Fast and Realistic Image Synthesis for Telemanipulation Purposes

J. Blanc S. Livatino* R. Mohr

Movi, Gravir-Imag
Inria Rhône-Alpes
655, av. de l'Europe
38330 Montbonnot St-Martin
France
Jerome.Blanc@inrialpes.fr

Abstract

We describe here a method to build fast and realistic synthetic images, which does not require a tedious a priori *modeling of the scene to be displayed. We only use photos or videos of a real scene taken from some viewpoints, which in some way we "wrap" to get new (synthetic) images of the same scene. This way, we can virtually see and move inside the 3D scene.*

The process consists of two steps: we first need to registrate the so-called reference views, *using computer vision techniques. Then we just project the obtained model onto the plane of a virtual camera; this last step is fast and gives realistic results, allowing to use the method for instance to simulate the environment of a tele-operated device.*

1 Introduction

Synthetic images have proven useful for telemanipulation: synthesizing the environment of the operated device is needed for visualization, simulation, and better remote positioning. The necessity of virtual reality (VR) itself during the design stage has now become obvious. At the Cern in Geneva, Switzerland, a VR software helped them to set-up the building of the future Large Hadron Collider; by simulating, they saw they could avoid the building of an extra shaft, thus sparing millions of dollars! [Bal 96].

For these simulation uses, we must synthesize the environment of the robot to be tele-operated, and the robot or vehicle itself inside this scenery. The scenery should be a real 3D model, for a better feeling of the environment and of the possible collisions with the operated device.

*Grant of the Heros network of the EC Hcm program

Whereas such a modeling remains feasible by hand for a world of blocks, obtaining a complete 3D model of a complex environment (say, a whole city) becomes a tough and annoying task. More, it will never lead to realistic images without an extra cost in modeling (rocks, trees), and rendering.

What we present here is a mostly automated fast and realistic image synthesis technique, which relies on some real pictures of the 3D scene we want to simulate. We'll call these pictures *reference views*.

Algorithms Outline From some known reference views, our process needs 3 distinct steps:

1. a preprocessing step consists of matching the reference views together, to get a disparity map (see section 2);

2. from the disparity map, we build a 3D model; this model can be a set of 3D points, or a more sophisticated one (see section 3);

3. the last step just consists of projecting the contructed model onto virtual cameras, so as to generate the synthesized views (see section 4).

The views we'll use in this paper are aerial views of the city of Marseilles, France; they're shown on fig. 1.

2 Matching the Reference Views

2.1 Several Methods

Many methods can be used to recover a depth map between the reference views. The use of each method depends on:

a priori knowledge of the scene: for instance, knowing the epipolar geometry between the reference views allows to restrict the search area: corresponding points are only searched for on conjugate epipolar lines.

measure type: comparing two points can be done with raw correlation measures, or robust correlation measures, or measures of distances between quasi-invariants of the intensities of points.

the required density for the depth map: we can match only sparse points on the images (interest points), or contour points (but they're often on an occluding edge), or even all points.

For each case, we need a different algorithm. One we used in the presented paper matches sparse points, using the epipolar constraints, and the measure is a plain correlation measure, for instance SAD [Asc 92]. We use cross-correlations, that is, for one point in image 1, we find the best match in image 2. For this match, we find the best matching point in image 1, and if it is the first point again, then we keep the match, else we reject it.

Another algorithm is a brute-force method consisting of matching all the possible points of the reference views, using a dynamic programming algorithm (or again cross-correlations).

2.2 Recovering the Epipolar Geometry

For the algorithms we described, we need to compute the epipolar geometry from the two reference views. The method we implemented is the following:

1. Extract interest points in the 2 images (such as corners). These points are easily trackable. We use an ameliorated Harris corner detector [Sch 96].

2. Match these points using cross-invariants [Sch 96].

3. Refine the matches by moving the points with a subpixel accuracy and evaluate their resemblance with correlations measures.

4. Estimate the epipolar geometry from these subpixel matches. We mix [Har 95] and [Der 94] methods to get a robust and precise estimation.

The obtained disparity map is shown on figure 2, along with a denser version.

