in: *Proceedings of the 5th International Workshop on Information Optics (WIO 2006)*, Toledo, Spain, June 5-7, 2006.

An Object Acquisition Library for Uncalibrated Cameras in C++

Klaus Häming and Gabriele Peters

University of Dortmund, LSVII, Otto-Hahn-Str. 16, D-44227 Dortmund

Abstract. The task of recovering the three-dimensional structure of a scene by images of that scene alone is known as the reconstruction problem. To solve this problem it is crucial to understand the geometry of multiple views and to implement the related algorithms. In the past years, many advanced algorithms were developed and published by the computer vision community. As a result, understanding the whole topic well enough to implement the underlying algorithms may mean a prohibitive, time consuming task. This is relevant especially for computer vision students, whose primary research interests *base* on projective or metric reconstruction. Those may want to use them as mere utilities. So, as always in such a situation, it is time to provide a library, easy to use for the beginner, but flexible enough for the expert. Moreover, such a library would provide a gentle introduction to the computational aspects of the topic. This paper presents the design of such a library, that mainly uses one central data structure and a collection of operators affecting it.

Keywords: computer vision, object acquisition, view reconstruction, multiple view geometry
PACS: 89.20.Ff

1. INTRODUCTION

Papers covering fully automated processes of object reconstruction from images [2, 5, 11] and comprehensive books such as [7] show, that this problem has been solved in the past years. The presented library implements the building blocks of such a reconstruction process, allowing to combine them easily. Because the aspects of this process are manifold, a very brief introduction to the most important ones is given. These aspects are named and put in the correct context, and then the reader is referred to the literature.

The usual and widely used techniques to reconstruct objects from their images, base on the use of the fundamental matrix F and the trifocal tensor T. These two geometric entities encode the epipolar geometry of the scene, based on a two-view relation in the case of F, on a three-view relation in the case of T. The computation of these two multi-view tensors is based on interest points or lines matched throughout a series of images. Once these feature correspondences are known, it is possible to compute F and T using these equations (given in tensor notation):

$$\begin{aligned} x^i x'^j F_{ij} &= 0 \\ x^i x'^j x''^k \varepsilon_{jqs}\varepsilon_{krt} T_i^{qr} &= 0_{st} && \text{(points)} \\ l_p l'_q l''_r \varepsilon^{piw} T_i^{qr} &= 0^w && \text{(lines)} \end{aligned}$$

where x, x', and x'' are interest points, and l, l', and l'' lines in the first, second, and third image, respectively. The indices i, j, k, p, q, r, s, t, and w all range from 1 to 3, because we deal with homogeneous 2D entities (e.g., x^i is a 3-vector, 0_{st} is a 3x3 array of zeros, and the tensor T_i^{qr} can be seen as a 3x3x3-array).

It can be seen, that F relates only points, while T can also make use of lines (in fact, there are mixed equations of points and lines for T, that are omitted for brevity). Once computed, F and T allow the extraction of generic cameras. After triangulation a projective reconstruction [7] is achieved. Because these reconstructions cover only image pairs or triplets, a subsequent merging step combines them into larger sequences[5]. Then, the projective reconstruction can be improved to be oriented [8, 20, 7] (basically meaning all points are in front of all cameras) and metric [18, 12, 14]. After that it resembles the structure of the observed object up to a similarity (translation, rotation, and scale).

So, the computation of the structure is based on features and their matches. Thus, to get a successful reconstruction, a great effort has to be put in computing these initial feature correspondences. While small baseline feature tracking can be considered solved (e.g., using the common Kanade-Lucas tracker, as described in [16]), wide baseline matching is still a little fiddling. This is because it is not easily possible to limit the search region, unless a good motion estimation has been achieved. As a consequence, a large search region gives rise to potentially many false matches and increases the running time.

The search region problem is often addressed by comparing a few points only, mostly strong corners. Often a good matching is achieved by using affine invariant descriptors and trying to make them as distinctive as possible [1, 9, 10]. Another approach is to use a modified Kanade-Lucas tracker [21] and initialize it with good initial guesses of the affine distortion using local Hough transform [15]. Then, the features can be used to vote for the most probable geometry using for example Hough transform [9], RANSAC [7], or tensor voting [17]. This information is then used to eliminate most false matches. At the end, a hopefully good estimation of F or T is found and can be used in the reconstruction process.

