as follows: the values of the mask image are accumulated by columns (lines 1–4), which gives us the estimate of the height of all possible rectangles; then, the width of all possible rectangles is calculated by testing if the histograms of the neighbor columns, in the same line, are equal or larger than the actual value. This process is first carried out on the left columns, using a temporal array, $L(j)$ (lines 7–13), and then on the right, using another temporal array, $R(j)$ (lines 15–21). The image is processed line by line, calculating the maximum area (lines 22–29). Using a stack, this algorithm has $O(N)$ complexity.

In practice, the fastest Algorithm 14 provides results which are usually close to the accurate Algorithm 15. It typically diverges by a few pixels for large values of the smoothing radius, which depends on σ, or for large amounts of camera shakiness.

One important issue is that, in general, the rectangle dimensions will not be proportional to the size of the images and an additional crop is necessary in one of the dimensions. One alternative is to give up preserving the size of the original video and keep the largest possible contents, although it is more usual to generate a video with the same resolution.

Definition 11. *Let* $(x_m, y_m) = \left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right)$ *be the center of the rectangle and let* $s = \min(\frac{x_2-x_1}{n_x}, \frac{y_2-y_1}{n_y})$ *be the scale factor of the shortest relative dimension of the rectangle. We define the Crop&Zoom transformation as*

$$\mathbf{T} := \begin{pmatrix} 1 & 0 & x_m \\ 0 & 1 & y_m \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -n_x/2 \\ 0 & 1 & -n_y/2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s & 0 & x_m - s\,n_x/2 \\ 0 & s & y_m - s\,n_y/2 \\ 0 & 0 & 1 \end{pmatrix}, \tag{27}$$

which is a similarity transform that allows to remove the empty regions, and

$$\mathbf{I}'_i(\mathbf{x}) := \mathbf{I}_i(\mathbf{H}'_i\mathbf{T}\mathbf{x}), \tag{28}$$

the stabilized images without empty regions.

Finally, Algorithm 16 details how to compute these transformations. The two corners of the rectangle, the size of the frames and the scale factor are used to define the similarity transform. The input homographies are composed with this similarity. Figure 4 shows an example of the Crop&Zoom strategy and the computed rectangle.

There exist other alternatives for removing the empty regions, as we have seen in the introduction. Other post–processing tasks are aimed at improving the quality of the video, such as reducing the effect of motion blur or the changes in illumination.

Algorithm 15: Accurate computation of inscribed rectangle

```
Input   : $I$
Output: $x_1, y_1, x_2, y_2$
// initialize histogram array to 0
$H \leftarrow \mathbf{0}$
// create the histogram array
for $i \leftarrow 2$ to $number\_of\_rows(I)$ do
    for $j \leftarrow 1$ to $number\_of\_columns(I)$ do
        if $I(i,j)$ then $H(i,j) \leftarrow 1 + H(i-1,j)$
$max\_area \leftarrow 0$
// find the maximum area axis-parallel rectangle
for $i \leftarrow 1$ to $number\_of\_rows(I)$ do
    // find the left limit for this position of the histogram
    for $j \leftarrow 1$ to $number\_of\_columns(I)$ do
        while Not $stack.empty()$ and $H(i,j) \leq H(i, stack.top())$ do
            $stack.pop()$
        if $stack.empty()$ then  $t \leftarrow 1$
        else $t \leftarrow stack.top()$
        $L(j) \leftarrow t$
        $stack.push(j)$
    empty $stack$
    // find the right limit for this position of the histogram
    for $j \leftarrow number\_of\_columns(I)$ to 1 do
        while Not $stack.empty()$ and $H(i,j) \leq H(i, stack.top())$ do
            $stack.pop()$
        if $stack.empty()$ then  $t \leftarrow number\_of\_columns(I)$
        else $t \leftarrow stack.top()$
        $R(j) \leftarrow t$
        $stack.push(j)$
    // calculate final area in the row and find the maximum
    for $j \leftarrow 1$ to $number\_of\_columns(I)$ do
        $A \leftarrow H(i,j) \cdot (R(j) - L(j))$
        if $A > max\_area$ then
            $max\_area \leftarrow A$
            $x_1 \leftarrow L(j)$
            $y_1 \leftarrow i - H(i,j)$
            $x_2 \leftarrow R(j)$
            $y_2 \leftarrow i$
```

Algorithm 16: Crop and Zoom

```
Input  : $\{\mathbf{H}'_i\}, n_x, n_y, x_1, y_1, x_2, y_2$
Output: $\{\mathbf{H}'_i\}$

// compute the scale factor
$s_1 \leftarrow (y_2 - y_1)/n_y$
$s_2 \leftarrow (x_2 - x_1)/n_x$
if $s_1 < s_2$ then $s \leftarrow s_1$ else $s \leftarrow s_2$

// center of the rectangle
$x_m \leftarrow (x_1 + x_2)/2$
$y_m \leftarrow (y_1 + y_2)/2$

// use a similarity transform for scaling and translating
$\mathbf{T} \leftarrow \begin{pmatrix} s & 0 & x_m - s\,n_x/2 \\ 0 & s & y_m - s\,n_y/2 \\ 0 & 0 & 1 \end{pmatrix}$

// modify the stabilizing transformations with the similarity transform
for $i \leftarrow 1$ to $N$ do
    $\mathbf{H}'_i \leftarrow \mathbf{H}'_i \cdot \mathbf{T}$
```

3 Evaluation and Results

3.1 Comparison of Methods

Table 2 presents the average image percentage for the empty regions left on the border of the compensated frames for moderate smoothing ($\sigma := 30$) and ten diverse videos. This average was computed for each kind of geometric transform, from translation to homography, and for all kinds of smoothing processes that we have considered, from compositional to local linear point–based with Neumann boundary condition.

A lower empty region percentage is a logical and factual criterion of success: For a fixed Gaussian smoothing, it measures the loss of resolution of the video caused by the camera shake correction.

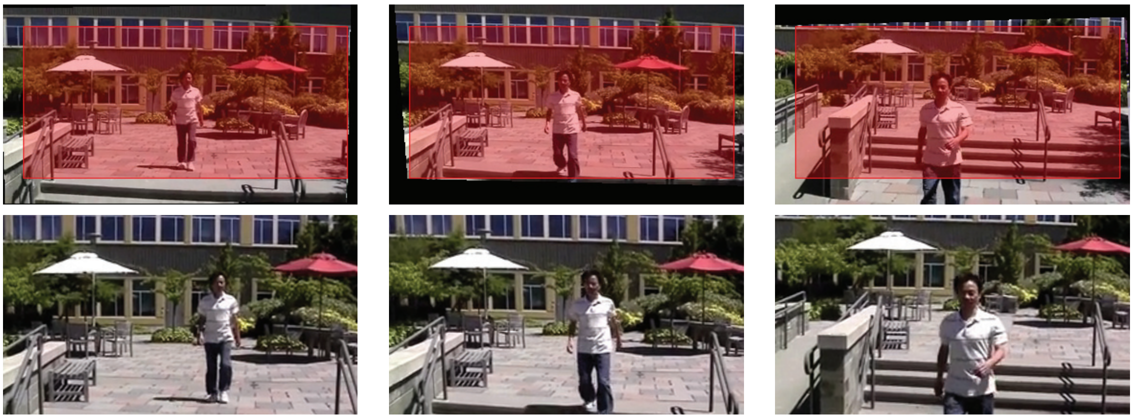


Figure 4: Crop&Zoom: First row, several frames of a video with the inscribed rectangle obtained through Algorithm 14; second row, the same frames after the Crop&Zoom process.