3 Computing a Model of the Scene

3.1 From a Projective to a Euclidean Reconstruction

From a disparity map, we need to obtain a 3D model. First of all, let's notice that if we only know the images - hence the epipolar geometry -, all we can get is a projective reconstruction of the scene [Fau 92]. That is not suitable for our use, where we would like to manipulate a real 3D euclidean model, using standard model viewers or editors. To obtain such a model, many techniques are available, including: setting euclidean constraints on the scene (lengths and angles measures), as in [Bou 93], or performing autocalibration (we need at least 3 images). But the best for us, as we don't need a full-automatic reconstruction, is simply to enter the internal parameters of the camera which took the reference views. These parameters can for instance be given by the manufacturer.

3.2 Building the 3D Points

From the epipolar geometry and the internal parameters, we compute the two projection matrices modeling the two cameras, using the algorithm described in [Luo 92]. Then for each match in the reference views, we compute the corresponding 3D point using a plain triangulation. The matches have to be precise and must exactly obey the epipolar constraint. If, as in our case, the matching process did not ensure these two points, we have to refine the obtained matches (we just move them by a subpixel amount, testing for a better correlation), and to make them fit the epipolar constraint. For the latter, we use the method of [Har 94].

3.3 Towards a Better Model

At this stage, we have a sparse cloud of 3D points. As we will see in section 4, this is a poor model, because we lost all information about connexity or surface. Moreover, we need to handle as many points as pixels could be matched in the reference views, i.e. tens or hundreds of thousands of points.

To solve this problem, we are currently experimenting another approach: from some sparse matches (dozens to hundreds), we build triangles in the reference views. By "backprojecting" them in the 3D sparse model, we obtain 3D textured triangles, that we can display with any viewer.

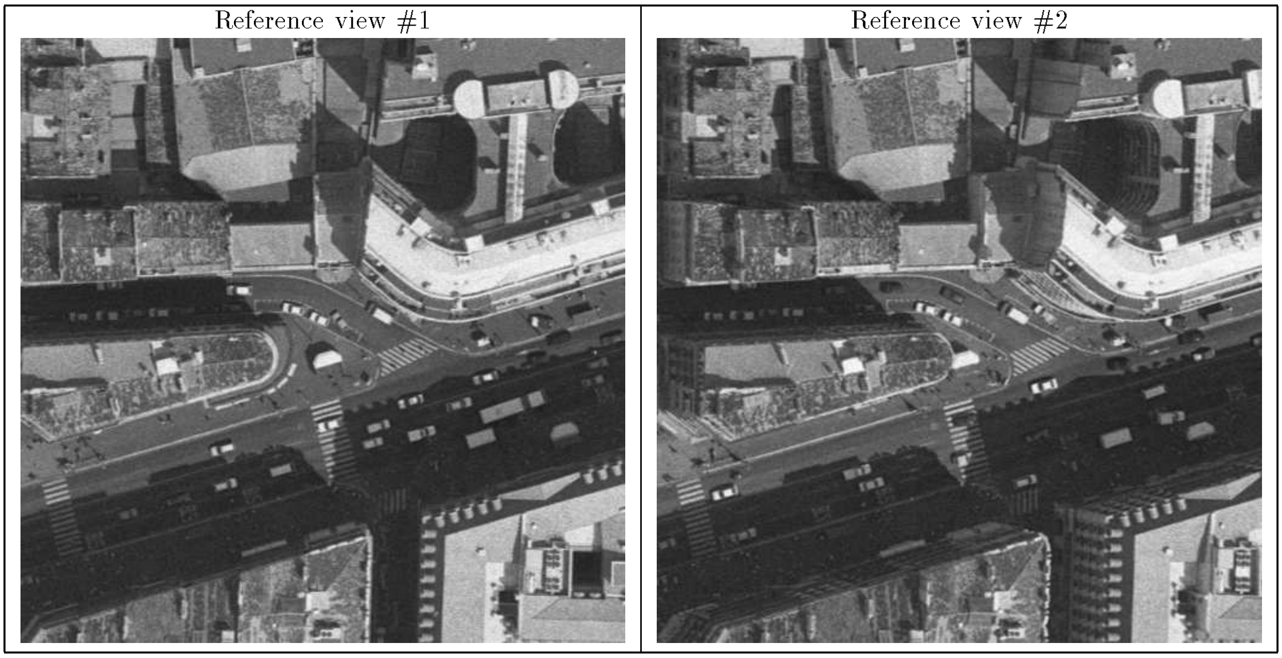

Figure 1: the 2 reference views.

