Published in Image Processing On Line on 2011–09–13.
Submitted on 2011–00–00, accepted on 2011–00–00.
ISSN 2105–1232 © 2011 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.2011.m_qer

Quasi-Euclidean Epipolar Rectification

Pascal Monasse

Université Paris-Est, LIGM (UMR CNRS 8049),
Center for Visual Computing, ENPC,
F-77455 Marne-la-Vallée (monasse@imagine.enpc.fr)

Abstract

The standard setup in reconstructing the three-dimensional geometry of a scene from a pair of stereo images is to have them rectified, in which case the apparent motion of points is horizontal. With pinhole cameras, it is always possible to find two homographies that rectify the images. The method of Fusiello and Irsara assumes that both cameras are the same with principal point at the center, but keeps the focal length as an unknown. The virtual rotations of the two cameras are then evaluated to minimize the vertical motion of points.

Source Code

C++ source code of the full pipeline (containing SIFT point detection and matching, ORSA, rectification) is provided on the IPOL web page of this article[1].

Keywords: homography; image matching; multiple image analysis

1 Overview

The binocular stereo pipeline performs the following task: from a pair of images of a scene captured by two cameras (or the same one) at different positions, compute the distance map to one camera. Under particular conditions (called the *rectified case*) corresponding points are at same ordinate in both images and the distance is the inverse of an affine transform (whose coefficients depend on camera parameters) of the abscissa difference, called the *disparity*. In general, the output is simply the disparity map.

If the cameras satisfy the pinhole model assumption, an adequately chosen combination of rotations and adjustment of cameras' internal parameters yield the rectified case. This amounts to applying homographies to the images. Finding and applying these homographies is called *epipolar rectification*. The input is a discrete set of corresponding points in images. Since there are more degrees of freedom than constraints, several methods exist, each trying to minimize the change applied to images according to its own measure. One method to achieve this and more discussion can be found in the book of Hartley and Zisserman [3].

[1] http://dx.doi.org/10.5201/ipol.2011.m_qer

The method of Fusiello and Irsara [1], expanded in [2], assumes both cameras are the same (thus they must have the same size) but only partial knowledge of camera's internal parameters (uncalibrated case): square pixels, unknown focal length and principal point at image center. It then simulates appropriate pure rotations of each view, which can be done when the correct focal length is estimated.

2 The Fusiello-Irsara Method

2.1 Background

The epipolar constraint equation is written

$$X_l^T F X_r = 0,$$

with X_l and X_r 3-vectors of homogeneous coordinates of corresponding points and F the 3×3 fundamental matrix. In the rectified case, F has the special form (up to a scale factor)

$$F = [e_1]_\times = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}.$$

In that case, epipoles are at infinity in horizontal direction since $Fe_1 = F^T e_1 = 0$, and corresponding points have same ordinate since

$$\begin{pmatrix} x_l & y_l & 1 \end{pmatrix} F \begin{pmatrix} x_r & y_r & 1 \end{pmatrix}^T = 0 \Leftrightarrow \begin{pmatrix} x_l & y_l & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & y_r \end{pmatrix}^T = 0 \Leftrightarrow y_l = y_r.$$

Any decomposition of a fundamental matrix F into

$$F = H_l^T [e_1]_\times H_r,$$

with H_l and H_r invertible 3×3 matrices, yields left and right homographies rectifying the images since then

$$X_l^T F X_r = 0 \Leftrightarrow (H_l X_l)^T [e_1]_\times (H_r X_r) = 0.$$

2.2 Rotations

The rotation of matrix R of a camera around its focus produces a homography of the image of matrix

$$H = KRK^{-1},$$

with K the upper-triangular matrix built from camera parameters, which is reasonably supposed written as:

$$K = \begin{pmatrix} f & 0 & x_C \\ 0 & f & y_C \\ 0 & 0 & 1 \end{pmatrix},$$

with (x_C, y_C) the principal point, assumed to be the image center, and f the unknown focal length.

The Fusiello-Irsara method looks for rotation matrices R_l and R_r and focal length f such that

$$E(x_l, y_l, x_r, y_r) = X_l^T K^{-T} R_l^T K^T [e_1]_\times K R_r K^{-1} X_r = 0,$$

which can be simplified by removing the innermost two instances of K since

$$K^T[e_1]_\times K = f^2[e_1]_\times.$$

This equality also shows that in the rectified case, we can always change the internal parameters of both cameras to K_n, amounting to homography

$$K_n K^{-1},$$

and remain rectified.