Method	**Translat.**	**Euclid.**	**Similar.**	**Affine**	**Homography**
Compositional	20,5%	23,8%	21,7%	23,2%	43,8%
Compositional smoothing	**4,0%**	**4,1%**	6,3%	6,4%	8,4%
Compositional local smth.	8,3%	10,5%	8,1%	8,5%	18,9%
Local matrix–based	**4,0%**	**4,0%**	6,3%	6,4%	6,1%
Local point–based	**4,0%**	**4,1%**	6,7%	6,8%	11,2%
Local linear matrix–based	**4,0%**	**4,1%**	**4,6%**	**4,6%**	**5,5%**
Local linear point–based	**4,0%**	**4,0%**	**4,6%**	**4,6%**	**5,9%**

Table 2: Average percentage of empty regions left on the border of the compensated frames for ($\sigma := 30$). This percentage is given for each kind of transform, from translation to homography, and for each kind of smoothing process, from compositional to local linear point–based. This criterion illustrates two phenomena: First, the growing numbers from left to right show that allowing more degrees of freedom to the transform implies also a more severe correction, as is logical; second, the table is a clear decider about the preferable smoothing process. The local linear matrix–based and the (similar) local linear point–based processes win as they guarantee the smallest distortion and the smallest distortion increase when increasing the number of degrees of freedom.

Clearly large percentages like 23% would not be acceptable. Fortunately, the best methods give a reasonable 5 to 6%.

The results demonstrate two phenomena: First, the increasing numbers from left to right on each row show that allowing more degrees of freedom in the transform implies also a more severe correction; second, the table is a clear decider about the preferable smoothing process. The local linear matrix–based and the (similar) local linear point–based processes win as they guarantee the smallest distortion and the smallest distortion increase when increasing the number of degrees of freedom. Table 3 shows the average percentage of empty regions left on the border of the compensated frames for strong smoothing ($\sigma := 60$). This percentage is again given for each kind of transform, from translation to homography, and for each kind of smoothing process, from compositional to local linear point–based. This table confirms, for more drastic smoothing, the conclusion given in Table 2: The local linear matrix–based and the local linear point–based processes win. They guarantee the smallest distortion and the smallest distortion increase when increasing the number of degrees of freedom. Interestingly, the winners are also the closest ones to the linear scale-space strategy designed for video analysis in the next section. The tables also confirm that the local matrix–based approach is the third best method and it is, on average, the best of the compositional approaches, as shown in [43].

Method	**Translat.**	**Euclid.**	**Similar.**	**Affine**	**Homography**
Compositional	20,5%	23,8%	21,7%	23,2%	43,8%
Compositional smoothing	**5,5%**	**5,6%**	12,6%	12,7%	13,5%
Compositional local smth.	11,9%	14,2%	10,6%	10,5%	21,7%
Local matrix–based	**5,5%**	**5,5%**	12,6%	12,7%	11,0%
Local point–based	**5,5%**	**5,5%**	14,0%	14,1%	18,3%
Local linear matrix–based	**5,5%**	**5,5%**	**7,3%**	**7,3%**	**8,9%**
Local linear point–based	**5,5%**	**5,6%**	**7,3%**	**7,3%**	**9,4%**

Table 3: Average percentage of the empty regions left on the border of the compensated frames for ($\sigma := 60$). This percentage is given for each kind of transform, from translation to homography and for each kind of smoothing process, from compositional to local linear point-based. This table confirms, for more drastic smoothing, the conclusion given for Table 2: The local linear matrix–based and the (similar) local linear point–based processes win.

3.2 The Motion Linear Scale-Space

If $\mathbf{H}_{i,i+1}$ are the original transformations between consecutive frames and $\mathbf{H}'_i$ the rectifying homographies, we calculate the set of transformations for the stabilized sequence as

$$\widetilde{\mathbf{H}}_{i,i+1} := (\mathbf{H}'_{i+1})^{-1}\mathbf{H}_{i,i+1}\mathbf{H}'_i, \tag{29}$$

with $\widetilde{\mathbf{H}}_{0,1} := \mathbf{Id}$. These are used for computing the trajectories and scale–space graphics in the following experiments. Our goal in this section is to define sets of motion signals that can undergo a multi–scale analysis and that give a reliable geometric account of the camera motion. We assume that a series of homographies $\{\mathbf{H}_i\}_{i=1,\ldots,N}$ between successive frames is given. These may be the initial homographies or the smoothed ones. We want to define and visualize representative virtual trajectories associated with this series. This analysis is useful to visualize and check the correct behavior of the methods and, on the other hand, characterize the ego-motion from the video.

Definition 12. *Let* $\mathbf{x}$ *be a fixed point in the video frame domain (for example its central point). Given a series* $\{\mathbf{H}_i\}_{i=1,\cdots,N}$ *of homographies between successive frames, we call* virtual trajectory of $\mathbf{x}$ *the series*

$$\mathbf{x}_i := \mathbf{x} + \sum_{j=1}^{j=i-1} (\mathbf{H}_i\mathbf{x} - \mathbf{x}). \tag{30}$$

We call instantaneous zoom *the area ratio between the quadrangles* $(\mathbf{H}_i\mathbf{w}, \mathbf{H}_i\mathbf{x}, \mathbf{H}_i\mathbf{y}, \mathbf{H}_i\mathbf{z})$ *and* $(\mathbf{w}, \mathbf{x}, \mathbf{y}, \mathbf{z})$*, where the second quadrangle is the image domain. We call* instantaneous rotation *of the video the angle between* $(\mathbf{H}_i\mathbf{u}, \mathbf{H}_i\mathbf{v})$ *and* $(\mathbf{u}, \mathbf{v})$*, where* $(\mathbf{u}, \mathbf{v})$ *denote the middle points of the left and right sides of the video frame.*

Virtual trajectories associated with a series of homographies are obtained through Algorithm 17.

Algorithm 17: Trajectory plot

```
Input  : ${\mathbf{H}_i}, n_x, n_y$
Output: ${\mathbf{v}_i}$

// central point
$\mathbf{x}_m \leftarrow (n_x/2, n_y/2, 1)^T$
$\mathbf{v}_0 \leftarrow (0, 0, 1)^T$

// project the central point and accumulate the velocity
for $i \leftarrow 1$ to $N$ do
    $\mathbf{x} \leftarrow \mathbf{H}_i \cdot \mathbf{x}_m$

    // normalize the vector by the third component
    $\mathbf{x} \leftarrow \mathbf{x}/\mathbf{x}_3$
    $\mathbf{v}_i \leftarrow \mathbf{v}_{i-1} + (\mathbf{x} - \mathbf{x}_m)$
```

Algorithm 18 is used to visualize the evolution of the zoom factor. It computes the observable area change rate induced by the homographies. This rate is an approximation of the square of the zoom rate. For similarities, this is equivalent to the scale factor, which can be obtained directly from their parameters.