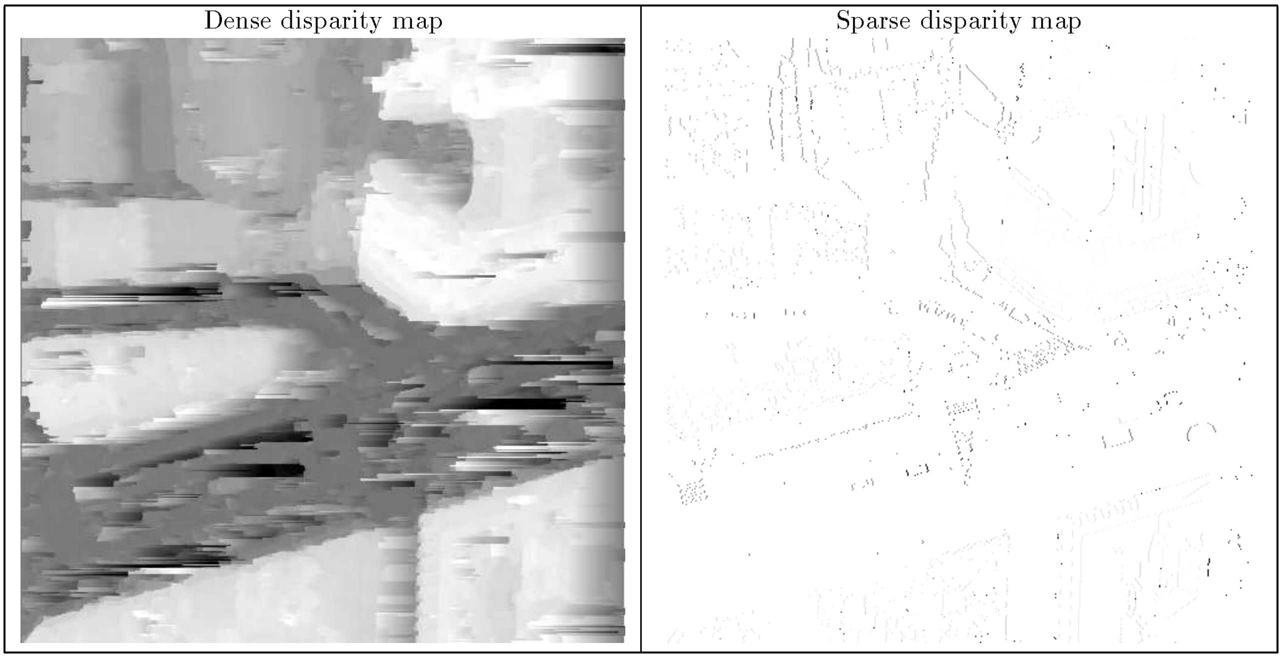

Figure 2: computed disparity maps.

Computing the Triangular Mesh We used several methods to compute the triangular mesh.

- We used a plain Delaunay triangulation. The problem is that each triangle area does not necessarily cover a plane in the reference views. This can lead to strange artifacts in the synthesized (texture-mapped) views. We are still developing automatic methods to ensure this; a way to do is to impede triangles from crossing occluding edges.
- This leads us to another method: a constrained Delaunay triangulation, where we set boundaries on the occluding edges. We need to detect precisely the occluding edges in the reference views, and this part is still under development.
- Our last method is experimental: for each triangle in an initial Delaunay triangulation, we project it to the second reference view, and compute the image difference. To do this, we just subtract the images, then filter them using an erosion/dilation operator to remove isolated insignificant points, and decide whether this difference is null (under a fixed threshold). We experimented this method with success, which is however excessively time consuming.

4 Synthesizing New Views - Results

4.1 From the Cloud of Points

We have enough to synthesize new views of the scene: we just project the points onto the plane of a virtual camera. We get some usual problems; for instance, when we get nearer, the pixels tend to part, and let the pixels behind show through. This technique is quite fast (~ 1 second to synthesize a 500×500 picture), but we need to apply a filter afterwards.