2. COVERED TERRAIN

The library addresses the hole process outlined in the last section. Most of the referred papers where implemented and obvious weaknesses were refined. One of these obvious weaknesses is the usage of gray-value image gradients, even while processing multi-channel images. Therefore, the note of Silvano Di Zenzo [22] was used in algorithms using gradients such as Canny edge detection [3], Harris features [6], and SIFT [9]. This gradient sure has an orientation, but lacks a direction. Therefore, the direction was added by a simple convention: The direction is defined by the first channel for which holds true that the two pixels closest to the edge have different values. The direction points then from low to high.

However, while designing the library we focus on the reconstruction part. Therefore, the feature matching is currently not quite sophisticated. In fact, only SIFT and a simple cross-correlation tracker initialized with Harris corners are used. The voting for the most probable geometry is carried out using RANSAC, improved by using a check on oriented geometry [4].

The reconstruction part features robust computation of fundamental matrices and trifocal tensors [7], polar rectification [13], a highly customizable version of Pollefeys' auto-calibration [12], and a flexible implementation of bundle adjustment [19]. The merging algorithm follows [5] and provides every option mentioned there (i.e. using cameras, points or a combination of both to estimate the merging homography and optionally improving the result non-linearly).

3. DESIGN

During reconstruction a very limited number of entities is important, namely

views These contain the images as received from the camera(s) and the interest points.

correspondences These are one to one mappings from views to features. If a correspondence maps a view to *nil*, that view doesn't contribute a feature.

cameras They are associated with views. More than one camera can be associated with a view because a view can be part of multiple reconstructions.

3D points Cameras and correspondences can be used to triangulate 3D points.

tensors These encode the epipolar geometry of a given subset of views. Currently only the fundamental matrix and the trifocal tensor are supported.

One central data structure, called *NView*, manages the correspondences, cameras, 3D points, and the tensor of a subset of views. Because only the smallest building blocks of the reconstruction contain one, a tensor is unique to an additional specialization called *NViewSet* (from *NView* and subSet). This data structure can be used to perform queries such as: "What is the current camera matrix of the third view of the particular subset of views that share a particular tensor?"

To address an *NView* by the subset of views whose scene it represents, a class providing the mapping $\{\text{view}_{i_1}, \text{view}_{i_2}, \ldots, \text{view}_{i_n}\} \rightarrow \text{NView}_k$, with i_j members of some index set, is provided. This class is called *NViewMap* and can be used to manage the views as well. The *NView* is designed to support a hierarchical approach to the reconstruction problem as stated in [5]. Therefore *NViews* can be organized as a tree and the *NViewMap* provides additional queries such as "find all roots/leafs". There are no more classes that manage or reference the geometric data.

Beside the *NView*, the most important and largest group of classes consists of so called *Operators*. These are used to modify an *NView*, hence their common method is `modify(NView &nview)`. Currently, the following *Operators* are available:

Autocalibrator Upgrades from projective to metric reconstruction

BundleAdjusterMetric Performs bundle adjustment using a metric parameterization of the cameras.

BundleAdjusterProjective Performs bundle adjustment taking cameras as matrices.

CameraCalculator Calculates cameras from tensors or via resectioning.

Point2DCorrector Recalculates the 2D points from cameras and 3D points.

Point3DCalculator Calculates the 3D Points via triangulation.

Quasiaffinator Upgrades from projective to quasi-affine reconstruction.