Algorithm 19 estimates the angle of rotation for each transformation. It projects the horizontal line that passes through the center of the image, using each homography, and computes the angle

Algorithm 18: Zoom plot

```
Input  : $\{\mathbf{H}_i\}, n_x, n_y$
Output: $\{\mathbf{z}_i\}$

// compute the zoom for each transformation
for $i \leftarrow 1$ to $N$ do
    $\mathbf{w} \leftarrow \mathbf{H}_i \cdot (0, 0, 1)^T$
    $\mathbf{x} \leftarrow \mathbf{H}_i \cdot (0, n_y, 1)^T$
    $\mathbf{y} \leftarrow \mathbf{H}_i \cdot (n_x, 0, 1)^T$
    $\mathbf{z} \leftarrow \mathbf{H}_i \cdot (n_x, n_y, 1)^T$

    // compute the area of the two triangles
    $A_1 \leftarrow TriangleArea(\mathbf{w}, \mathbf{x}, \mathbf{y})$
    $A_2 \leftarrow TriangleArea(\mathbf{x}, \mathbf{y}, \mathbf{z})$
    $z_i \leftarrow (A_1 + A_2)/(n_x \cdot n_y)$
```

Algorithm 19: Rotation plot

```
Input  : $\{\mathbf{H}_i\}, n_x, n_y$
Output: $\{\mathbf{r}_i\}$

// compute the angle of rotation of the central line
for $i \leftarrow 1$ to $N$ do
    $\mathbf{x} \leftarrow \mathbf{H}_i \cdot (0, n_y/2, 1)^T$
    $\mathbf{y} \leftarrow \mathbf{H}_i \cdot (n_x, n_y/2, 1)^T$
    $\mathbf{z} \leftarrow \mathbf{y} - \mathbf{x}$
    $\mathbf{r}_i \leftarrow arctan(z_y/z_x) \cdot 180/\pi$
```

formed by these two lines. For similarities and Euclidean transformations, this can again be obtained directly from the parameters.

Our scale-space analysis receives discrete signals (virtual trajectories) obtained from the series of transition homographies. The trajectories are calculated using Definition 12. These trajectories will be smoothed by Gaussian convolutions to preserve all the typical scale properties, namely translation invariance, locality and causality. Thus, we need first to specify how the Gaussian definition must be done to preserve these three properties.

Consider a digital signal $\mathbf{u}_k$ with $k = 0, \ldots, N-1$ and its Discrete Fourier Transform (DFT) interpolate

$$u(x) = \sum_{m \in [-N/2, N/2-1]} \tilde{u}_m e^{\frac{2i\pi mx}{N}}, \tag{31}$$

where $\tilde{u}_m$ are the DFT coefficients of the N samples $\mathbf{u}_k$. The classic image analysis theory called *scale-space* convolves u with a Gaussian $G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$. A direct calculation shows that the result of the convolution of u with G_σ is

$$v(x) := (G_\sigma * u)(x) = \sum_{m \in [-\frac{N}{2}, \frac{N}{2}-1]} \tilde{u}_m \widehat{G_\sigma}\left(\frac{2m\pi}{N}\right) e^{\frac{2i\pi mx}{N}} \tag{32}$$

Algorithm 20: Scale-Space plot

```
Input  : ${\mathbf{H}_{1,i}}, {\mathbf{H}_{2,i}}, n_x, n_y$
Output: ${\mathbf{s}_i}$

// coordinates of the central point
$\mathbf{x}_m \leftarrow (n_x/2, n_y/2, 1)^T$
$\mathbf{v}_{1,0} \leftarrow (0,0,1)^T$
$\mathbf{v}_{2,0} \leftarrow (0,0,1)^T$
$\mathbf{s}_0 \leftarrow (0,0,1)^T$

// project and accumulate the velocity at the central point
for $i \leftarrow 1$ to $N$ do

    // project the central point using the two homographies
    $\mathbf{x} \leftarrow \mathbf{H}_{1,i} \cdot \mathbf{x}_m$
    $\mathbf{y} \leftarrow \mathbf{H}_{2,i} \cdot \mathbf{x}_m$

    // normalize vectors by the third component
    $\mathbf{x} \leftarrow \mathbf{x}/\mathbf{x}_3$
    $\mathbf{y} \leftarrow \mathbf{y}/\mathbf{y}_3$

    // update the accumulated velocities with the position increment
    $\mathbf{v}_{1,i} \leftarrow \mathbf{v}_{1,i-1} + (\mathbf{x} - \mathbf{x}_m)$
    $\mathbf{v}_{2,i} \leftarrow \mathbf{v}_{2,i-1} + (\mathbf{y} - \mathbf{x}_m)$
    $\mathbf{s}_i \leftarrow \mathbf{v}_{2,i} - \mathbf{v}_{1,i}$
```

with $\widehat{G_\sigma}(\xi) = e^{\frac{-\sigma^2\xi^2}{2}}$. Indeed,

$$G_\sigma * e^{\frac{2i\pi mx}{N}} = \int_{\mathbf{R}} G_\sigma(y) e^{\frac{2i\pi m(x-y)}{N}} dy = e^{\frac{2i\pi mx}{N}} \int_{\mathbf{R}} G_\sigma(y) e^{-\frac{2i\pi my}{N}} dy = e^{\frac{2i\pi mx}{N}} \widehat{G_\sigma}(\frac{2\pi m}{N}).$$

The convolved discrete digital signal therefore simply is

$$\mathbf{v}_k := G_\sigma * \mathbf{u}_k = \sum_{m \in [-\frac{N}{2}, \frac{N}{2}-1]} \widehat{G_\sigma}\left(\frac{2m\pi}{N}\right) \tilde{u}_m e^{\frac{2i\pi mk}{N}}. \tag{33}$$

These calculations lead us to the following definition of Gaussian filtering of a discrete motion signal.

Definition 13. *Given a signal $\mathbf{u}_k$, $k = 0, \dots, N-1$ we call DFT convolution of this signal by a Gaussian G_σ the discrete signal $G_\sigma * \mathbf{u}_k$ obtained by (33).*

Nevertheless this definition implicitly assumes the signal to be N-periodic, which is generally not adequate, as it introduces artificial discontinuities at both ends of the signal. For this reason, the convolution must respect the constant, Neumann, or Dirichlet boundary conditions used in all of our previous definitions of motion smoothing. This leads to the obvious next definition, which proposes again to extend the signal adequately on an interval of length $3N$ before applying the DFT convolution.

Definition 14. *Given a signal $\{\mathbf{v}_k\}$, and a standard deviation for the Gaussian σ we set, for $k = N$ to $2N-1$, $\mathbf{va}_k = \mathbf{v}_{k-N+1}$.*

(i) For a constant boundary condition, for $k = 1$ to $N - 1$, $\mathbf{va}_k = \mathbf{v}_1$ and for $k = 2N$ to $3N - 2$, $\mathbf{va}_k = \mathbf{v}_N$.

(ii) For a Neumann boundary condition, for $k = 1$ to $N - 1$ $\mathbf{va}_k = \mathbf{v}_{N-k-1}$ and for $k = 2N$ to $3N - 2$ $\mathbf{va}_k = \mathbf{v}_{3N-k-1}$.

(iii) For a Dirichlet boundary condition, for $k = 1$ to $N - 1$, $\mathbf{va}_k = 2\mathbf{v}_1 - \mathbf{v}_{N-k-1}$ and for $k = 2N$ to $3N - 2$, $\mathbf{va}_k = \mathbf{v}_N - \mathbf{v}_{3N-k-1}$.

Then we apply the DFT convolution of the signal $\mathbf{va}_k$ with the Gaussian G_σ by Definition 13 to obtain a discrete signal $\mathbf{sv}_k$, $k = 1, \ldots, 3N - 2$. Finally, we extract the meaningful part of the signal for $k = 1$ to N, by setting $\mathbf{sv}_k = \mathbf{vb}_{N+k-1}$.

Our definition of the motion scale-space will be applied to four signals: The virtual trajectory (x and y components) of the central point, the instantaneous zoom and the instantaneous rotation.