Some synthesized images can be seen on figure 3.

4.2 From Triangles

On another model (the Movi house), we experimented the triangle technique, generating a Vrml file we displayed with VrWeb. We used this model because it is well adapted to flat triangles, and because we could get enough precision in the matches. Using a standard tool as VrWeb allows to benefit from special hardware as on Silicon Graphics workstations, especially for anti-aliased texture mapping.

The triangulation can be seen on figure 4; the points were matched and refined automatically. For this experiment the triangles were connected by hand, helped by an initial Delaunay triangulation, although we could have used the experimental technique described before.

Some synthesized views of the Movi house can be seen on figure 5.

5 Conclusion

We proposed a method to use real views of the scenery as models for image synthesis. That is to say, from some photographs of a real scene, we can build a synthetic model, allowing to virtually navigate inside the scene and appreciate the tri-dimensional structure.

The advantages are speed and realistic effects.

Speed in the pre-processing stage: extracting the 3D model from the photos is mostly automated, thus avoiding a tedious modeling stage by hand.

Speed in the rendering process: as real photos are mapped to synthesize new views, we don't need a sophisticated rendering computation; this information is already available in the photos.

Realistic viewing: for the same reason, using real images as a basis ensures a realistic viewing. A fine level of detail is impossible to achieve in a reasonable time using a model which would be hand-coded from scratch; if we did so, we'd need to encode the details of the rocks and dust on the ground, the leaves of the trees, etc...

Future Work At this stage, we have sparse points in the 3D euclidean space. An automated and fast building of the triangle mesh having these points as vertices, and texture extracted from the reference views, is still to be developed.

Acknowledgements

This work was partially supported by the Heros network of the EC Hcm program.

References

[Asc 92] P. Aschwanden and W. Guggenbühl. Experimental results from a comparative study on correlation-type registration algorithms.

