Using geometric corners to build a 2D mosaic from a set of images

I. Zoghlami and O. Faugeras and R. Deriche
INRIA, 2004 route des Lucioles, BP 93
F-06902 Sophia-Antipolis Cedex
Email : Imad.Zoghlami@sophia.inria.fr

Keywords

2D mosaic, corner models, homography estimation, line matching.

Abstract

The main problem for building a mosaic is the computation of the warping functions (homographies). In fact two cases are to be distinguished. The first is when the homography is mainly a translation (i.e. the rotation around the optical axis and the zooming factor are small). The second is the general case (when the rotation around the optical axis and zooming are arbitrary). Some efficient methods have been developed to solve the first case. But the second case is more difficult, in particular, when the rotation around the optical axis is very large (90 degrees or more). Often in this case human interaction is needed to provide a first approximation of the transformation that will bring us back to the first case. In this article we present a method to solve this problem without human interaction for any rotation around the optical axis and fairly large zooming factors.

1 Introduction

In the last few years the interest in mosaicing has grown in the vision community because of its many applications (i.e. image compression, detection, video conferencing...). A mosaic is a collection of images and the transformations between them. In the case of a collection of images of a planar scene taken from different points of view or a collection of images of a 3D scene taken from the same point of view (e.g. the only difference between images is a rotation around the optical center of the camera), the transformation between the images is a linear transformation of the projective space $\mathcal{P}^2$, called a collineation or a homography [Fau93, Har94]. We call these mosaics 2D mosaics because when we choose a reference image from the collection, all other images can be warped in the 2D coordinate system attached to this image by the corresponding homographies. Other possibilities such as using a cylindrical projection also exist[MB95].

In this article we deal only with the 2D mosaics, in particular with the homography estimation. We distinguish two cases. The first is when the homography is mainly a translation (i.e. the rotation around the optical axis and zooming are small). The second is the general case (when the rotation around the optical axis and zooming are large). Some efficient methods have been developed to solve the first case. For example, if the overlap of the images is very large, (i.e. the motion is very small) it has been shown that a non linear criterion minimization using the Levenberg-Marquardt method yields very good results [Sze94], but it is very sensitive to the local minima and computationally expensive. In another case when the overlap is smaller we can use a *hierarchical matching* to avoid local minima[Qua84, WTK87, BAHH92]. For larger camera motion the *phase correlation* method has been used [KH75, Bro92]. However for large rotations around the optical axis, very few methods are efficient. Among the best methods we find the work of Dani and Chaudhuri: Their method works for up to 15 degrees rotations using angles between edges [DC95]. The *mutual Information* method for up to 30 degrees of viola works [Vio95], but it requires a large overlap between images (i.e. larger than 50%).

This article presents a corner-based method to compute the homography between two images with small overlap (≈50%) and arbitrary rotation around the optical axis. The computation takes a few seconds.

Section 2 presents the homography estimation using four points and the algorithm derived when no further assumptions are made. In section 3 we present the corner model and in section 4 an efficient algorithm that uses this model and a dual space representation of lines to estimate homographies. The method is illustrated with some results.

2 Computing a homography with four points

2.1 The equations

We use homogeneous coordinates to represent a point, thus a point m in an image will be represented by the vector $(x\ y\ z)^t$, with $(x/z\ y/z)^t$ its cartesian coordinates. As mentioned in the introduction, the images of a plane seen from two points of view are

related by a homography H. Thus a point $m_1 = (x_1\ y_1\ z_1)^t$ of the first image has a corresponding point in the second image $m_2 = (x_2\ y_2\ z_2)^t$ defined by:

$$m_2 = Hm_1 \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad (1)$$

Homographies and points are defined up to a nonzero scalar, thus we have 8 degrees of freedom for H (often, if h_{22} is different from zero, we take $h_{22} = 1$, and $(x/z\ y/z\ 1)^t$ for the points). Every correspondence (m_1, m_2) gives us two equations (2), thus to compute H we need four correspondences.

$$\begin{cases} x_2/z_2 = \frac{h_{00}x_1+h_{01}y_1+h_{02}}{h_{20}x_1+h_{21}y_1+1} \\ y_2/z_2 = \frac{h_{10}x_1+h_{11}y_1+h_{12}}{h_{20}x_1+h_{21}y_1+1} \end{cases} \quad (2)$$

2.2 The algorithm

Now, if no assumptions are made about the camera motion, we have the following algorithm called ALGO1:

- step 1: Extract points of interest from the two images.
- step 2: Compute all possible homographies defined by pairs of fourtuples of points of interest.
- step 3: Keep the *best one*.