Definition 15. *Let $\mathbf{u}_k$, $k = 1, \ldots, N - 1$ be one of the movie motion signals given by Definition 12. We call* scale-space *of $\mathbf{u}_k$ the following series of signals, which are also displayed in this order from bottom to top in all graphical representations:*

a) the original virtual trajectory $\mathbf{u}_k$,

b) the high pass filtered signal $\mathbf{v_{10}} := \mathbf{u}_k - (G_{10} * \mathbf{u})_k$ where $G_{10} * \mathbf{u}$ is defined in Definition 14,

c) the low passed version of b) defined by $\mathbf{v}_{20} := (G_{10} * \mathbf{u})_k - (G_{20} * \mathbf{u})_k$,

d) the low passed version of c) defined by $\mathbf{v}_{40} := (G_{20} * \mathbf{u})_k - (G_{40} * \mathbf{u})_k$,

e) the low passed version of d) defined by $\mathbf{v}_{80} := (G_{40} * \mathbf{u})_k - (G_{80} * \mathbf{u})_k$,

f) the final low passed version of $\mathbf{u}$, $\mathbf{u}_{80} = (G_{80} * \mathbf{u})_k$.

These definitions define a pyramid that can be collapsed back as we have $\mathbf{u} = \mathbf{u}_{80} + \mathbf{v}_{80} + \mathbf{v}_{40} + \mathbf{v}_{20} + \mathbf{v}_{10}$. Of course the basis value of $\sigma = 10$ can be replaced by any other. In addition our scale-space contemplates the frequency analysis of the pyramid by DFT, namely the DFTs of $\mathbf{v}_{80}$, $\mathbf{v}_{40}$, $\mathbf{v}_{20}$ and $\mathbf{v}_{10}$, presented on the right side of all the graphics from top to bottom.

Algorithm 21: Linear scale-space plot

```
Input  : $\{\mathbf{v}_{1,i}\}, \{\mathbf{v}_{2,i}\}$
Output: $\{\mathbf{vs}_i\}$
for $i \leftarrow 1$ to $N$ do
    $\mathbf{vs}_i \leftarrow \mathbf{v}_{2,i} - \mathbf{v}_{1,i}$
```

3.3 Scale-Space Experiments

As an exemplary scale-space experiment, we analyze the sequence of a man running with a camera fixed on his head. The trajectories of the camera and the smoothing techniques are shown in Figure 5.

We observe in Figure 6 the characteristic motion of a running person, where the displacement is periodic in both the horizontal and vertical directions. There is a dominant peak in the x component

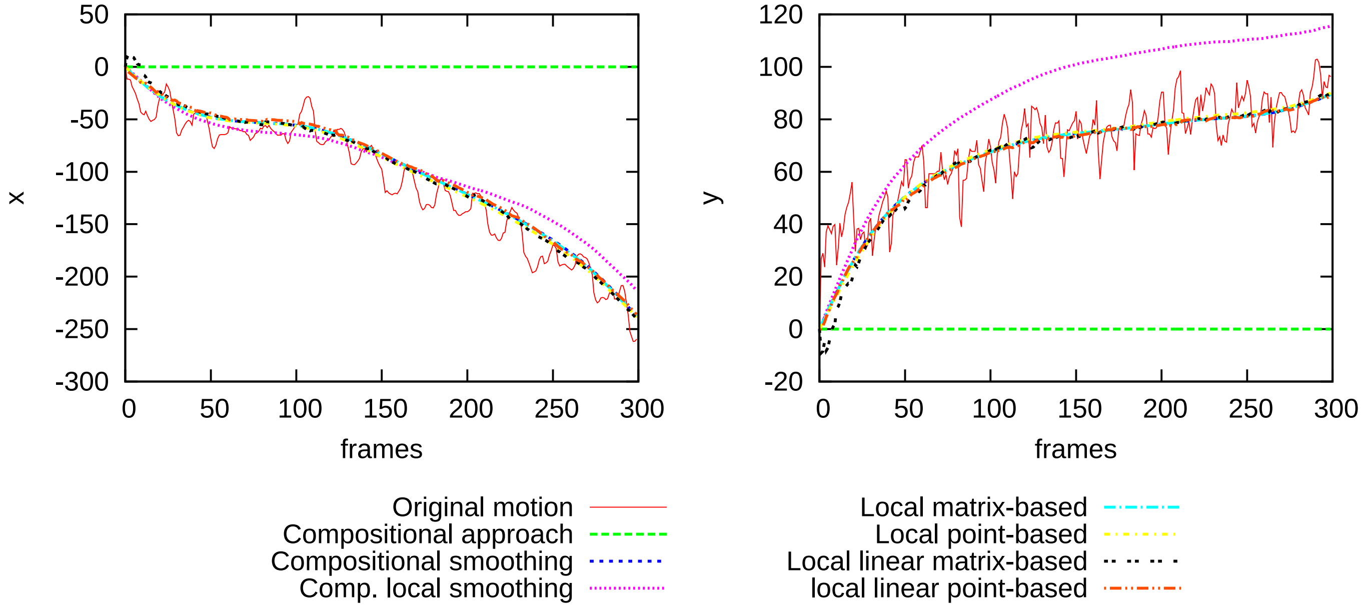


Figure 5: *Man running with a GoPro in his head* trajectory. These graphics show the virtual trajectory of the central pixel after successively applying the transformations. We show the trajectory for each smoothing strategy. The left graphic show the x component and the right the y component of the tracked point.

