Published in Image Processing On Line on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.YYYY.XXXXXXXX

Estimation of a video sequence camera motion

Léo Girardin[1] & David Lafontaine[2]

[1]Ecole Normale Supérieure de Cachan (leo.girardin@ens-cachan.fr)
[2]Ecole Normale Supérieure de Cachan (david.lafontaine@ens-cachan.fr)

1 Abstract

The problem of camera motion estimation appears in many image processing problems, such as augmented reality, motion compensation, or estimation of depths in a video scene. The present work is a study of a camera motion model proposed by C. Jonchéry, which is based on a parametrization of the camera motion in the pinhole camera model and the estimation of the optical flow between two consecutive images in a video sequence under the core hypothesis of a uniform depth. Two algorithms are implemented and compared: Jonchéry's method and another method that involves the estimation of the optical flow using Horn-Schunck algorithm. Both sequences presenting simulated motions and real ones are processed in the tests in order to evaluate the accuracy, robustness and limitations of Jonchéry's method.

References

[1] B.K. Horn and B.G. Shunk. Determining optical flow. *Artificial Intelligence*, 1981.

[2] Claire Jonchéry. *Estimation d'un mouvement de caméra et problèmes connexes*. PhD thesis, ENS Cachan, 2007.

[3] Enric Meinhardt-Llopis and Javier Sánchez Pérez. Horn-schunk optical flow with a multi-scale strategy. *IPOL*, 2001.

[4] M.R Osborne. Finite algorithms in optimization and data analysis. *John Wiley*, 1985.

2 Context

We consider two consecutive images f and g taken in a video sequence and would like to estimate the motion of the camera between them.

There are three main categories of methods to estimate such a motion : direct ones, which use directly the image information without optical flow computations or point tracking, discrete ones, which track singular points, and differential ones, which compute the optical flow before the motion. Discrete methods are more accurate with large motions, whereas differential and direct ones are adapted to smaller motions.

The method developed by Jonchéry in [2] is a direct method, which is expected to run faster than differential and discrete ones. A differential method is also implemented as a reference for evaluating performances of Jonchéry's model.

3 Model

3.1 Pinhole camera model

We choose to work with the pinhole camera model which considers that map projections are linear in projective coordinates. In figure 1 [1],

- the optical center C corresponds to the camera position
- the image plane $\mathcal{R}$ is where the space is projected through the aperture of the camera
- (i, j, k) is an orthonormal coordinate system where k is pointing in the direction orthogonal to $\mathcal{R}$
- the optical axis is the line (C, k)
- the principal point c is the intersection of the optical axis and the image plane
- the focal length f_c is the distance between C and c
- a projection line is a ray starting at C which intersects $\mathcal{R}$
- a real point $M = (X, Y, Z)$ has for projection on $\mathcal{R}$ the point $m = (x, y) = (CM) \cap \mathcal{R}$

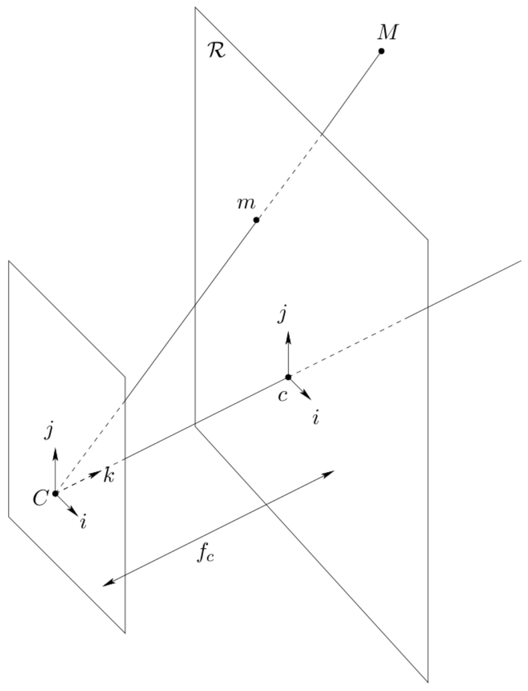

Figure 1: Pinhole camera model

[1] image from [2]