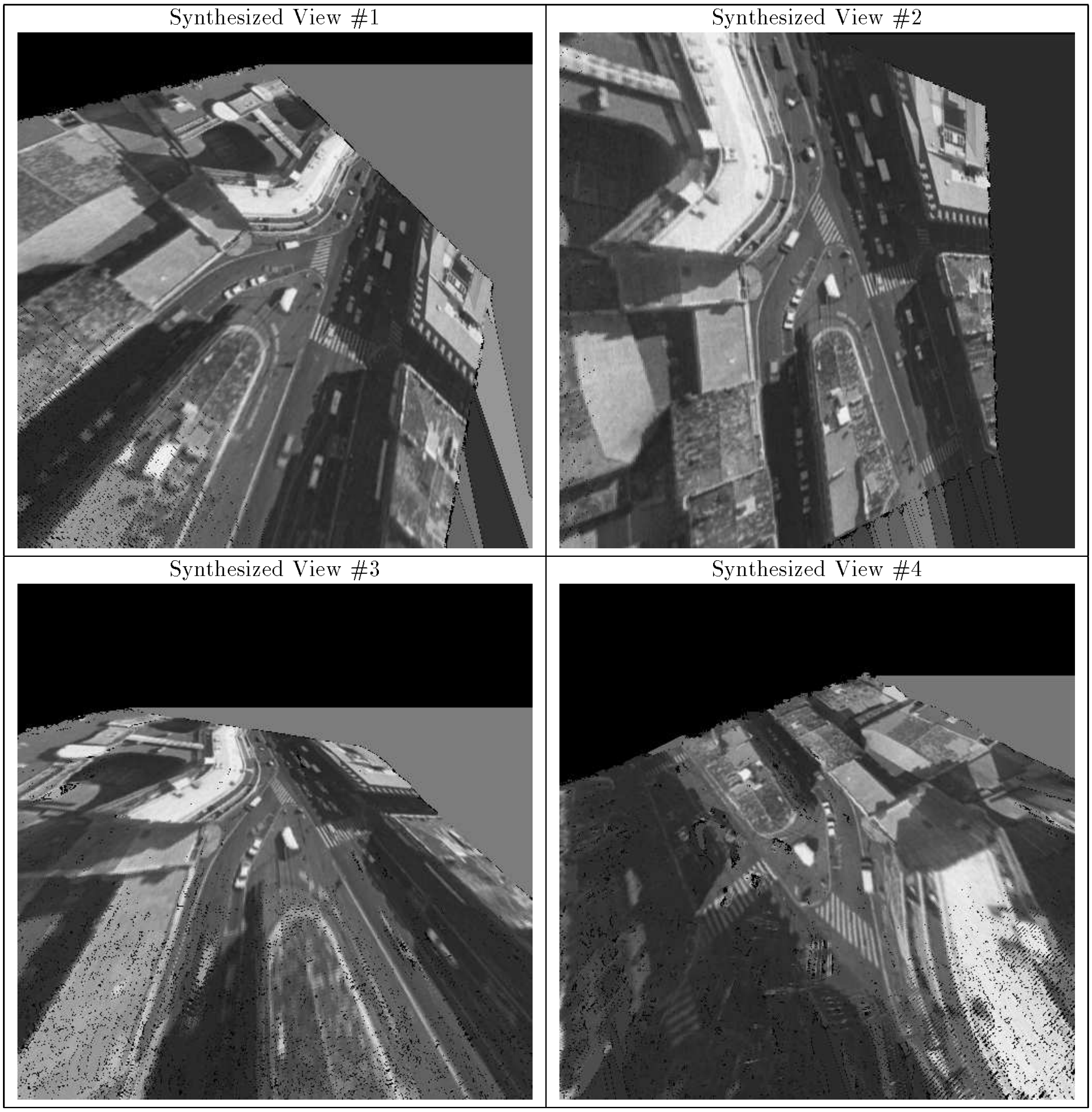

Figure 3: some computed views using only points.

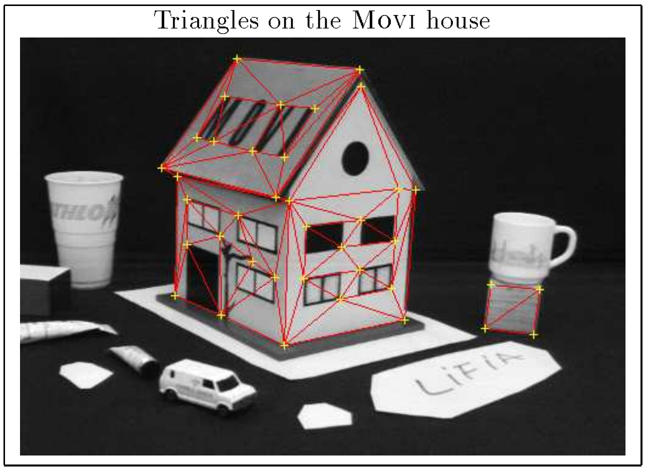

Figure 4: the generated triangulation.