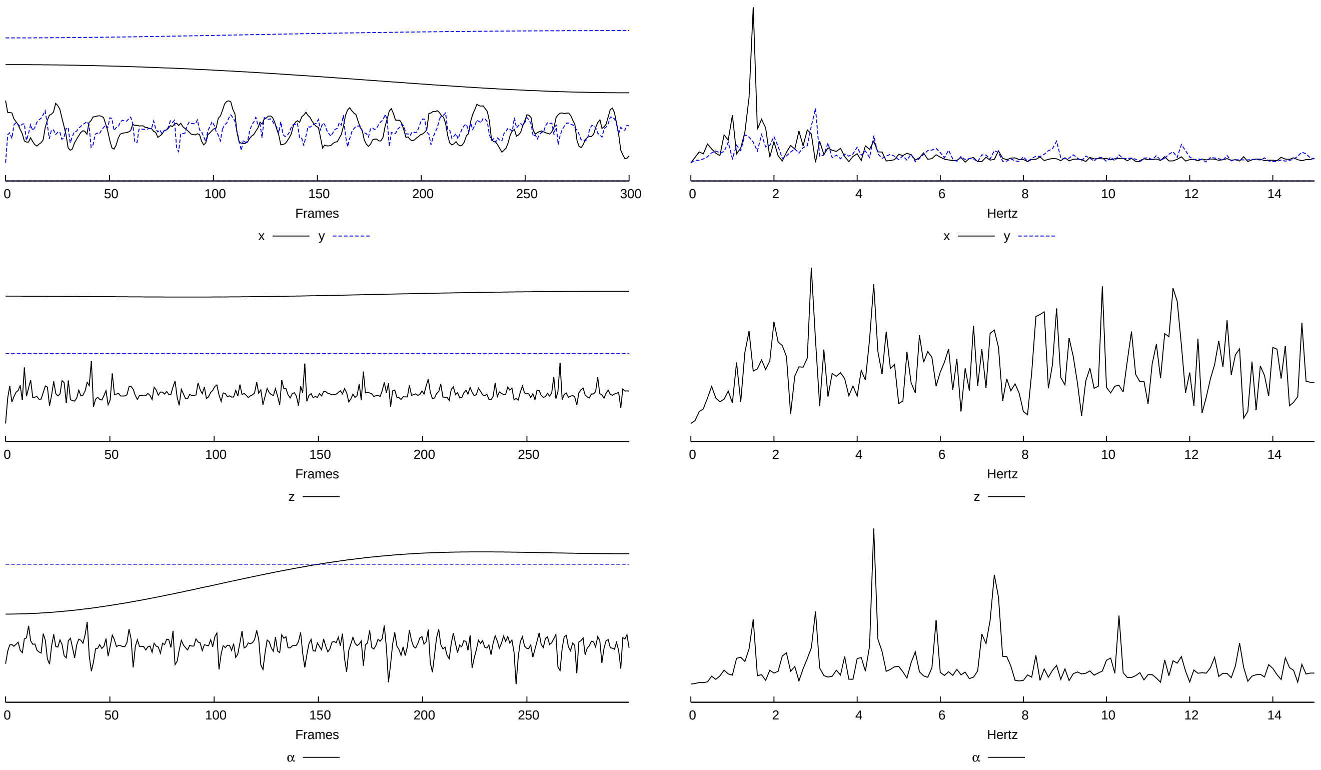


Figure 6: *Man running with a GoPro in his head* linear scale–space analysis. In the top–left figure, we show the scale–space corresponding to the trajectory of the central point; in the middle–left figure, we show the scale–space of the zoom signal; and, in the bottom–left figure, the scale–space of the rotation signal. In these figures, the graphics above show the original signal smoothed with $\sigma := 80$ and the graphics in the bottom show the difference between the smoothed signal with $\sigma := 10$ and the original one. The right figures depict the DFT signals of these graphics.

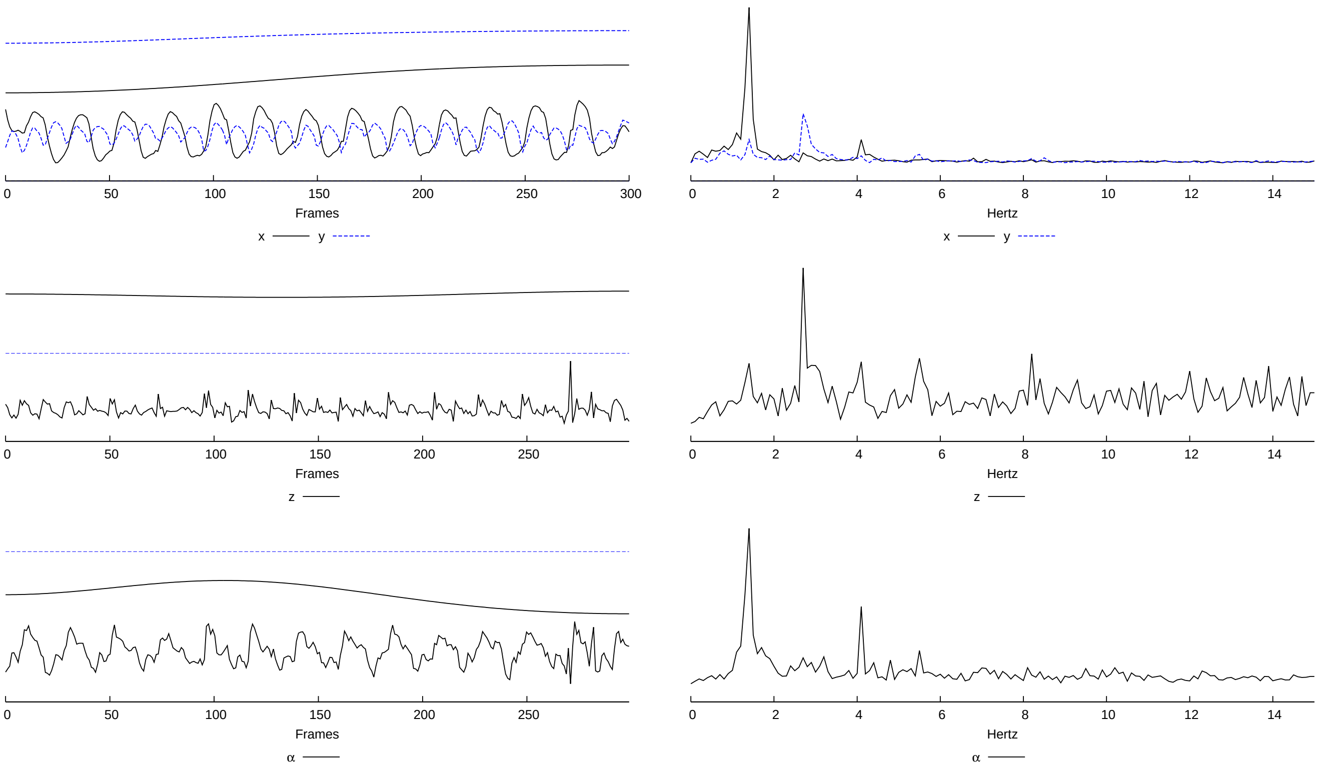


Figure 7: *Man running with a GoPro in his chest* linear scale–space analysis. In the top–left figure, we show the scale–space corresponding to the trajectory of the central point; in the middle–left figure, we show the scale–space of the zoom signal; and, in the bottom–left figure, the scale–space of the rotation signal. In these figures, the graphics above show the original signal smoothed with $\sigma := 80$ and the graphics in the bottom show the difference between the smoothed signal with $\sigma := 10$ and the original one. The right figures depict the DFT signals of these graphics.

around 1.5 Hz, which denotes the oscillation from left to right. In the y component, there is a peak at 3 Hz corresponding to the vertical oscillation.

In the zoom figure, we observe that the graphics present fast periodic motions which are, on average, above the blue line, corresponding to a static zoom. There is a steady zoom–in due to the forward motion of the camera. On the other hand, the oscillations are probably due to the balancing of the head while running.

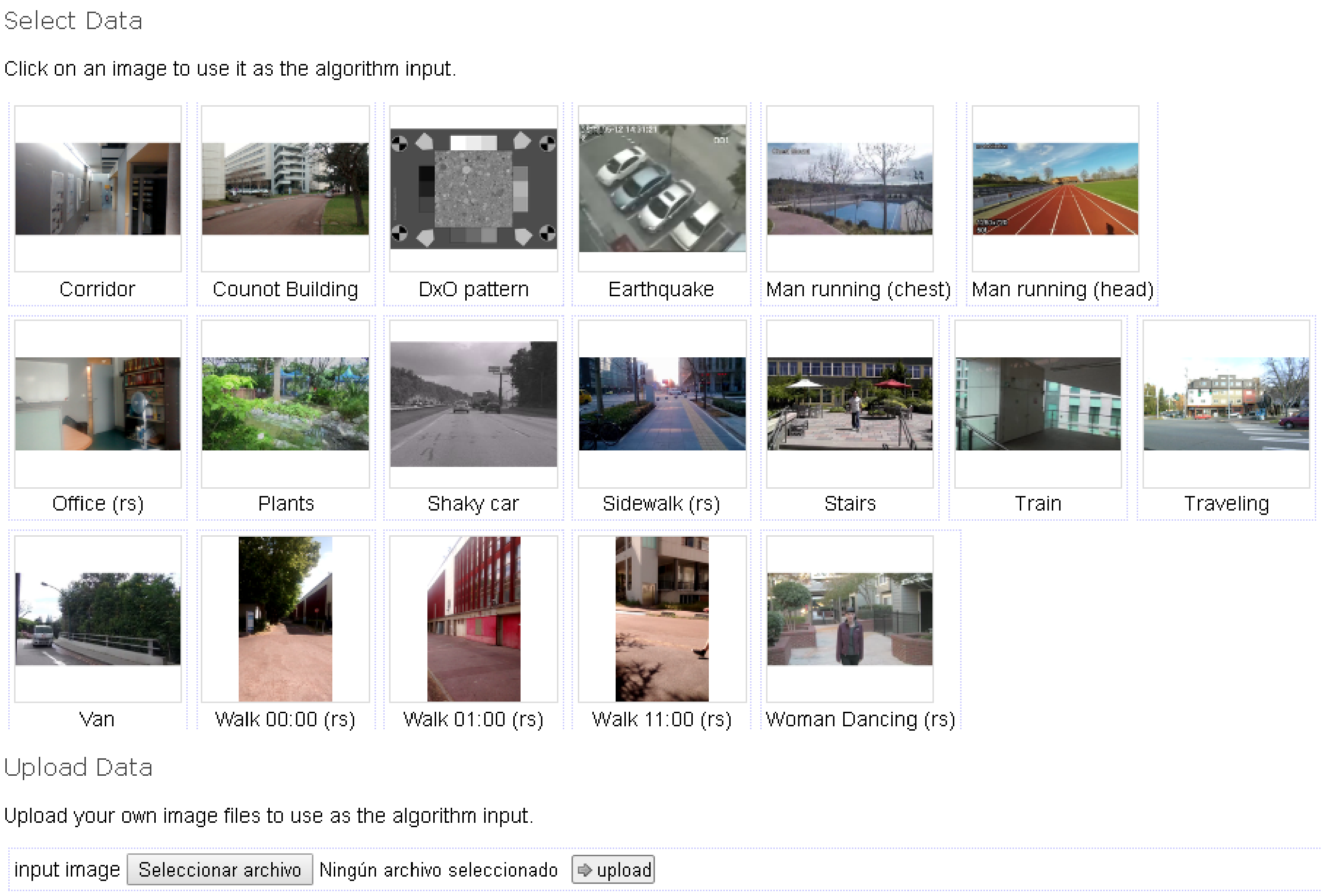
We can compare the previous analysis with the analysis video of a man running with a camera in his chest. The results are similar to the previous experiment, although the shake is more important than when the camera is fixed to the head. This is reasonable as persons tend to stabilize their head when they run or walk.