The projection function is defined as

$$\begin{array}{rccc}\pi : & \mathbb{R}^3 & \to & \mathbb{R}^2 \\ & (X, Y, Z) & \mapsto & (x, y)\end{array}$$

where $x = f_c \frac{X}{Z}$ and $y = f_c \frac{Y}{Z}$. So two points of $\mathbb{R}^3$ whose coordinates are proportional in (C, i, j, k) share the same image on $\mathcal{R}$, and we cannot determine the depth Z of M if we only know m on a single image.

We assume that the camera pixels are squares and that (c, i, j) is the pixel coordinate system, so there is no need to change the coordinates in the camera.

3.2 Motion parameters

Given this model, we want to parametrize camera motions. We consider rigid displacements, defined as below:

Definition. *A rigid displacement is a transformation from $\mathbb{R}^3$ to $\mathbb{R}^3$ which respects angles and distances. It is the composition of a rotation and a translation. Their set is the group $SE(3)$.*

We denote D the camera motion between images f and g. We define $R = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} \in SO_3(\mathbb{R})$ and $t = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$ such that $D = (R, t) \in SE(3)$.

A point $M \in \mathbb{R}^3$ is projected onto $(x, y) \in f$ and then onto $(x', y') \in g$.

For a unitary focal length, we have the following equalities, proved in [2]:

$$\begin{cases} x' = \dfrac{a_1 x + a_2 y + a_3 - \left\langle \frac{t}{Z(x,y)}, R(i) \right\rangle}{c_1 x + c_2 y + c_3 - \left\langle \frac{t}{Z(x,y)}, R(k) \right\rangle} \\ \\ y' = \dfrac{b_1 x + b_2 y + b_3 - \left\langle \frac{t}{Z(x,y)}, R(j) \right\rangle}{c_1 x + c_2 y + c_3 - \left\langle \frac{t}{Z(x,y)}, R(k) \right\rangle} \end{cases} \tag{1}$$

We denote by $Z(x, y)$ the depth of the point which is projected in (x, y).

For another focal length, it suffices to replace x, x', y, y', by $\frac{x}{f_c}$, $\frac{x'}{f_c}$, $\frac{y}{f_c}$, $\frac{y'}{f_c}$.

Note that for any λ, $(\lambda t, \lambda Z)$ leads to the same system (1). For this reason, we can only estimate the direction of the translation. From now on, let Z_0 be the mean depth of the scene and $\tilde{t} = \frac{t}{Z_0}$.

Let us parametrize R and $\tilde{t}$ in a comfortable form.

We decompose R in two rotations, $R = R_2 \circ R_1$, where R_1 has its axis in the plane (c, i, j) and leads to a purely projective deformation of f, while R_2 has $R_1(k)$ for axis and leads to a rotation of f. Let $\Delta \subset (C, i, j)$ be the axis of R_1, θ the angle between i and Δ, and α the angle of rotation of R_1. β is the angle of rotation of R_2, as illustrated in figure 2[2].

The normalized translation $\tilde{t}$ is parametrized by A, B, C, where $\tilde{t} = (-A, -B, -C)$ in $(R(i), R(j), R(k))$. The number C (not to be confused with the optical center) corresponds to a homothetic transformation of f and A and B to a translation of it.

From now on, we set $\Theta = (\theta, \alpha, \beta, A, B, C)$ which codifies the motion D.