Also if R_x is any rotation around the x-axis, we have

$$R_x^T[e_1]_\times R_x = [e_1]_\times,$$

so we can ignore the x-axis rotation in R_l for example. Finally, the unknown matrices can be parameterized by

$$R_l = R_z(\theta_{lz})R_y(\theta_{ly}), \quad R_r = R_z(\theta_{rz})R_y(\theta_{ry})R_x(\theta_{rx}), \quad K = K(f = 3^g(w+h)),$$

so that the expected value of g is in the interval $[-1, 1]$, the same order of magnitude as angles.

The Sampson error associated to algebraic error E is

$$E_s^2 = E^T(JJ^T)^{-1}E,$$

with J the 1×4 matrix of partial derivatives of E w.r.t. the 4 variables:

$$J = \begin{pmatrix}(FX_r)_1 & (FX_r)_2 & (F^TX_l)_1 & (F^TX_l)_2\end{pmatrix},$$

leading to

$$E_s(X_l, X_r)^2 = \frac{E(X_l, X_r)^2}{\|[e_3]_\times F^T X_l\|^2 + \|[e_3]_\times F X_r\|^2}.$$

The method looks for the set of parameters

$$\begin{pmatrix}\theta_{ly} & \theta_{lz} & \theta_{rx} & \theta_{ry} & \theta_{rz} & g\end{pmatrix}$$

minimizing the sum of Sampson errors over matching pairs by Levenberg-Marquardt method.

2.3 Jacobian Computation

The iterative error minimization requires computation of the Jacobian matrix. Noting any variable p, we have

$$\frac{\partial E_s}{\partial p} = \frac{X_l^T F' X_r}{D} - N\frac{\overline{F^T X_l}^T \overline{F'^T X_l} + \overline{FX_r}^T \overline{F'X_r}}{D^3},$$

with

$$F = (R_l K^{-1})^T[e_1]_\times(R_r K^{-1}), \quad F' = \frac{\partial F}{\partial p}, \quad N = X_l^T F X_r, \quad D = \sqrt{\|\overline{F^T X_l}\|^2 + \|\overline{FX_r}\|^2}$$

and the overline operator defined as

$$\overline{\begin{pmatrix}a & b & c\end{pmatrix}^T} = \begin{pmatrix}a & b\end{pmatrix}^T.$$

The partial derivatives w.r.t. any of the 5 angles are easy to compute, as each one is involved in only one rotation matrix:

- $\dfrac{\partial F}{\partial \theta_{ly}} = (R_z(\theta_{lz})R'_y(\theta_{ly})K^{-1})^T[e_1]_\times R_r K^{-1}$
- $\dfrac{\partial F}{\partial \theta_{lz}} = (R'_z(\theta_{lz})R_y(\theta_{ly})K^{-1})^T[e_1]_\times R_r K^{-1}$
- $\dfrac{\partial F}{\partial \theta_{rx}} = (R_l K^{-1})^T[e_1]_\times R_z(\theta_{rz})R_y(\theta_{ry})R'_x(\theta_{rx})K^{-1}$
- $\dfrac{\partial F}{\partial \theta_{ry}} = (R_l K^{-1})^T[e_1]_\times R_z(\theta_{rz})R'_y(\theta_{ry})R_x(\theta_{rx})K^{-1}$
- $\dfrac{\partial F}{\partial \theta_{rz}} = (R_l K^{-1})^T[e_1]_\times R'_z(\theta_{rz})R_y(\theta_{ry})R_x(\theta_{rx})K^{-1}$,

with

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}, \quad R'_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \cos(\theta+\frac{\pi}{2}) & -\sin(\theta+\frac{\pi}{2}) \\ 0 & \sin(\theta+\frac{\pi}{2}) & \cos(\theta+\frac{\pi}{2}) \end{pmatrix}$$

being its derivative, and similarly for R_y and R_z.

Now, the derivative w.r.t. g is

$$\frac{\partial F}{\partial g} = (R_l K^{-'}(g))^T[e_1]_\times R_r K^{-1}(g) + (R_l K^{-1}(g))^T[e_1]_\times R_r K^{-'}(g)$$

with

$$K^{-1}(g) = \begin{pmatrix} 1/f & 0 & -w/(2f) \\ 0 & 1/f & -h/(2f) \\ 0 & 0 & 1 \end{pmatrix} \quad K^{-'}(g) = -\log 3 \begin{pmatrix} 1/f & 0 & -w/(2f) \\ 0 & 1/f & -h/(2f) \\ 0 & 0 & 0 \end{pmatrix}.$$

2.4 Fixing Free Parameters

We take advantage of the invariance properties of the rectified configuration, w.r.t. to rotation around the baseline (x-axis) and change of internal parameters, to fix the center point of each image through application of rectifying homography.