The trajectory, in the linear scale–space graphics of Figure 7, seems to be more regular and periodic; see the two graphics at the bottom. This is due to the fact that the chest is more rigid and better represents the motion of the run. The peaks are also situated about 1.5 Hz for the horizontal displacement and slightly above 3Hz for the vertical one.

The zoom and rotation linear scale–spaces are also similar to the previous experiment, although the rotation seems to be more regular and aligned with the horizontal displacement. This may be produced by the motion from side to side of the chest while moving the arms, which is synchronized with the motion of the legs.

3.4 Online Demo

The online demo allows selecting or uploading a video in several formats. There are some predefined videos with different amounts of shakiness and behaviors. We have also included several challenging videos which are difficult to stabilize, see Figure 8.



Select Data

Click on an image to use it as the algorithm input.

Upload Data

Upload your own image files to use as the algorithm input.

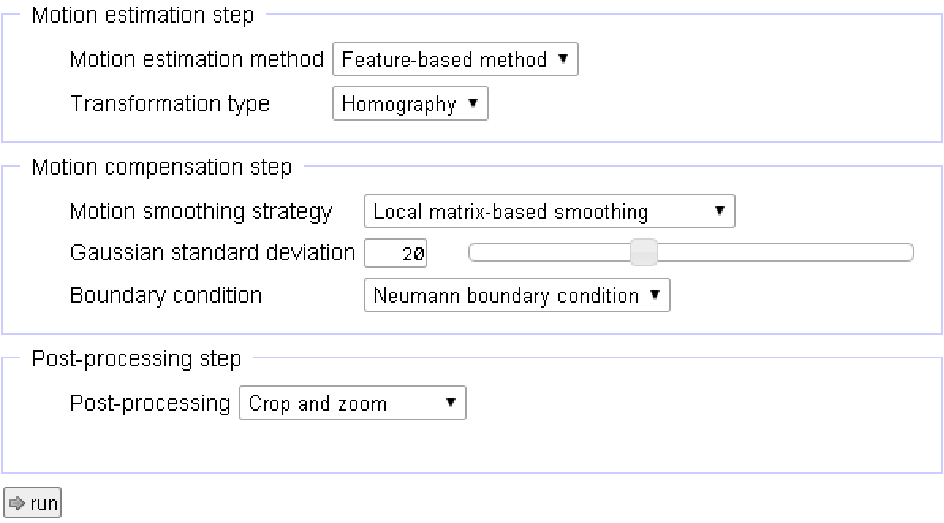
input image Seleccionar archivo Ningún archivo seleccionado

Figure 8: Input window of the online demo.

If the user uploads a video, it is converted to MPEG–4, its size is reduced to 720 pixels width, and the duration is limited to 10 seconds, using the `avconv` program.

The parameter window is divided in three parts, as shown in Figure 9. In the first part, the user can choose the motion estimation method —direct or feature–based method— and the type of transformation. For the featured–based method, only homographies can be chosen. In the second part, the user can select any of the motion smoothing strategies proposed in Section 2, the standard deviation of the smoothing kernel and the boundary conditions to be applied. The third part corresponds to the post–processing step, where the user can select the Crop&Zoom algorithms proposed in Section 2.3 or leave the empty regions untouched choosing the *No post–processing* option. This option is recommended for several default videos, since the crop operation can eliminate a large portion of the contents.

The result window shows the original and stabilized videos and provides several buttons to manage both videos together, as in Figure 10. The user can play both videos simultaneously to appreciate the differences with respect to the original input. This window also shows several curves related to the trajectory and scale–space analysis that we have used in the experiments, see Figure 11. In particular, we show the trajectory of the original and smoothed sequences, the scale–space of the trajectory, rotation and zoom, and their corresponding DFT signals, as defined in Section 3.2. The



Motion estimation step

Motion estimation method: Feature-based method ▾

Transformation type: Homography ▾

Motion compensation step

Motion smoothing strategy: Local matrix-based smoothing ▾

Gaussian standard deviation: 20

Boundary condition: Neumann boundary condition ▾

Post-processing step

Post-processing: Crop and zoom ▾

Figure 9: Input parameters of the online demo.

parameters chosen in the previous step and a link to the output of the program are shown at the bottom of the window.

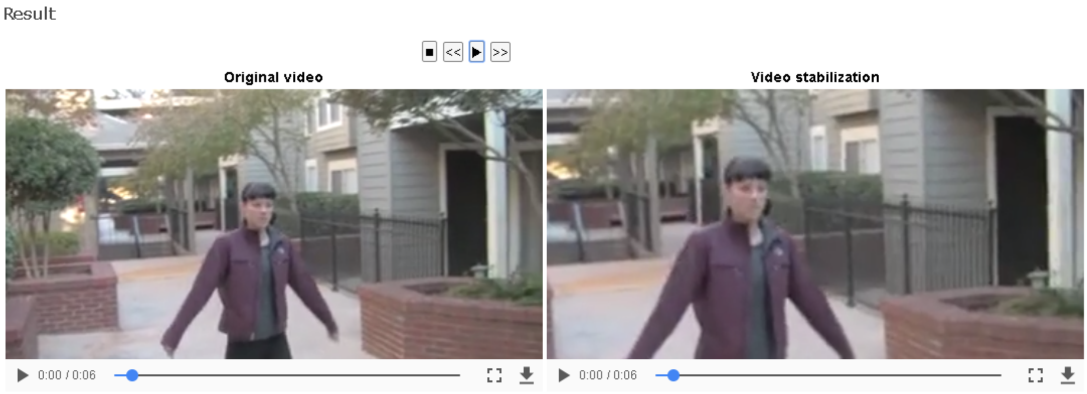


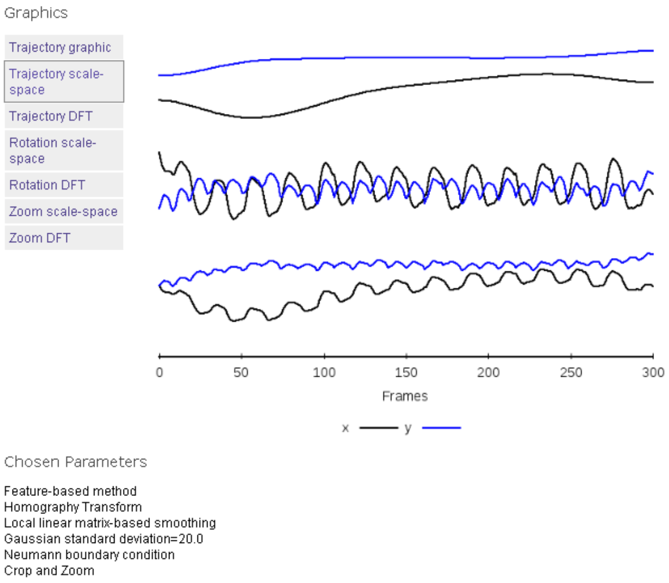
Figure 10: Output videos of the online demo.