[2]image from [2]

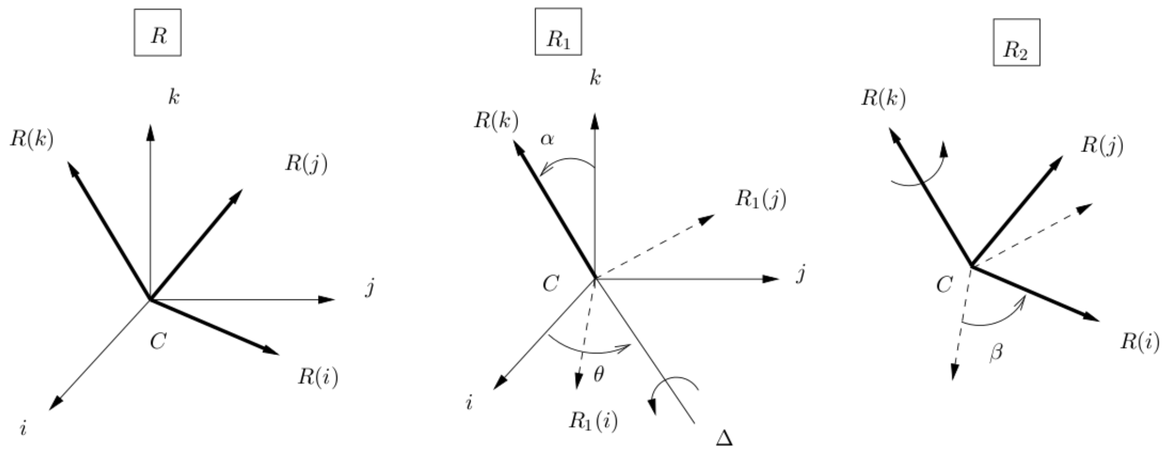

Figure 2: R and its decomposition

3.3 Optical flow approximation

3.3.1 Hypotheses

First, let us define the swelling between two consecutive images:

Definition. *We call swelling related to* $D = (\begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}, t)$ *and to the depth function* Z *the number*

$$S_{D,Z} = \max_{(x,y)\in K} \left|\frac{1}{c_1 x + c_2 y + c_3 - \langle \frac{t}{Z(x,y)}, R(k)\rangle}\right|$$

Now, in order to approximate the optical flow between f and g, we need to do some hypotheses on the motion.

Hypothesis 1. *There exists* S_{max} *such that, for every deplacement* D *and depth function* Z *we are working with, we have*

$$S_{D,Z} \leq S_{max}$$

This first hypothesis states that the swelling is bounded. It corresponds to the fact that the displacement from k to $R(k)$ is very small in an effective camera motion.

The next hypothesis expresses that the displacement of an apparent point is not too big between the two images.

Hypothesis 2. *We assume that*

$$\max_{(x,y)\in K} \max(|x' - x|, |y' - y|) \leq \frac{L}{2}$$

where K *is the set of points* (x, y) *for which* (x', y') *are defined, and L is the larger dimension of the images.*

In practice, these two hypotheses are always verified because the motion between two images in a real video sequence is very small.

Finally, we wish to subsitute $Z(x, y)$ by a uniform depth Z_0 in the formula (1). In contrast with the first two hypotheses, the one we are presenting now is really limiting applicationwise.

Hypothesis 3. *Let us fix* $\epsilon > 0$. *We have*

1. $\left(\frac{1}{Z_{inf}} - \frac{1}{Z_{sup}}\right) ||t||(L+1)S_{max} \leq 2\epsilon$

2. $\frac{1}{Z_{inf}} - \frac{\epsilon}{||t||(L+1)S_{max}} \leq \frac{1}{Z_0} \leq \frac{1}{Z_{sup}} + \frac{\epsilon}{||t||(L+1)S_{max}}$

where $Z_{inf} = \inf_{(x,y)\in K} Z(x,y)$ *and* $Z_{sup} = \sup_{(x,y)\in K} Z(x,y)$*.*