First, we apply an x-rotation so as to keep the ordinate of the central pixel fixed, meaning replacing the homographies by

- $H'_l = KR_x(\alpha)R_l K^{-1}$
- $H'_r = KR_x(\alpha)R_r K^{-1}$

with the angle

$$\alpha = \operatorname{atan}\frac{(H_l C)_y - y_c}{f} = \operatorname{atan}\frac{(H_r C)_y - y_c}{f}.$$

Finally, we adjust independently the abscissa of left and right camera to keep the center point at center. Noting

$$K_n(x) = \begin{pmatrix} f & 0 & x \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{pmatrix}$$

and knowing the resulting abscissa x' of center point through H'_l, based on matrix $K = K_n(w/2)$, we need to translate by $w/2 - x'$, yielding the final homographies

- $H''_l = K_n(w - (H'_l C)_x)R_x(\alpha)R_l K^{-1}$,
- $H''_r = K_n(w - (H'_r C)_x)R_x(\alpha)R_r K^{-1}$.

3 Online Demo

The online demo of the algorithm has additional steps to make it more widely testable, as the demonstrated algorithm itself takes as input only corresponding points between images and the size of images and outputs homographies:

1. Extract key points with SIFT algorithm in each image;
2. Match key points based on descriptors;
3. Remove outlier correspondences by *a contrario* RANSAC algorithm (ORSA) looking for a consensus on fundamental matrix;
4. Apply Fusiello-Irsara epipolar rectification algorithm described above;
5. Apply homographies to both images.

Remember that input images must have the same size. A common failure case is when one epipole is in the image. Another problem is when geometrical distortion is noticeable in images (this happens especially with wide-angle lenses). In that case, it is preferable to correct this distortion (use for example this IPOL algorithm[2]).

The image rectangle can be mapped to a large quadrilateral if the epipole is outside but close to the image boundary. To avoid potentially very long transfer time over the network, we do not show the rectified image containing the full quadrilateral. Instead, we show only the portion of same size as the original image and with same center point, since this point has been fixed by adjustment of parameters of homography.

Here is an example of algorithm output:

```
01 sift:: 1st image: 473 keypoints
02 sift:: 2nd image: 496 keypoints
03 sift:: matches: 146
04 seed: 1286188619
05 Remove 19/146 duplicate matches
06 Optimized stochastic mode (ORSA).
07   nfa=-130.795 size=111 (niter=1)
08   nfa=-147.135 size=102 (niter=6)
09   nfa=-156.207 size=106 (niter=10)
10   nfa=-190.564 size=103 (niter=16)
11   nfa=-207.249 size=107 (niter=32)
12   nfa=-207.436 size=105 (niter=424)
13   nfa=-211.225 size=106 (niter=475)
14 best matching found:  106 points  log(nfa)=-211.225  (500 iterations)
15 F= [ -3.42803e-09 4.35331e-07 -0.000178794; -5.00755e-07 -4.76413e-08 ... ]
16 Geometric error threshold: 0.869643
17 LM iterations: 8 f=760.855
18 Final rectification error: 0.206727 pix
19 Disparity: -90 -26
```

[2] http://www.ipol.im/pub/algo/ags_algebraic_lens_distortion_estimation/

Lines 01 and 02 show the number of SIFT points extracted in each image. Line 03 shows the number of SIFT correspondences. Lines 04 to 16 are output from the ORSA algorithm. Line 04 shows the seed used for random number generation and is useful only for debugging purpose. Line 05 shows the number of remaining correspondences when duplicates are removed: some SIFT points may be found to have several main directions in their descriptor, which creates duplicates and can fool the ORSA match independence assumption. Lines 07 to 13 show each time a more meaningful group of correspondences is found. Line 14 sums up the best consensus. Line 15 shows the fundamental matrix in Matlab format. Line 16 displays the geometric threshold that was selected by ORSA to discriminate inliers and outliers. Here, every SIFT point must have its corresponding SIFT point at less than 0.87 pixel from the epipolar line. Lines 17 and 19 show the result of the rectification itself. The number of Levenberg-Marquardt iterations is here 8 and the found focal length is displayed. Line 18 shows the resulting mean Sampson error. Finally, line 19 gives the minimum and maximum disparity of SIFT points in the rectified images.