In detail:

step 1: To extract points we use the Harris' detector [HS88], keeping only the best one in a given neighborhood. This is to get a homogeneous distribution of points within the whole image.

step 2: Knowing nothing about the motion, every point of the first image can match any point in the second image. Assuming that the camera is always on the same side of the object we know that any non self intersecting quadrilateral $(p_{11}\ p_{12}\ p_{13}\ p_{14})$ in image 1 matches a non self intersecting quadrilateral $(p_{21}\ p_{22}\ p_{23}\ p_{24})$ in image 2 with the same orientation(a non self intersecting quadrilateral has two possible orientations). If n_1 and n_2 are the numbers of extracted points in image 1 and in image 2, respectively, we have $4\binom{n_1}{4}\binom{n_2}{4}$ possible homographies H_k between these points.

step 3: We have to find, among the homographies H_k, the best estimation of the homography between the two images. The number of homographies is very large, so we use a simple method to validate them. To check an H_k we correlate the intensity values at all points of interest p_{1i} of the first image with those of their corresponding points $H_k p_{1i}$. The *best one* is thus the homography which maximizes the following correlation criterion (3).

$$H_{opt} = \max_{H_k}\left(\frac{1}{n}\sum_i ZNCC(p_{1i}, H_k p_{1i})\right) \quad (3)$$

Where ZNCC is the zero mean cross-correlation (between -1 and +1, for more information see [FHM+93]) using sub-pixel values for the points in the second image. We use only the points $H_k p_{1i}$ lying within the second image. n is the number of those points.

The necessary condition for the algorithm to find a solution is that among the extracted points in the first image there exist four points having their corresponding points in the second image. So we need a stable point extractor. In table 1 we present a few results to show the stability of the Harris extractor. For this test we used the two images of figure 1. The original images are 768*512.

Nb pts	Nb pts inter.	% match	Dist.(*pixels*)
12	6	66	0.9
54	31	75	1.1
133	77	76	1.1
467	276	80	1.2

Table 1: The first column shows the numbers of the extracted points in the first image, the second column the number of points matched in the common part of the two images, the third column shows the percentages of matched points in the common part of the two images, and the last column the distances between the extracted points and the corresponding points warped with the estimated homography.

2.3 Results

We use table 1 to produce an empirical formula 4 to estimate the number of points that we need to extract to have a sufficient number of matches to find a good estimation for the homography.

$$nb_{ext} \approx \frac{nb_{int}}{overlap * match} \quad (4)$$

Where nb_{ext} the number of points that we need to extract. nb_{int} is the number of point desired within the overlap image. *overlap* is the percentage of the overlap image. *match* is the percentage of match (colon 3 of table 1).

Figure 1 shows an example of a mosaic built from the two images. We need 4 matches within the overlap image, but usually we overestimate this number to insure to find 4 good matches (e.g. we usually take 6 points). Thus for the example of figure 1 with formula

4 we have nb_{int}=6, $overlap$≈70%, $match$=66% and thus nb_{ext}≈12. In image 1, 12 points are extracted, points 1 to 9 are the points lying in the common part of the two images, points 1 to 6 are matched, and points 1 to 4 are the points giving H_{opt}. Note that only the first 6 points have a corresponding point thus 1 corresponds to 1, 2 corresponds to 2,... but points 7 to 12 do not correspond to 7 to 11 in the second image.

Figure 1: Image 1 at the top, image 2 in the middle, in black the extracted points, in white the points matched, the mosaic built with H_{opt} at the bottom.

2.4 Complexity problem

The results obtained with this method are satisfactory. Notice that the overlap is about 70%, in this case only twelve extracted points are sufficient to have at least four corresponding points in the common part. But in spite of this small number of extracted points the computation time is fairly long (i.e. 12 minutes CPU), this is because there are a lot of homographies to check (Step 2 of the algorithm has approximatively 1 million homographies to check). For a smaller overlap, for example 50%, we have to extract twenty points in the two images, this gives us approximately 100 millions homographies to check and requires a few hours of computation. Due to this complexity, the method is not usable for small overlap.

3 Corner Model

The main idea is to use more information from the corner than just its coordinates. To obtain more information from the corner, we use the corner model developed by Blaszka and Deriche [DB93]. This method uses a corner model which utilizes a 2D blurring filter for the modelization and fits the model to the image data by non-linear minimization, for more information see [BD94].