3.3.2 Approximation

Theorem. *Under hypotheses 1, 2, 3 we have*

$$|x' - x''| \leq \epsilon$$
$$|y' - y''| \leq \epsilon$$

where

$$\begin{cases} x'' = \dfrac{a_1 x + a_2 y + a_3 - \left\langle \frac{t}{Z_0}, R(i) \right\rangle}{c_1 x + c_2 y + c_3 - \left\langle \frac{t}{Z(x,y)}, R(k) \right\rangle} \\ \\ y'' = \dfrac{b_1 x + b_2 y + b_3 - \left\langle \frac{t}{Z(x,y)}, R(j) \right\rangle}{c_1 x + c_2 y + c_3 - \left\langle \frac{t}{Z_0}, R(k) \right\rangle} \end{cases} \tag{2}$$

Given this result, we subsitute $Z(x,y)$ by Z_0 in the formula (1) (and still denote the coordinates (x', y')). We express Θ in function of R and t, and after a limited development, we get the following estimate u_Θ of the true optical flow u for a unitary focal length:

$$\begin{pmatrix} x' - x \\ y' - y \end{pmatrix} \simeq \begin{pmatrix} A - \alpha \sin\theta \\ B + \alpha\cos\theta \end{pmatrix} + \begin{pmatrix} -C & \beta \\ -\beta & -C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -\alpha\sin\theta & \alpha\cos\theta & 0 \\ 0 & -\alpha\sin\theta & \alpha\cos\theta \end{pmatrix} \begin{pmatrix} x^2 \\ xy \\ y^2 \end{pmatrix}$$

This estimate is the core of the algorithm that follows. Its error is controled by the following theorem.

Theorem. *Under hypotheses 1, 2, 3, we have, for every* (x,y) *in* K

$$||u(x,y) - u_\Theta(x,y)|| \leq T(S_{max}, L, \alpha, A, C) + T(S_{max}, L, \alpha, B, C) + 2\epsilon$$

and

$$\begin{aligned} T = \quad & S_{max}(L^3 \tfrac{\alpha^2}{2} + L^2(|C\alpha| + \tfrac{|\beta\alpha|}{2} + \tfrac{|\alpha|^3}{3}) \\ & + L(\tfrac{\alpha^2}{4}(6 + 3|\beta| + |C-1|) + |A\alpha| + \tfrac{\beta C}{2} + \tfrac{\beta^2}{4} + \tfrac{C^2}{2} + \tfrac{|\beta|^3}{12}) \\ & + |\alpha|(\tfrac{\beta^2}{2} + |\beta| + |C| + \tfrac{|\alpha A|}{2} + \tfrac{2\alpha^2}{3}) \\ & + |AC| \;) \end{aligned}$$

Theses two theorems are proved in [2].

4 Algorithms

4.1 A differential algorithm

If the optical flow is known, a first idea to compute an approximation of the camera motion is to minimize over Θ the quantity

$$E(\Theta) := \sum_{(x,y)\in S} (u_\Theta(x,y) - u(x,y))^2$$

The optical flow u can be computed with any classical method. In this work, we use Horn-Schunck method. For more details, refer to [1] and [3].

Setting

$$\begin{aligned} c_1 &= \quad A - \alpha \sin\theta \\ c_2 &= \quad B + \alpha \cos\theta \\ a_1 &= \quad -C \\ a_2 &= \quad \beta \\ q_1 &= \quad -\alpha \sin\theta \\ q_2 &= \quad \alpha \cos\theta \end{aligned}$$

the function u_Θ is linear in $(a_1, a_2, c_1, c_2, q_1, q_2)$, thus we can use a classical least-squares method: the function $F(x) = x^t A x + b \cdot x + c$, where A is symmetric positive definite and b a vector, has an unique minimum $x_0 = -\frac{1}{2}A^{-1}b$.

4.2 Jonchéry's algorithm

Jonchéry's idea is to minimize in mean over Θ the quantity

$$DF_\Theta(x,y) = g((x,y) + u_\Theta(x,y)) - f(x,y) + \xi$$

where ξ is an lighting constant.