4 Algorithm

The algorithm is a Levenberg-Marquardt iterative minimization of the Sampson error, starting with all 6 unknowns at 0. It stops when one of the following conditions is satisfied:

1. the RMSE is below 0.1 pixel.
2. the relative error change from one iteration to the next is below 10^{-3}.
3. 300 iterations have been performed.

The first case is considered a success and the last one a failure. The second case may be either: the global minimum is reached but the data inaccuracy prevents the error from dropping below 0.1 (success) or a local minimum is reached (failure).

5 Implementation

5.1 Eliminating Null Equations

During Levenberg-Marquardt minimization (see [3]), one has to solve a system of 6 equations

$$HX = J^T E, \text{ with } H = J^T J + \lambda \text{diag}(J^t J),$$

with J the Jacobian matrix of the error term, of size $N \times 6$ with N the number of correspondences. The solution X is the update to apply to the current set of parameters. One of these 6 equations may happen to be the trivial one

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} X = 0.$$

This is the case when one column of J is null. This happens in particular at the first iteration for the g-derivative, when all 5 angles are 0, since

$$F' = K^{-'T}[e_1]_\times K^{-1} + K^{-T}[e_1]_\times K^{-'} = k[e_1]_\times \Rightarrow \frac{\partial E_s}{\partial g}(0,0,0,0,0,g) = \frac{kN}{D} - N\frac{kD^2}{D^3} = 0.$$

Notice that the diagonal elements of $J^T J$ are the square norms of column vectors of J. So, to avoid division by zero or by a negligible quantity, we find the maximum diagonal entry M of H and put in `m_nullCols` the indices of diagonal elements lower than kM, with $k = 10^{-9}$. We remove these lines and columns from the linear system to solve in the method

`void MinLM::compress(matrix<flnum>& JtJ, vector<flnum>& B).`

The solution of the smaller system is then completed with 0 at removed indices in the method

`void MinLM::uncompress(vector<flnum>& B),`

since these elements correspond to variables with null derivative, which must not be changed.

5.2 Homography Application with Anti-Aliasing Filter

Applying directly the computed homographies to images can result in aliasing artifacts. Although less severe than in the case of a zoom out for example, because we simulate rotations in our case, it is nonetheless important to attenuate the aliasing. This happens when there is a compression in one direction: at each point of the original image, the homography can be approximated by its 2×2 Jacobian matrix. When one of the singular values is below 1, compression occurs and anti-aliasing filter is recommended. For this, we have to know the maximum unidirectional compression factor over the image. Under normal conditions (image of line at infinity of original image does not meet the frame of the result image), the maximum compression (lowest singular value of Jacobian) is reached at one of the four corners of the original image. The procedure is the following:

1. At each corner of the original image frame, compute 2×2 Jacobian matrix of homography H and its minimum singular value. Note z the minimum of these over all 4 corners.

2. If $z < 1$, compose to the left H with a zoom in of factor $s = 1/z$; apply the resulting homography to the image, yielding an image of size $sw \times sh$; convolve with a Gaussian filter of standard deviation $0.8 \cdot \sqrt{s^2 - 1}$; downsample the smoothed image by a factor s.

3. If $z \geq 1$, regular transform is applied without any anti-aliasing filter.

The discrete Gaussian kernel is cut at 4 times its standard deviation. Interpolation is done with splines of order 5.

5.3 Differences with Authors' Implementation

Fusiello and Irsara's Matlab implementation of their algorithm, available here[3], does not correspond exactly to the text in their paper. We underline here the differences, both with the paper version and their implementation.

1. They write the epipolar constraint $X_r^T F X_l = 0$, whereas we write it $X_l^T F X_r = 0$. In other words, the fundamental matrices are transpose of each other. We find our convention more logical as subscript l (left) corresponds to the term on the left of the fundamental matrix.

2. The Matlab code contains fully developed formulas for the Jacobian matrix, probably gotten from some symbolic computation program. We use matrix calculus to compute it. Moreover, the Matlab code has the *wrong* Jacobian: it corresponds to the Jacobian matrix of the square errors. This is harmless in their implementation as they specify to use finite differences for the computation, not the closed formulas.

3. They decompose a rotation matrix with Euler angles in this order: $R_x R_z R_y$, we prefer the more natural $R_z R_y R_x$.