3.1 Corner Model

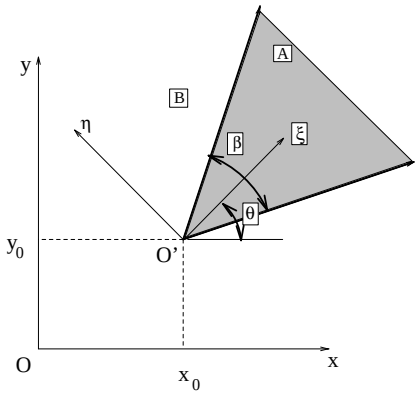

Figure 2: A Corner Model.

The corner model in figure 2 is completely described by 7 parameters noted $(x_0, y_0, \theta, \beta, \sigma, A, B)$. We initialize the corner model with a Harris point and a window size and obtain the 7 parameters of the best fitted model in this window and a measure of the quality of the fit. This measure is the mean least-squares difference between the grey-levels in the image and those in the model (between 0 and 255). We keep only the corners with a small measure. The points fitted by the model are called D.B. corners.

3.2 Results

Figure 3 shows the results obtained with a window size of 16*16 for point 5 in the image 1 of figure 1, and figure 4 gives the result obtained for the corresponding point (i.e. point 5) in the image 2 of figure 1.

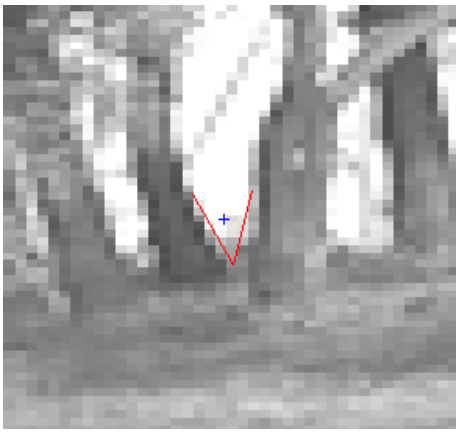

Figure 3: Zoom of image 1 at the point 5, the cross is the Harris point, the two segments are the corner, measure=2.7.

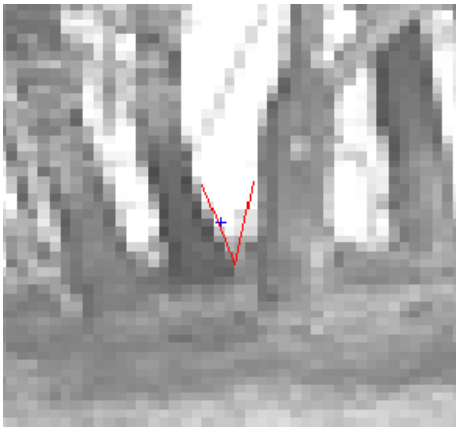

Figure 4: Zoom of image 2 at the point 5, the cross is the Harris point, the two segments are the corner, measure=2.1.

4 Computing an homography with two D.B. corners

In this section we present a method to compute homographies using only two D.B. corners instead of four simple corners. This is to decrease the complexity of the algorithm presented in section 2. We use the two lines defined by the corner model. These two lines d and d' can be computed from the corner parameters, and more precisely from $(x_0, y_0, \theta, \beta)$ as defined in section 3.

4.1 Homography computation using lines

Whilst in the projective plane, $\mathcal{P}^2$, we can use the duality principle between points and lines in this space (i.e. that points and lines are algebraically equivalent). Thus we have an homography between the corresponding lines in two images. We can compute this homography as in section 2. We need four lines to compute a homography. Using the notation of section 3, the two lines d and d' of a D.B. corner are defined by the following formula:

$$\begin{cases} d = & (-\sin(\alpha_1) & \cos(\alpha_1) & x_0\sin(\alpha_1) - y_0\cos(\alpha_1))^t \\ d' = & (-\sin(\alpha_2) & \cos(\alpha_2) & x_0\sin(\alpha_2) - y_0\cos(\alpha_2))^t \end{cases} \quad (5)$$

Where $\alpha_1 = \theta - \frac{\beta}{2}$ and $\alpha_2 = \theta + \frac{\beta}{2}$

We need only two D.B. corners instead of four simple corners to compute an homography. If n_1 and n_2 are the numbers of D.B. corners extracted in images 1 and 2 respectively, we have only $2\binom{n_1}{2}\binom{n_2}{2}$ homographies to compute which reduces the complexity of the algorithm to $O(n_1^2 n_2^2)$. Since we need only two D.B. corners to compute the homography, consequently we have to extract fewer points to insure two corresponding D.B. corners in the image overlap. Thus the use of D.B. corners decreases the overall complexity as well as the magnitudes of n_1 and n_2.