To perform this, an incremental least-squares scheme is used. We set

$$\begin{aligned} \Theta_0 &= \quad 0 \\ \xi_0 &= \quad 0 \end{aligned}$$

and

$$\begin{aligned} \Theta_{k+1} &= \quad \Theta_k + \Delta\Theta_k \\ \xi_{k+1} &= \quad \Theta_k + \Delta\xi_k \end{aligned}$$

By a limited development at first order, we obtain

$$DF_{\Theta_{k+1},\xi_{k+1}} \simeq r_{\Delta\Theta_k,\Delta\zeta_k} := DF_{\Theta_k,\zeta_k}(x,y) + \nabla g((x,y) + u_{\Theta_k}(x,y)) \cdot u_{\Delta\Theta_k}(x,y) + \Delta\zeta_k$$

To compute the increment $(\Delta\Theta_k, \Delta\xi_k)$, given (Θ_k, ξ_k), we minimize over $(\Delta\Theta_k, \Delta\xi_k)$ the quantity

$$\sum_{(x,y)\in S} r^2_{\Delta\Theta_k,\Delta\zeta_k}$$

Because $r_{\Delta\Theta_k,\Delta\zeta_k}$ is linear in $(a_1, a_2, c_1, c_2, q_1, q_2)$, this can be done once more by a classical least-squares method. Osborne proved in [4] that such a scheme converges.

We iterate the computation until convergence of (Θ_k, ξ_k). But there is no obvious stopping criterion, because as we observed after some experimentations, the increments begin to oscillate around zero after enough iterations, with an amplitude that depends on the image and on the motion. We empirically founded the following criterion: at step N, we stop the iterations if

$$|\sum_{k=N-5}^{N} \Delta a_{1k}| \leq \frac{|\Delta a_{1N} - \Delta a_{1N-1}|}{2}$$

The idea is to detect when the increments begin to oscillate.

5 Implementation

We have implemented both algorithms in language C.

To compute the minimum of the least-squares problems, $-\frac{1}{2}A^{-1}b$, we used the Choleski decomposition of a symetric defined positive matrix: $A = L^tL$, where L is upper triangular. After L is computed, we just have to solve two triangular linear systems.

To avoid border problems, we work only with the center of the image: we define a margin equal to ten percents of the smaller dimension and we trim each side of the image by a number of pixels equal to this margin.

To improve the computation time, we subsample images larger than 400 pixels by merging four adjacent pixels into one.

6 Online Demo

In the demo, both algorithms are executed simultaneously. One can use its own pair of images or a proposed one. The focal length is to be set to its real value in pixels. If it is unknown, the half of the width of the images, which corresponds to an angle of view of ninety degrees, should give satisfying results. The angles are given in radians. If the motion between the two images is unknown, the motion compensation computed with the parametrized optical flow should give an idea of the accuracy of the result, even if our algorithm is not an optical flow one.

7 Experimentations

7.1 Images verifing all hypothesis

For images verifing all hypotheses, that is corresponding to small motions with a quasi-constant depth, both algorithms give correct results, as can be seen in the following examples: baboon[3], vcbox[4], yosemite[5].

7.2 Images with two depth plans

For images corresponding to scenes with two depth plans, and for this reason which do not verify the hypothses, both algorithms see only the bigger one, because they minimize in mean, and almost correctly estimate the motion of the camera.

[3]http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?01.x=55&01.y=10
[4]http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?02.x=94&02.y=62
[5]http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?11.x=21&11.y=53

In the following examples, we use both algorithms on a complete scene with two depths, and then only on the first depth plan, and only on the second depth plan: soda can[6], soda can (only foreground)[7], soda can (only background)[8].

In the three cases, the direction of the translation is nearly the same. Recall that (A, B, C) are the coordinates of $\frac{t}{Z_0}$. Numbering the images with only the foreground 1 and the images with only the background 2, We should have $(A_1, B_1, C_1) \times Z_1 = (A_2, B_2, C_2) \times Z_2$. We don't know Z_1 and Z_2, but as expected, we have $B_1 > B_2$.