[3] http://profs.sci.univr.it/~fusiello/demo/rect/

4. They cancel the x-rotation in the right term of F, we do it in the left term.

5. The paper claims using Levenberg-Marquardt (LM) minimization with `lsqnonlin` Matlab function. The way this function is called in their code uses the default "trust region reflective" algorithm. On some examples, we found this minimization to be less susceptible to get caught in a local minimimum of the energy. However, we implemented it with the better known LM algorithm, as claimed in the paper. Modifying their Matlab code to use LM in `lsqnonlin`, we get results very similar to ours.

6. The Matlab code tries the minimization up to 20 times with random initialization of the focal length while the results are not satisfactory. We try only once with $g = 0$.

7. The Matlab code has the option of keeping the center image point fixed. This is done by translating the principal point. We prefer using only an x-translation (to fix abscissa) of the principal point and an x-rotation (to fix ordinate).

6 Examples

We show some examples of successful rectification by the above algorithm in figures 1 to 5. In most cases, 10 or fewer iterations of the Levenberg-Marquardt algorithm are necessary. To check visually the result, change the height of your window so as to cut the images at some level; check that discernible features (blobs, corners) move along that line.

In each case, the first two images are the original ones and the last two the rectified ones.

Notice that different runs of the full pipeline with same data do not necessarily yield the exact same results as there is a random component in ORSA.

Image Credits

All images copyright Pascal Monasse, license CC BY-SA.

Acknowledgments

This work was supported by Centre National d'Etudes Spatiales (CNES) and by Agence Nationale de la Recherche ANR-09-CORD-003 (Callisto project). Different parts of the support code used in the demo were written by Toni Buades, Nicolas Limare, Lionel Moisan and Zhongwei Tang.

References

[1] A. Fusiello and L. Irsara. Quasi-euclidean uncalibrated epipolar rectification. In *19th International Conference on Pattern Recognition*, pages 1–4. Tampa, Florida, US, 2008. http://dx.doi.org/10.1109/ICPR.2008.4761561.

[2] A. Fusiello and L. Irsara. Distinctive image features from scale-invariant keypoints. *Machine Vision and Applications*, 22(4):663–670, 2010. http://dx.doi.org/10.1007/s00138-010-0270-3.

[3] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. ISBN 978-0-521-54051-3.

Figure 1: Beijing lion: 107 correspondences are found and the final RMSE is 0.25 pixels after 6 iterations.