4.2 The relationship between the homography relating points and its dual relating lines

It is well known that if there is an homography H between points in the projective plane, then H^{-t} is the dual homography between lines [Fau93]. If we match lines we estimate H^{-t} instead of H.

Note that this is equivalent to computing H using the four points of intersection of the four lines of the two D.B. corners. For example in figure 5 we show the two corners (a_1 and b_1 in image 1 and a_2 and b_2 in image 2), and the two line intersections (c_1 and d_1 in image 1 and c_2 and d_2 in image 2).

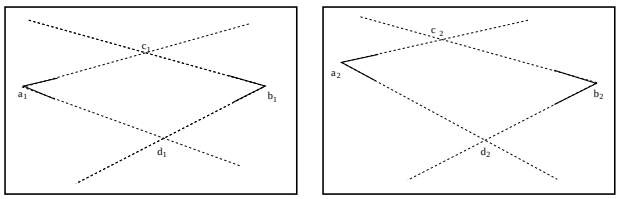

Figure 5: Left: a_1 and b_1 the two corners, c_1 and d_1 the two intersection points, Right: a_2 and b_2 the two corresponding corners, c_2 and d_2 the two intersection points.

4.3 The algorithm

We can thus use directly the four points method (section 2), replacing the four points by four lines. This gives us the algorithm below called ALGO2:

- step 1: extract points of interest from the two images.

- step 2: fit the corner model to the extracted points.
- step 3: Compute all homographies between all pairs of D.B. corners in image 1 and in image 2.
- step 4: keep the *best one* .
- step 5: refine the *best one* .

With respect to ALGO1 we added two steps. Obviously we added step 2, to fit the corner model to the extracted points to get the lines and compute the homographies. We also added step 5 (i.e. the refining step). In spite of a better localization of D.B. corners (see table 2), we must not forget that we compute the homography using lines and that an error in the lines induces an error in the homography. To have an idea of the effect of this error we can think of the dual case of computing the homography using points $(a_i, b_i, c_i, d_i\ i$=1,2) from the four lines as shown in figure 5. The two D.B. corners (a_i, b_i) are very well localized, but the intersections of lines (c_i, d_i) may not be very well localized. Thus we use the best homography to initialize a local refining method to compute an improved one (see section 4.4).

We can further improve the algorithm, by keeping the first refinable homography instead of computing all the homographies between the D.B. corners and keeping the best one. This gives us the following algorithm called ALGO3:

- step 1: extract points of interest from the two images.
- step 2: fit the corner model.
- step 3: for all computed homographies :
  - if criterion(3) is close to 1, try to refine H
  - Stop when a successful refinement is found.

This algorithm gives us the first pair of good corresponding D.B. corners, which avoids computing all the homographies. This is useful if we extract too many points from the two images, because we usually quickly get a good pair of corresponding D.B. corners.

Nb pts	Nb pts inter.	% match	Dist.(*pixels*)
10	6	85	0.4
20	12	85	0.5
39	18	66	0.5
93	52	69	0.7

Table 2: The first column shows the number of extracted points, the second column the number of points matched in the common part of the two images, the third column shows the percentages of matched points, and the last column the distances between the extracted points and the corresponding points.

In table 2 we notice that for a small number of D.B. corners extracted in the two images the percentage of matches is very high (i.e. 85%) in contrast with the percentage of matches for the simple corners (table 1). Thus the matching based on the D.B. corner model seems more robust than the one based on the Harris extractor.

4.4 The refining step

For the refining step we could use the methods cited in the introduction. Instead we use a very fast and simple method to refine the homography H_{opt}. Taking advantage of the good initialization, we consider that the scale and the rotation around the optical axis are roughly contained in H_{opt}, and assume that locally the real homography is H_{opt} up to a small translation. Each point p_{1i} in image 1 has its corresponding point $H_{opt}p_{1i}$, thus to refine H_{opt}, we translate this point in a small neighborhood and keep the point which correlates best with p_{1i}. This yields a new list of matched points, which we use to compute a new estimation of the homography by a non-linear minimization technique. We check this new homography using the correlation criterion (3) with more Harris points, if the criterion is close to 1 (e.g. $>$0.8) we stop the algorithm.

4.5 Speeding up the algorithm using a coarse to fine approach

For a small number of D.B. corners the fitting step takes up the main computation time. To speed up the algorithm we use a Multiscale Method: the fitting step takes less time at the coarse level, because we fit the corners with a smaller window size. So, we run first the algorithm at the coarsest level and refine iteratively the homography at all subsequent levels until we reach the highest resolution (i.e. the original image).