4 Conclusion

We have described motion compensation strategies for video stabilization and characterized the different types of boundary conditions for the involved smoothing.

The smoothing strategies are divided in compositional and additive approaches. The former are further divided in global methods, which compose the transformations from a given reference frame, and local methods, which compose the transformations in a neighborhood around each image. Two additional linear methods work directly on the coefficients of the transformations or on the virtual trajectories obtained as integrals of the instant velocity of fixed frame points.

Analyzing the frequency patterns and the smooth tendency in the scale-space of the virtual trajectories yields crucial temporal information such as frequency peaks linked to periodic ego-motions and a smoothed evaluation of the movement forward and of the rotations.



Chosen Parameters

Feature-based method
Homography Transform
Local linear matrix-based smoothing
Gaussian standard deviation=20.0
Neumann boundary condition
Crop and Zoom

Figure 11: Output graphics of the online demo.

Acknowledgements

Work partly founded by BPIFrance and Région Ile de France, in the framework of the FUI 18 Plein Phare project, the Office of Naval research (ONR grant N00014-14-1-0023), ANR-DGA project ANR-12-ASTR-0035, and by the Spanish Ministry of Economy, Industry and Competitiveness through the research project TIN2017-89881-R.

Image Credits

All images by the author (license CC-BY-SA) except:

Shuaicheng et al. [40].

Grundmann et al. [18].

References

[1] H. Alt, D. Hsu, and J. Snoeyink, *Computing the largest inscribed isothetic rectangle*, in Proceedings of the Canadian Conference on Computational Geometry (CCCG), 1995, pp. 67–72.

[2] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro, *A variational framework for exemplar-based image inpainting*, International Journal of Computer Vision, 93 (2011), pp. 319–347. https://doi.org/10.1007/s11263-010-0418-7.

[3] S. Baker, R. Gross, T. Ishikawa, and I. Matthews, *Lucas-kanade 20 years on: A unifying framework: Part 2*, tech. report, Robotics Institute, Carnegie Mellon University, 2003.

[4] S. Baker and I. Matthews, *Lucas-Kanade 20 years on: A unifying framework*, International Journal of Computer Vision, 56 (2004), pp. 221–255. https://doi.org/10.1023/B:VISI.0000011205.11775.fd.

[5] P. Bouthemy, M. Gelgon, and F. Ganansia, *A unified approach to shot change detection and camera motion characterization*, IEEE Transactions on Circuits and Systems for Video Technology, 9 (1999), pp. 1030–1044. https://doi.org/10.1109/76.795057.

[6] M. Brown and D.G Lowe, *Recognising panoramas.*, in Proceedings of the International Conference on Computer Vision, vol. 3, 2003, p. 1218. https://doi.org/10.1109/ICCV.2003.1238630.

[7] C. Buehler, M. Bosse, and L. McMillan, *Non-metric image-based rendering for video stabilization*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, 2001, pp. II–609–II–614 vol.2. https://doi.org/10.1109/CVPR.2001.991019.

[8] B-Y. Chen, K-Y. Lee, W-T. Huang, and J-S. Lin, *Capturing intention-based full-frame video stabilization*, Computer Graphics Forum, 27 (2008), pp. 1805–1814. http://dx.doi.org/10.1111/j.1467-8659.2008.01326.x.

[9] A. Criminisi, P. Pérez, and K. Toyama, *Region filling and object removal by exemplar-based image inpainting*, IEEE Transactions on Image Processing, 13 (2004), pp. 1200–1212. http://dx.doi.org/10.1109/TIP.2004.833105.

[10] K. Daniels, V. Milenkovic, and D. Roth, *Finding the largest area axis-parallel rectangle in a polygon*, Computational Geometry, 7 (1997), pp. 125 – 148. http://dx.doi.org/10.1016/0925-7721(95)00041-0.

[11] J. Delon, *Movie and video scale-time equalization application to flicker reduction*, IEEE Transactions on Image Processing, 15 (2006), pp. 241–248. https://doi.org/10.1109/TIP.2005.860328.

[12] P. Fischer and K-U. Höffgen, *Computing a maximum axis-aligned rectangle in a convex polygon*, Information Processing Letters, 51 (1994), pp. 189–193. http://doi.org/10.1016/0020-0190(94)00079-4.

[13] P-E. Forssén and E. Ringaby, *Rectifying rolling shutter video from hand-held devices*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 507–514. https://doi.org/10.1109/CVPR.2010.5540173.

[14] P. Getreuer, *A Survey of Gaussian Convolution Algorithms*, Image Processing On Line, 3 (2013), pp. 286–310. http://dx.doi.org/10.5201/ipol.2013.87.

[15] A. Goldstein and R. Fattal, *Video stabilization using epipolar geometry*, ACM Transactions on Graphics (TOG), 31 (2012), p. 126. https://doi.org/10.1145/2231816.2231824.

[16] G. Greiner and K. Hormann, *Efficient clipping of arbitrary polygons*, ACM Transactions on Graphics (TOG), 17 (1998), pp. 71–83. http://doi.org/10.1145/274363.274364.

[17] M. Grundmann, V. Kwatra, D. Castro, and I. Essa, *Calibration-free rolling shutter removal*, in Proceedings of the IEEE International Conference on Computational Photography (ICCP), 2012, pp. 1–8. https://doi.org/10.1109/ICCPhot.2012.6215213.

[18] M. Grundmann, V. Kwatra, and I. Essa, *Auto-directed video stabilization with robust L1 optimal camera paths*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 225–232. https://doi.org/10.1109/CVPR.2011.5995525.

[19] B.t. Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, eds., *Scale-Space Theory in Computer Vision*, vol. 1252, Springer Science & Business Media, Lecture Notes in Computer Science, 1997.

[20] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt, *Real-time scene stabilization and mosaic construction*, in Proceedings of the Second IEEE Workshop on Applications of Computer Vision, Dec 1994, pp. 54–62. https://doi.org/10.1109/ACV.1994.341288.

[21] C. Harris and M. Stephens, *A combined corner and edge detector*, in Proceedings of the 4th Alvey Vision Conference, vol. 15, 1988, p. 50.

[22] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, second ed., 2004.

[23] J. Hedborg, P. E. Forssén, M. Felsberg, and E. Ringaby, *Rolling shutter bundle adjustment*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2012, pp. 1434–1441. https://doi.org/10.1109/CVPR.2012.6247831.

[24] T. Huynh, M. Fritz, and B. Schiele, *Discovery of activity patterns using topic models*, in Proceedings of the 10th international conference on Ubiquitous computing, 2008, pp. 10–19. https://doi.org/10.1145/1409635.1409638.