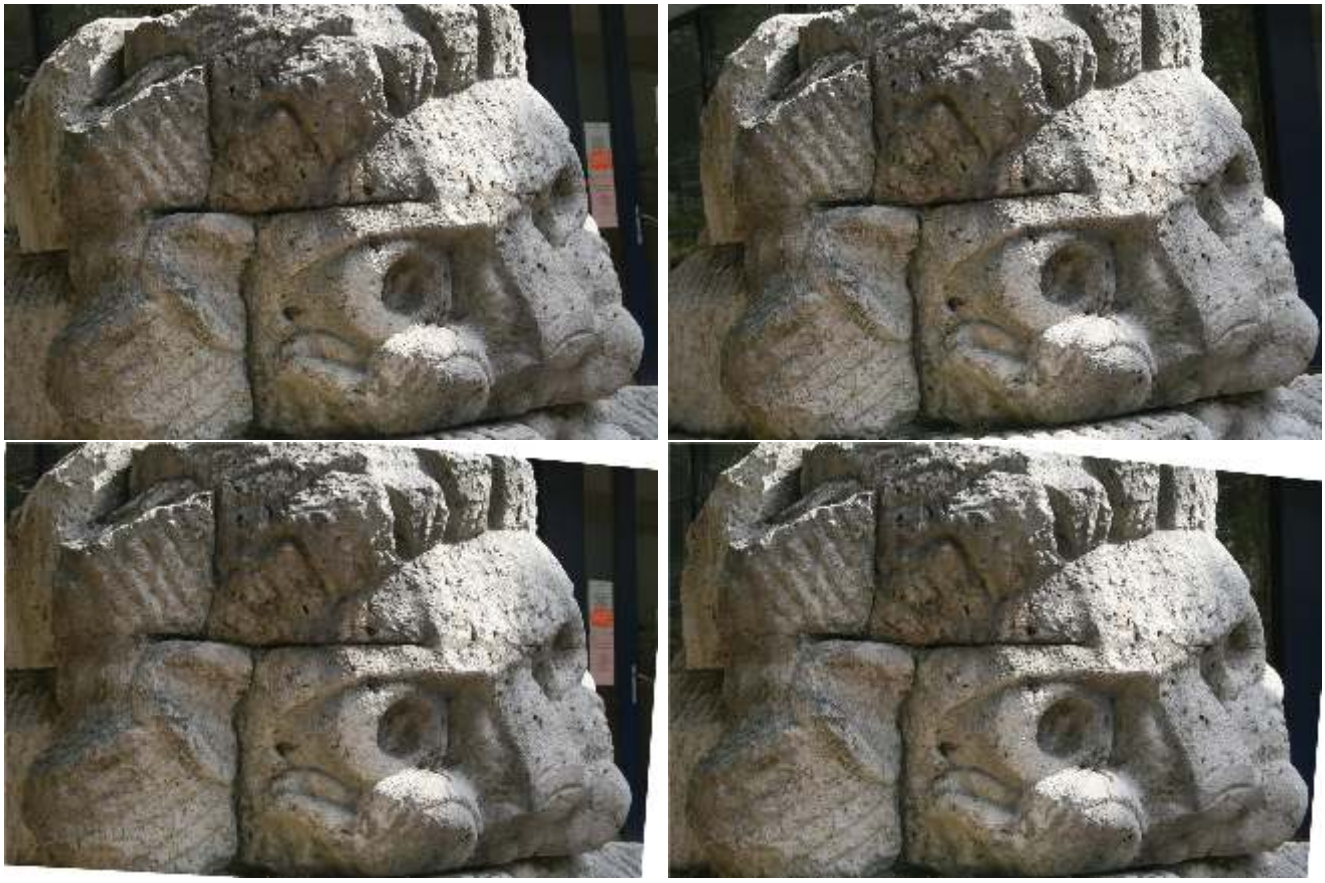

Figure 2: Cachan statue: 964 correspondences are found and the final RMSE is 0.13 pixels after 5 iterations.