4.6 Results

All the results are for a SPARC Station 20.

Figure 6 shows the results obtained with ALGO3, we see that they are comparable with those obtained

with ALGO1 (figure 1). As the overlap is about 70%, 7 D.B. corners are sufficient to insure that we have at least two corresponding D.B. corners. The computation time is 10 seconds, this is because we have only approximatively 1600 homographies to check, and in this case the computation time is mainly the corner fitting operation.

In figure 7 the overlap is about 50% and we have add a rotation of 30 degrees. Having a 50% overlap we need about 10 D.B. corners, and there are approximatively 8000 homographies. In this case the computation time is 12 seconds, again most of it for the corner fitting operation.

In figure 8, we scanned two postcards of Pythagoreio (Samos Island-Greece), we still have 12 seconds CPU.

Figure 9 shows an interesting case, the overlap is about 50% and the second image is upside down. We still need 10 corners and we still have 12 seconds CPU.

In figure 10 we have a very large zoom (≈ 2), this gives us a 20% overlap. In this case we have two problems, the first is the small overlap, the second is the size of the features. For this reason we had to extract more points (about 30). The number of homographies is roughly 500000, the computation time is 8 minutes.

5 Conclusions

In this article we have presented a method to compute the homography between two images, with any rotation around the optical axis and a large zooming factor. The method relies on finding geometric features,(i.e. corners), which contain more information than simple points. This allows us to cut down the complexity of the matching between image features and hence to tackle more difficult cases than with points-based methods.

References

[BAHH92] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In Sandini, editor, *Proc. of the 2nd ECCV*, pages 237–252. Springer Verlag, 1992.

[BD94] Thierry Blaszka and Rachid Deriche. Recovering and characterizing image features using an efficient model based approach. Technical Report 2422, INRIA, November 1994.

[Bro92] L.G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, December 1992.

[DB93] R. Deriche and T. Blaszka. Recovering and characterizing image features using an efficient model based approach. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 530–535, New-York, June 1993. IEEE Computer Society, IEEE.

[DC95] P. Dani and S. Chaudhuri. Automated assembling of images: Image montage preparation. *Pattern Recog.*, 28(3):431–445, March 1995.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint.* MIT Press, 1993.

[FHM+93] Olivier Faugeras, Bernard Hotz, Hervé Mathieu, Thierry Viéville, Zhengyou Zhang, Pascal Fua, Eric Théron, Laurent Moll, Gérard Berry, Jean Vuillemin, Patrice Bertin, and Catherine Proy. Real time correlation based stereo: algorithm implementations and applications. Technical Report 2013, INRIA Sophia-Antipolis, France, 1993. Submitted to *The International Journal of Computer Vision.*

[Har94] Richard Hartley. Self-calibration from multiple views with a rotating camera. In J-O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision*, volume 800-801 of *Lecture Notes in Computer Science*, pages 471–478, Stockholm, Sweden, May 1994. Springer-Verlag.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conf.*, pages 189–192, 1988.

[KH75] C.D Kuglin and D.C Hines. The phase correlation image alignment method. In *Conference on Cybernetics and Society*, pages 163–165, New York, September 1975. IEEE.

[MB95] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH*, Los Angeles, CA, August 1995.

[Qua84] L.H. Quam. Hierarchical warp stereo. In *Image Understanding Workshop*, pages 149–155, New Orleans, Louisiana, december 1984. SAIC.

[Sze94] R. Szeliski. Image mosaicing for tele-reality applications. Technical Report CRL94/2, DEC-CRL, May 1994.

[Vio95] P. Viola. Alignment by maximization of mutual information. TR 1548, MIT, MIT Department Electrical Engineering and Computer Science, Cambridge,Mass, June 95. Phd thesis.

[WTK87] A. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *IJCV*, 1(2):133–144, 1987.

Figure 6: Top: the first image, Middle: the second image, Bottom: the mosaic.

Figure 7: Top: the first image, Middle: the second image (30 degrees rotation arround the optical axis and different viewing direction), Bottom: the mosaic.

Figure 8: Top: the two images, Bottom: the mosaic.

Figure 9: Top: the first image, Middle: the second image (180 degrees rotation arround the optical axis and different viewing direction), Bottom: the mosaic.

Figure 10: Top: the first image, middle: the second image (scale factor of 2 and different viewing direction), Bottom: the mosaic.