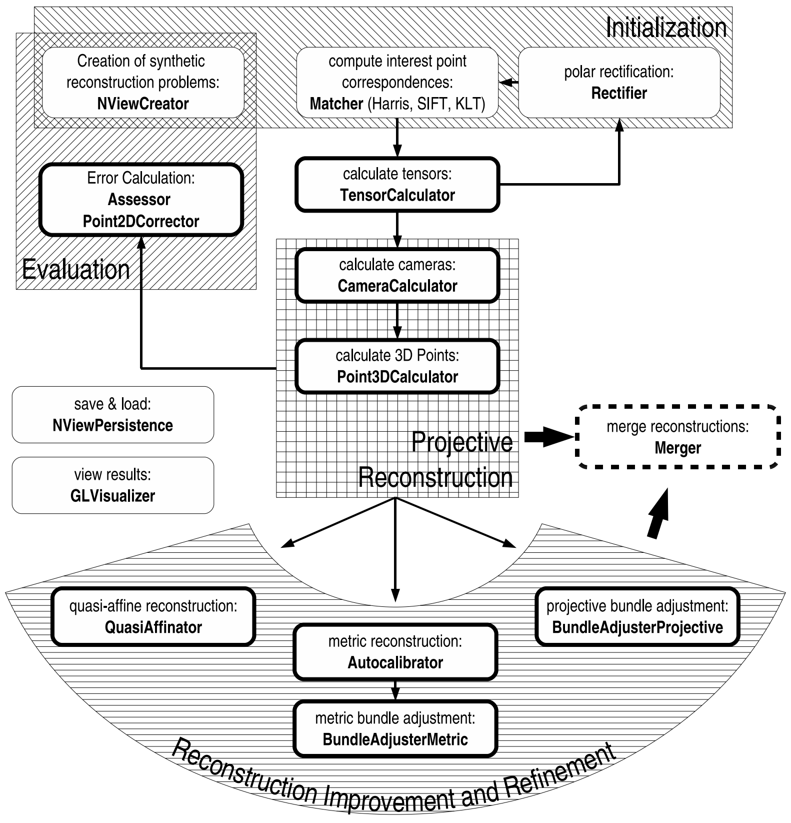
Initialization
Creation of synthetic reconstruction problems: NViewCreator
compute interest point correspondences: Matcher (Harris, SIFT, KLT)
polar rectification: Rectifier
calculate tensors: TensorCalculator
Error Calculation: Assessor Point2DCorrector
Evaluation
calculate cameras: CameraCalculator
calculate 3D Points: Point3DCalculator
save & load: NViewPersistence
view results: GLVisualizer
Projective Reconstruction
merge reconstructions: Merger
quasi-affine reconstruction: QuasiAffinator
projective bundle adjustment: BundleAdjusterProjective
metric reconstruction: Autocalibrator
metric bundle adjustment: BundleAdjusterMetric
Reconstruction Improvement and Refinement

FIGURE 1. The main building blocks of the library. These classes can affect *NViews*. The *Operators* feature a thick border.

TensorUpdater Calculates the tensor of an *NViewSet* from interest point correspondences.

The relation of these operators to the reconstruction pipeline is visualized in Fig. 1. So, in most cases it is sufficient to choose a type of reconstruction, parameterize the appropriate Operators and call them in the correct order, e.g:

```
matcher.match(views, nview); (initialization by a matcher)
tensorCalculator.modify(nview);
projReconstructor.modify(nview);
metricReconstructor.modify(nview);
```

To use *Operators* on multiple *NViews* there is a class called *Modifier* that will do this for you. There are other classes that take an *NView* as an argument and are not *Operators*. The most notable may be those for generating a synthetic reconstruction problem (NViewCreator), loading and saving the *NView* as an XML file (NViewPersistence), and visualizing it via OpenGL (GLVisualizer). The latter class uses an OpenGL context as provided by many GUI toolkits such as QT or WxWidgets, but it does not manage a window itself (which would make it much less flexible).

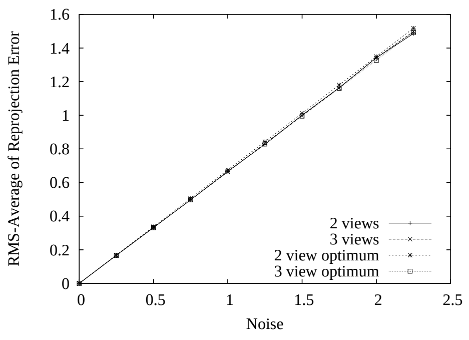

FIGURE 2. Re-projection error after the calculation of a projective reconstruction. The standard deviation of the noise is given in pixels. The image size is 500x500. The result is nearly indistinguishable from the theoretical optimum.