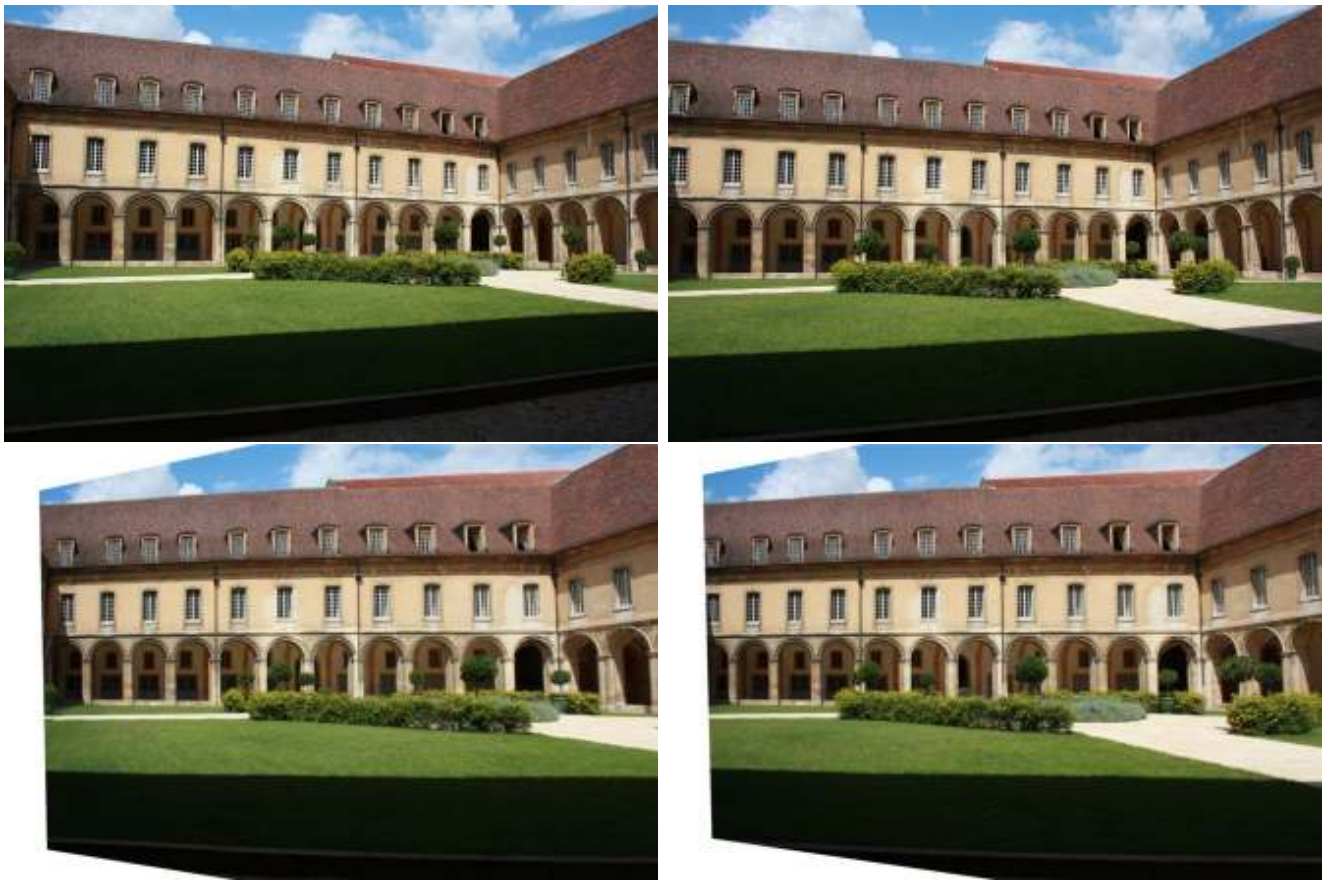

Figure 3: Cluny church: 110 correspondences are found and the final RMSE is 0.27 pixels after 35 iterations.

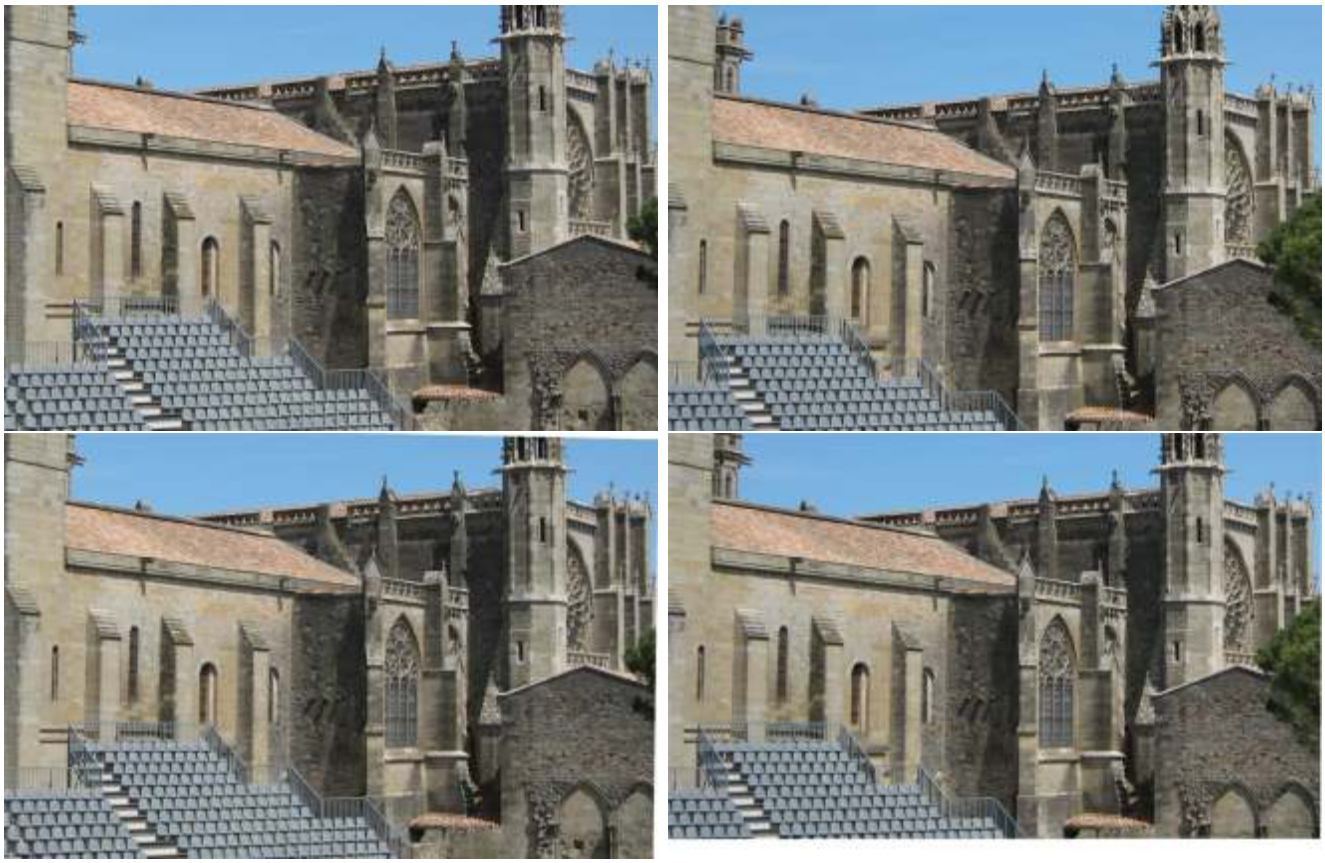

Figure 4: Carcassonne: 503 correspondences are found and the final RMSE is 0.19 pixels after 55 iterations.