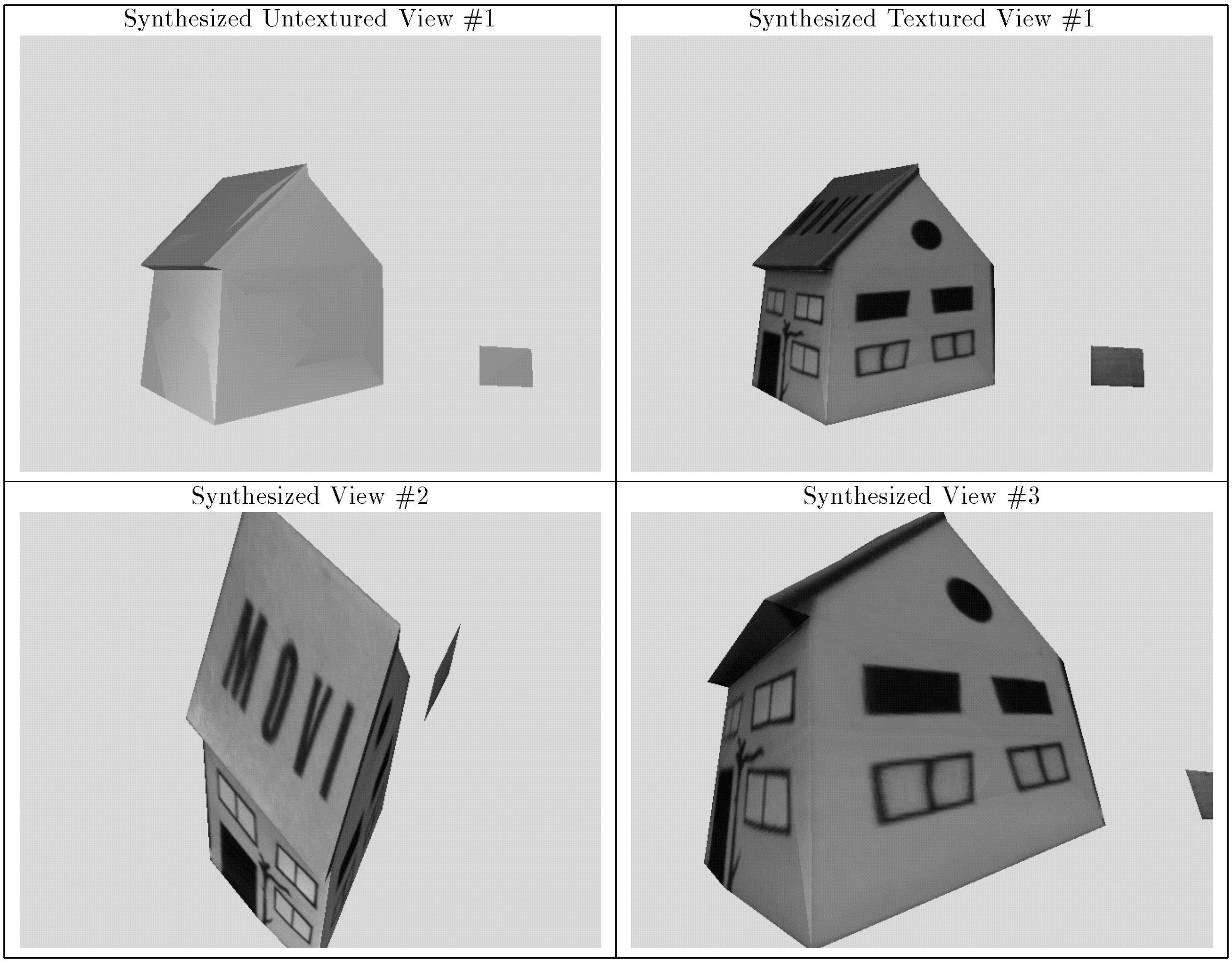

Figure 5: some computed views using triangles.

In Förstner and Ruwiedel, editors, *Robust Computer Vision*, pages 268–282. Wichmann, 1992.

[Bal 96] J.F. Balaguer. VRML for LHC engineering. In *Journées Nationales du Groupe de Travail "Réalité Virtuelle" - Toulouse, France*, pages 67–72, October 1996.

[Bou 93] B. Boufama, R. Mohr, and F. Veillon. Euclidean constraints for uncalibrated reconstruction. In *Proceedings of the 4th International Conference on Computer Vision, Berlin, Germany*, pages 466–470, May 1993.

[Der 94] R. Deriche, Z. Zhang, Q.T. Luong, and O. Faugeras. Robust recovery of the epipolar geometry for an uncalibrated stereo rig. In J.O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden*, volume 800 of *Lecture Notes in Computer Science*, pages 567–576. Springer-Verlag, May 1994.

[Fau 92] O. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pages 563–578. Springer-Verlag, May 1992.

[Har 94] R. Hartley and P. Sturm. Triangulation. In *Proceedings of* ARPA *Image Understanding Workshop, Monterey, California*, pages 957–966, November 1994.

[Har 95] R. Hartley. In defence of the 8-point algorithm. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 1064–1070, June 1995.

[Luo 92] Q.T. Luong. *Matrice fondamentale et autocalibration en vision par ordinateur.* Thèse de doctorat, Université de Paris-Sud, Orsay, France, December 1992.

[Sch 96] C. Schmid. *Appariement d'images par invariants locaux de niveaux de gris.* PhD thesis, Institut National Polytechnique de Grenoble, July 1996.