4. EVALUATION

After the 3D points and cameras are computed, the 2D points can be re-estimated by projection. The distance between the original 2D points (x) and the re-projected ones ($\hat{x}$) is known as the re-projection error:

$$\varepsilon_{repro} = \sum_{v=0}^{V} \sum_{p=0}^{P} \operatorname{dist}(x_p^v, \hat{x}_p^v)$$

where V denotes the total number of views, P the number of interest point correspondences and $\operatorname{dist}(\cdot,\cdot)$ the (usually euclidean) distance.

This error is used to assess the quality of the implementation. The algorithms are tested on synthetic data with Gaussian noise added to the interest point correspondences. This allows us to compare the result to the theoretical optimum, that can be computed from

$$\varepsilon_{opt} = \sigma \cdot \operatorname{sqrt}((1 - \frac{d}{N})),$$

where σ denotes the standard deviation of the added Gaussian noise, d the number of degrees of freedom, and N the number of measurements. In the case of two views, using the fundamental matrix (which has seven degrees of freedom), we have:

$$\begin{aligned} d &= 7 + 2 \cdot \text{number of correspondences} \\ N &= 4 \cdot \text{number of correspondences} \end{aligned}$$

In the case of three views, using the trifocal tensor (which has 18 degrees of freedom), we have:

$$\begin{aligned} d &= 18 + 3 \cdot \text{number of correspondences} \\ N &= 6 \cdot \text{number of correspondences} \end{aligned}$$

Figure 2 shows the RMS-average of the re-projection error at a given noise level for a two and three view reconstruction problem. The averaging was carried out over 100 runs. A comparison of the result to the theoretical optimum yields almost no difference between them.

5. CONCLUSION

The design of a library addressing multiple view geometry was presented. It includes state of the art algorithms that solve the reconstruction problem arising from images of unknown cameras. Figure 3 shows the result of actually applying the library to images taken of two objects. The 3D points computed from the found correspondences were triangulated and the triangles were textured using parts of the images. The points are superimposed as white dots.

So, once the interest point calculation was successful, the library can easily be used as a black box. Further improvements will mainly address the feature matching and outlier rejection, because the current tracker needs many RANSAC iterations or finds only a few correspondences when using SIFT. The approaches presented in [21, 17] are promising enough to evaluate them. The goal is to establish a collection of algorithms that allow a fully automated reconstruction based on a set of images.