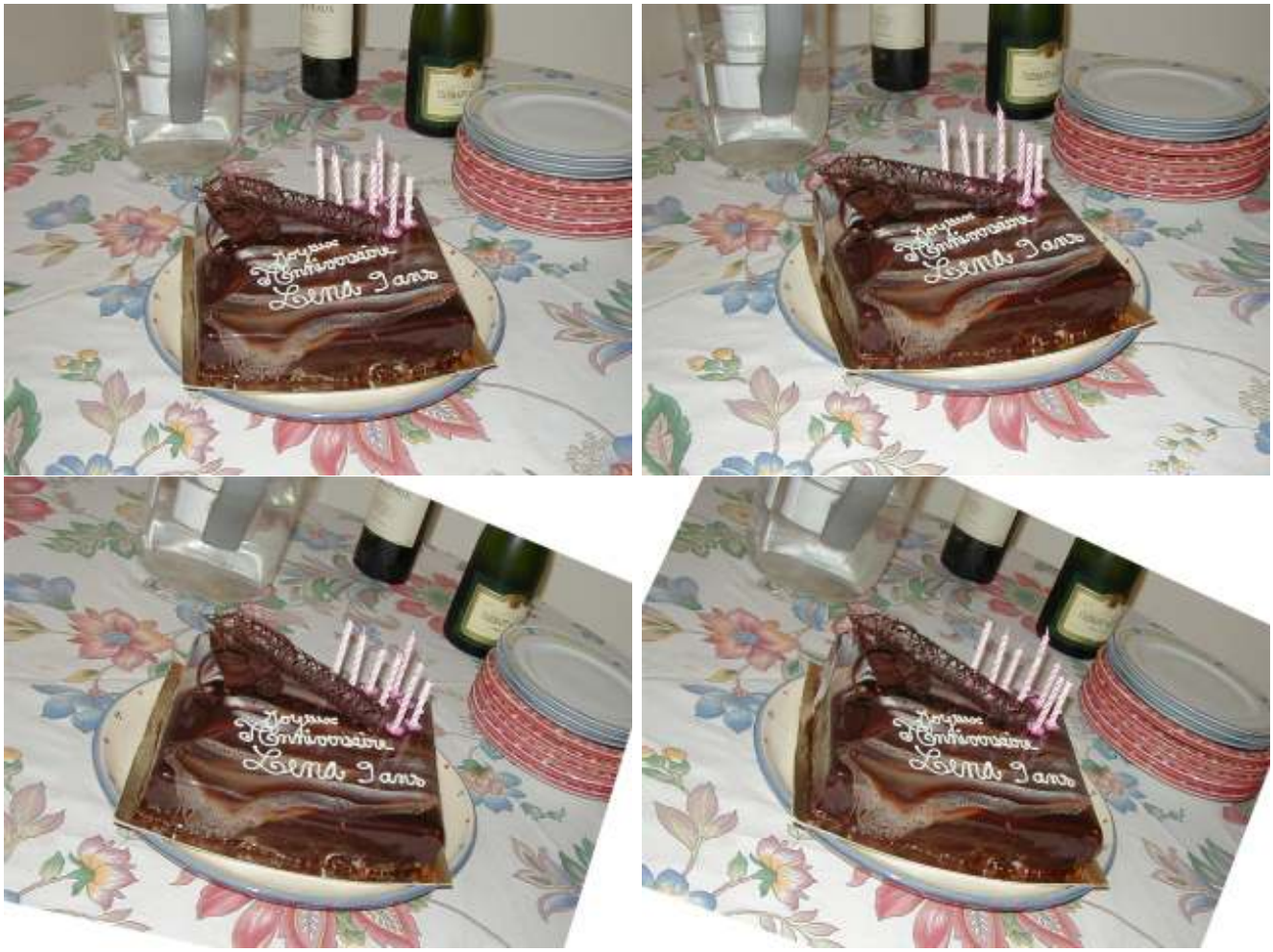

Figure 5: Birthday cake: 146 correspondences are found and the final RMSE is 0.69 pixels after 11 iterations. The final error is higher than in previous examples, due to some geometric distortion of the camera.