[25] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, *Digital video stabilization and rolling shutter correction using gyroscopes*, tech. report, Stanford, 2011.

[26] J-G. Kim, H-S. Chang, J. Kim, and H-M. Kim, *Efficient camera motion characterization for mpeg video indexing*, in Proceedings of IEEE International Conference on Multimedia and Expo (ICME), vol. 2, 2000, pp. 1171–1174. https://doi.org/10.1109/ICME.2000.871569.

[27] S-J. Ko, S-H. Lee, and K-H. Lee, *Digital image stabilizing algorithms based on bit-plane matching*, IEEE Transactions on Consumer Electronics, 44 (1998), pp. 617–622. https://doi.org/10.1109/30.713172.

[28] J.J. Koenderink, *The structure of images*, Biological Cybernetics, 50 (1984), pp. 363–370.

[29] D. Kundur and D. Hatzinakos, *Blind image deconvolution*, IEEE Signal Processing Magazine, 13 (1996), pp. 43–64. https://doi.org/10.1109/79.489268.

[30] K-Y. Lee, Y-Y. Chuang, B-Y. Chen, and M. Ouhyoung, *Video stabilization using robust feature trajectories*, in Proceedings of the IEEE 12th International Conference on Computer Vision, 2009, pp. 1397–1404. https://doi.org/10.1109/ICCV.2009.5459297.

[31] T. Lindeberg, *Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention*, International Journal of Computer Vision, 11 (1993), pp. 283–318.

[32] ——, *Scale-space theory: A basic tool for analyzing structures at different scales*, Journal of Applied Statistics, 21 (1994), pp. 225–270.

[33] ——, *Scale-space theory in computer vision*, vol. 256, Springer Science & Business Media, 1994.

[34] A. Litvin, J. Konrad, and W.C Karl, *Probabilistic video stabilization using Kalman filtering and mosaicing*, in Proceedings of SPIE Electronic Imaging, 2003, pp. 663–674. https://doi.org/10.1117/12.476436.

[35] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, *Content-preserving warps for 3D video stabilization*, ACM Transactions on Graphics (TOG), 28 (2009), p. 44. https://doi.org/10.1145/1531326.1531350.

[36] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, *Subspace video stabilization*, ACM Transactions on Graphics (TOG), 30 (2011), p. 4. https://doi.org/10.1145/1899404.1899408.

[37] S. Liu, P. Tan, L. Yuan, J. Sun, and B. Zeng, *Meshflow: Minimum latency online video stabilization*, in Proceedings of the 14th European Conference on Computer Vision (ECCV), Springer International Publishing, 2016, pp. 800–815. https://doi.org/10.1007/978-3-319-46466-4_48.

[38] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun, *Video stabilization with a depth camera*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 89–95. https://doi.org/10.1109/CVPR.2012.6247662.

[39] S. Liu, L. Yuan, P. Tan, and Jian Sun, *Bundled camera paths for video stabilization*, ACM Transactions on Graphics (TOG), 32 (2013), p. 78. https://doi.org/10.1145/2461912.2461995.

[40] S. Liu, L. Yuan, P. Tan, and J. Sun, *Steadyflow: Spatially smooth optical flow for video stabilization*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014, pp. 4209–4216. https://doi.org/10.1109/CVPR.2014.536.

[41] D.G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 60 (2004), pp. 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[42] B.D. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision*, in Proceedings of the 7th International Joint Conference on Artificial intelligence, vol. 81, 1981, pp. 674–679.

[43] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H-Y. Shum, *Full-frame video stabilization with motion inpainting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 1150–1163. https://doi.org/10.1109/TPAMI.2006.141.

[44] L. Moisan, P. Moulon, and P. Monasse, *Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers*, Image Processing On Line, 2 (2012), pp. 56–73. http://dx.doi.org/10.5201/ipol.2012.mmm-oh.

[45] C. Morimoto and R. Chellappa, *Fast electronic digital image stabilization*, in Proceedings of the 13th International Conference on Pattern Recognition, vol. 3, 1996, pp. 284–288. https://doi.org/10.1109/ICPR.1996.546956.

[46] ——, *Evaluation of image stabilization algorithms*, in Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 5, 1998, pp. 2789–2792. https://doi.org/10.1109/ICASSP.1998.678102.

[47] H. Pirsiavash and D. Ramanan, *Detecting activities of daily living in first-person camera views*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 2847–2854. https://doi.org/10.1109/CVPR.2012.6248010.

[48] E. Ringaby and P-E. Forssén, *Efficient video rectification and stabilisation for cell-phones*, International Journal of Computer Vision, 96 (2012), pp. 335–352. https://doi.org/10.1007/s11263-011-0465-8.

[49] J. Sánchez, *The Inverse Compositional Algorithm for Parametric Registration*, Image Processing On Line, 6 (2016), pp. 212–232. https://doi.org/10.5201/ipol.2016.153.

[50] ——, *Midway Video Equalization*, Image Processing On Line, 7 (2017), pp. 65–80. https://doi.org/10.5201/ipol.2017.181.

[51] J. Shi and C. Tomasi, *Good features to track*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 1994, pp. 593–600.

[52] B.M. Smith, L. Zhang, H. Jin, and A. Agarwala, *Light field video stabilization*, in Proceedings of the IEEE 12th International Conference on Computer Vision, 2009, pp. 341–348. https://doi.org/10.1109/ICCV.2009.5459270.

[53] J. Sporring, M. Nielsen, L. Florack, and P. Johansen, *Gaussian scale-space theory*, vol. 8, Springer Science & Business Media, 2013. https://doi.org/10.1007/978-94-015-8802-7.

[54] E.H. Spriggs, F. De La Torre, and M. Hebert, *Temporal segmentation and activity classification from first-person sensing*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2009, pp. 17–24. https://doi.org/10.1109/CVPRW.2009.5204354.

[55] I.E Sutherland and G.W. Hodgman, *Reentrant polygon clipping*, Communications of the ACM, 17 (1974), pp. 32–42.

[56] R. Szeliski, *Computer vision algorithms and applications*, Springer, London; New York, 2011. http://dx.doi.org/10.1007/978-1-84882-935-0.

[57] R. Szeliski and H-Y. Shum, *Creating full view panoramic image mosaics and environment maps*, in Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, 1997, pp. 251–258. https://doi.org/10.1145/258734.258861.

[58] M. Trajković and M. Hedley, *Fast corner detection*, Image and Vision Computing, 16 (1998), pp. 75–87. https://doi.org/10.1016/S0262-8856(97)00056-5.

[59] B.R. Vatti, *A generic solution to polygon clipping*, Communications of the ACM, 35 (1992), pp. 56–63. http://dx.doi.org/10.1145/129902.129906.

[60] F. Vella, A. Castorina, M. Mancuso, and G. Messina, *Digital image stabilization by adaptive block motion vectors filtering*, IEEE Transactions on Consumer Electronics, 48 (2002), pp. 796–801. https://doi.org/10.1109/TCE.2002.1037077.

[61] Y-S. Wang, F. Liu, P-S. Hsu, and T-Y. Lee, *Spatially and temporally optimized video stabilization*, IEEE Transactions on Visualization and Computer Graphics, 19 (2013), pp. 1354–1361. https://doi.org/10.1109/TVCG.2013.11.

[62] Y. Wexler, E. Shechtman, and M. Irani, *Space-time completion of video*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 463–476. http://dx.doi.org/10.1109/TPAMI.2007.60.

[63] A. Witkin, *Scale-space filtering: A new approach to multi-scale description*, in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 9, 1984, pp. 150–153.