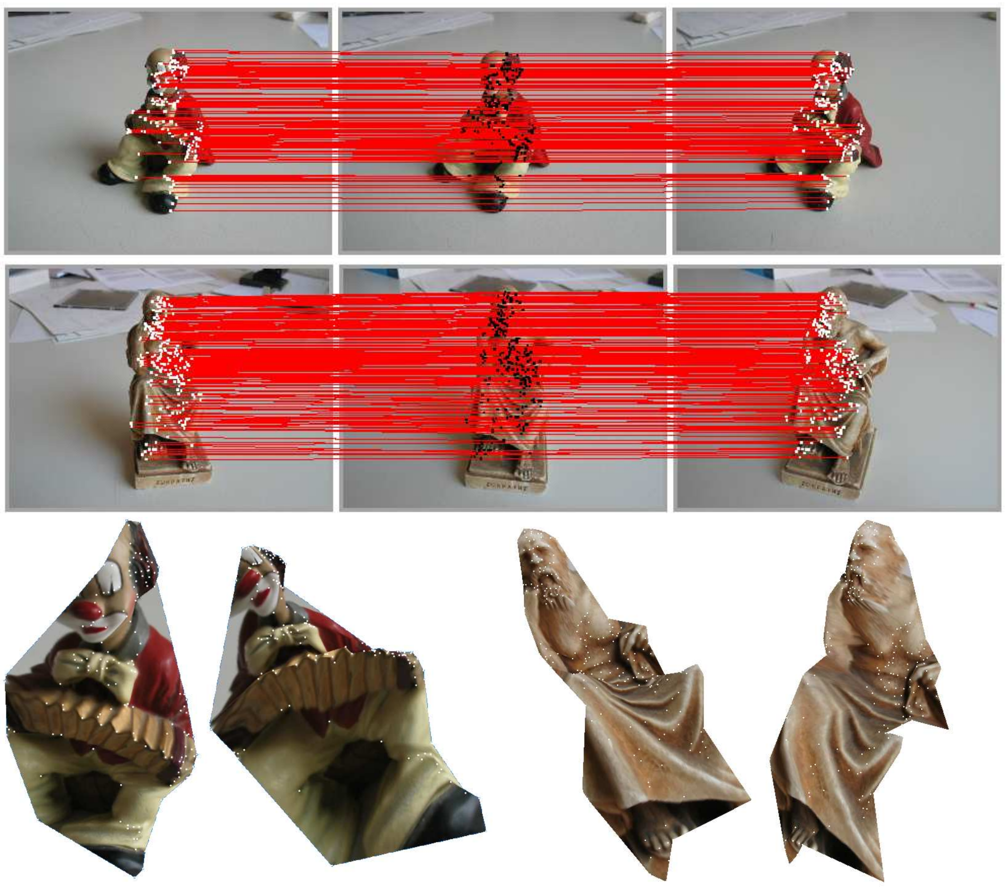

FIGURE 3. Corresponding features and their reconstruction

REFERENCES

1. Adam Baumberg. Reliable feature matching across widely separated views. *cvpr*, 01:1774, 2000.
2. Paul A. Beardsley, Philip H. S. Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, pages 683–695, London, UK, 1996. Springer.
3. J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.
4. Ondrej Chum, Tomás Werner, and Jiri Matas. Epipolar geometry estimation via ransac benefits from the oriented epipolar constraint. In *ICPR (1)*, pages 112–115, 2004.
5. A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *Proceedings of the European Conference on Computer Vision*, pages 311–326, June 1998.
6. C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *4th ALVEY Vision Conference*, pages 147–151, 1988.
7. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
8. Stéphane Laveau and Olivier D. Faugeras. Oriented projective geometry for computer vision. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 147–156, London, UK, 1996. Springer.
9. David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.
10. Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.
11. Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3):207–232, 2004.
12. Marc Pollefeys, Reinhard Koch, and Luc J. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *ICCV*, pages 90–95, 1998.
13. Marc Pollefeys, Reinhard Koch, and Luc J. Van Gool. A simple and efficient rectification method for general motion. In *ICCV*, pages 496–501, 1999.
14. Jean Ponce, Theo Papadopoulo, Monique Teillaud, and Bill Triggs. On the absolute quadric complex and its application to autocalibration. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages I 780–787, June 2005.
15. F. Shen and H. Wang. A local edge detector used for finding corners. *Proc. ICICS*, 2001.
16. Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.
17. Chi-Keung Tang, Gérard Medioni, and Mi-Suen Lee. N-dimensional tensor voting and application to epipolar geometry estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):829–844, 2001.
18. Bill Triggs. Autocalibration and the absolute quadric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 609–614. IEEE Computer Society Press, June 1997.
19. Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372, London, UK, 2000. Springer.
20. Tomáš Werner, Tomáš Pajdla, and Václav Hlaváč. Oriented projective reconstruction. In M. Gengler, M. Prinz, and E. Schuster, editors, *Pattern Recognition and Medical Computer Vision: 22-nd Workshop of the Austrian Association for Pattern Recognition (ÖAGM/IAPR)*, pages 245–254, Wien, Austria, May 14–15 1998. Österreichische Computer Gesselschaft.
21. Jiangjian Xiao and Mubarak Shah. Two-frame wide baseline matching. In *ICCV*, pages 603–609, 2003.
22. Silvano Di Zenzo. A note on the gradient of a multi-image. *Comput. Vision Graph. Image Process.*, 33(1):116–125, 1986.