7.3 Images with a small object in movement

The algorithms are not expected to work for sequences with non static scenes. But both algorithms work for scenes with a small object in movement, because the minimizations are done in mean, and for this reason do not see the small objects.

That can be seen in the following example: street[9]. The camera is translated, filming a road with a moving car. The translation in correctly estimated: after motion compensation using the results of the algorithms, everything is perfectly as the right place except the car.

Another example of this phenomenon can be seen in yosemite with clouds[10].

7.4 Images with no movement, but lighting change

An interesting limit case if that of sequences captured with a fixed camera, presenting illuminations variations in time. Both algorithms detect that there is no motion, as showed by the example fortuny[11]. The illumination constant in the Jonchéry's algorithm plays its role.

7.5 Textures

Neither Jonchéry's algorithm nor the differential one work for textures images (such as a wall with roughcast), even if the hypotheses (of depth and displacment) are verified, as can be seen in the following example: wall[12].

7.6 Images with many depths

If the scene is made of several large parts of different depths, so that it is not possible to define a predominant depth, then both algorithms return a false estimation of the camera motion. See the sequence urban[13]: the scene is made of cubes at several distance from the camera.

Without a prior estimation of the image depth zones, and a depth-adaptive estimation of the camera motion, the proposed algorithms are inefficient.

7.7 Comparison between the two algorithms

If hypotheses are satisfied (small motion, quasi-constant depth), both algorithm give satisfying results and are robust: as an example, in the sequence car the car displacement does not affect the estimation of the camera motion since it represents a local perturbation in comparison of the global scene motion.

[6] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?03.x=78&03.y=92`
[7] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?04.x=58&04.y=87`
[8] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?05.x=60&05.y=86`
[9] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?12.x=108&12.y=50`
[10] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?12.x=68&12.y=77`
[11] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?07.x=96&07.y=98`
[12] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?08.x=30&08.y=91`
[13] `http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?09.x=75&09.y=92`

In addition, in our experiments, for examples satisfying hypothesis, both algorithms are of similar accuracy.

Our analysis of the time computation efficiency of the two algorithms does not reveal the superiority of one or another, whereas Jonchéry's method is presented in [2] as a fast méthod: see table 1.

Pair of images	Jonchéry's algorithm	Differential algorithm
baboon	1,01	0,91
vcbox	4,25	1,32
pepsi-complete	0,60	0,65
street	1,30	0,57
fortuny	2,74	4,57
urban	5,56	4,26
homography	2,24	0,99
pluto	9,70	29,04

Table 1: Running times in seconds

However, Jonchéry's algorithm can adress larger motions, as can be seen in the following example: 0.08726 radians[14]. See table 2 [15].

	θ	α	β	A	B	C
Expected result	any	0	$8.726 \cdot 10^{-2}$	0	0	0
Jonchéry's algorithm	2.068	$1.550 \cdot 10^{-4}$	$8.651 \cdot 10^{-2}$	$3.607 \cdot 10^{-4}$	$-6.702 \cdot 10^{-5}$	$7.574 \cdot 10^{-3}$
Differential	4.122	$8.021 \cdot 10^{-2}$	$7.964 \cdot 10^{-2}$	$-7.395 \cdot 10^{-2}$	$4.501 \cdot 10^{-2}$	$3.079 \cdot 10^{-2}$

Table 2: Result of the algorithms for a rotation of 5 degrees

Last, mention that the filtering step by M-estimator proposed in [2] was not implemented. Its benefices in terms of robustness remain unknown.

8 Source Code

The following git repository contains the source codes of the two main programs, another one to compute the motion on a complete camera sequence, and a tool to simulate camera motions: http://github.com/david64/estimation-mvt-cam[16].

[14]http://dev.ipol.im/~leo/ipol_demo/gl_camera_motion/input_select?19.x=63&19.y=11

[15]As α is equal to zero, the value of θ does not matter.

[16]http://github.com/david64/estimation-mvt-cam