FIT3D Toolbox: multiple view geometry and 3D reconstruction for MATLAB

Isaac Esteban[1,2] and Judith Dijk[2] and Frans Groen[1]

Abstract— **FIT3D is a Toolbox built for Matlab that aims at unifying and distributing a set of tools that will allow the researcher to obtain a complete 3D model from a set of calibrated images. In this paper we motivate and present the structure of the toolbox in a tutorial and example based approach. Given its flexibility and scope we believe that FIT3D represents an exciting opportunity for researchers that want to apply one particular method with real data without the need for extensive additional programming.**

I. INTRODUCTION

The process of reconstructing the scene depicted by a set of images is usually called Structure from Motion [2] (SFM). It is related to the more traditional photogrametry field in which artificial markers are used within the real scene in order to obtain a high accurate reconstruction. SFM on the contrary does not rely on unnatural artifacts, rather it makes use of the geometry of the camera and the scene itself in order to reconstruct the imaged world. However, the number of steps required to obtain a full 3D model and the complexity and method variety in each of those steps enforces a disjoint field. There are many publications and variations of methodologies that aim at solving a certain aspect of the process of reconstructing a scene. We characterize the complete process in 5 differentiable steps: *camera calibration*, *motion estimation*, *optimization*, *reconstruction* and *modeling*.

Each step is a subfield in its own, which together with the complexity of the problem translates into a lack of unification and integration of the literature and more importantly of the available tools. This lack of global framework makes very difficult for researches to develop and test their specific techniques in a global context. For this reason we present the FIT3D Toolbox that aims at bridging the gap between the five independent subfields, providing a unifying framework. FIT3D is a highly flexible Toolbox built for Matlab. The functionality provided is either based on well founded research or exciting and promising novel approaches. The toolbox consist of a large number of independent Matlab functions written in an educative and clear fashion and contain a large amount of inline comments. References to the original methods are provided within the code. In our spirit of integration and distribution, we provide example scripts and data sets for every of the five steps as well as the integration with related projects. This article is written in a tutorial form and assumes that the reader is familiar with Matlab and understands the concept of Structure from Motion and the geometry involved. Section II relates and compares FIT3D to similar frameworks. The integration with related packages is also presented. In Section III we discuss in more detail each of the five steps of the FIT3D pipeline along with the conventions and definitions. Section IV focusses on the first three steps, camera calibration, motion estimation and iterative refinement, while in Section V the 3D modeling process is explained. Finally in Section VI we draw some conclusions and point out future work. FIT3D will be available from April 2010 at the website (web not shown due to double blind revision).

Intelligent Systems Laboratory Amsterdam[1] (ISLA), University of Amsterdam

Electro-Optical Systems[2], TNO Defence, Security and Safety {isaac.esteban, judith.dijk}@tno.nl and groen@science.uva.nl

II. RELATED WORK

In recent years the literature on reconstructing in a 3D model the scene recorded in images has

Electro-Optical Remote Sensing, Photonic Technologies, and Applications IV, edited by Gary W. Kamerman, Ove Steinvall, Keith L. Lewis, Richard C. Hollins, Thomas J. Merlet, Gary J. Bishop, John D. Gonglewski, Proc. of SPIE Vol. 7835, 78350J · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.864112

grown considerably. Along with this growth, a number of tools or software packages have been made available to the research community, and in some cases, to the general public. However, these packages are either very specific to the task at hand or are enclosed in a format that makes integration with novel techniques virtually impossible.

Photo Tourism is one of the most well known programs available. It provides a sparse 3D reconstruction from a set of unordered images of the same scene. The techniques used for estimating the relative motion between the images are well known and are also available in a more scientific package called Bundler [12]. Bundler is built in C++ for efficiency and it takes a collection of images, image features and image matches as input and produces a sparse 3D pointcloud. The optimization step is performed with a modified version of the Sparse Bundle Adjustment of Lourakis and Argyros [9]. FIT3D presents a number of advantages using novel motion estimation techniques and refinement procedures. It also provides a complete system where an ordered set of images needs to be provided and a complete model is obtained. Additionally, FIT3D is built as a set of independent functions and example scripts in Matlab, which makes the integration with novel techniques an easy task.

Hartley and Zisserman [7] is one of the most relevant references in the field of multiple view geometry and SFM. With the second edition they provide some Matlab code to perform a number of operations. However the number of functions is very limited with a scarce amount of inline comments and only a handful of example scripts. No full 3D models can be obtained from the raw material they provide. FIT3D is similar to these in that the functions perform very specific tasks and are coded in Matlab. FIT3D however is different in that we provide a larger amount of functions with more comprehensive inline comments and references. Data sets and example scripts are also provided.

Peter Kovesi [8] provides a number of Matlab functions for both image processing, multiple view geometry and model fitting. The inline code provided by Kovesi is very generous in the input/output/example but somewhat short in some of the inner workings of the functions. No examples scripts are provided. FIT3D overlaps with Kovesi's work in the area of model fitting and projective geometry. FIT3D extends Kovesi's work in that it is focussed on the full process of 3D reconstruction and provides full examples on the use of the Toolbox.

With respect to the integration to other packages, FIT3D is fully compatible with the Camera Calibration Toolbox of Bouguet [1]. We also provide integration with the SIFT implementation provided by Lowe [10] and the open source VLFeat [14], the 5-point algorithm from Nister [11] available online and provide interface tools for PMVS2 [6].

III. Preliminaries

Before we begin with each of the steps in the process of building a 3D model we need to understand the basic geometry of the problem at hand. In this section we present the basic concepts and for more advanced descriptions we refer the reader to [7].

A. Conventions and Definitions

For the problem of reconstructing a scene depicted by two or more images we adopt the pinhole camera model (PCM). In this model the camera is considered a point in space towards which all light rays are directed to. This point is referred as camera center (CC). In this setup, the image is the result of the projection of those light rays intersecting a plane (the image plane) located at a certain distance from the CC. We consider 3 different reference systems (see Figure 1). The first one describes the positions of cameras and objects in the 3D space and we call it the World Coordinate frame (WCf). The second one is fixed in the CC with a given orientation and is called the Camera Coordinate Frame (CCf). The last coordinate system is located at the image level and is called Image Coordinate frame (ICf). Following Lowe's convention for image feature locations, we set the ICf center to be at the TOP/LEFT corner of a given image. This convention also follows Matlab's convention. For convenience, we set the image plane perpendicular to the axis Z of the CCf. The distance from the CC to the image plane is called the focal distance (f).

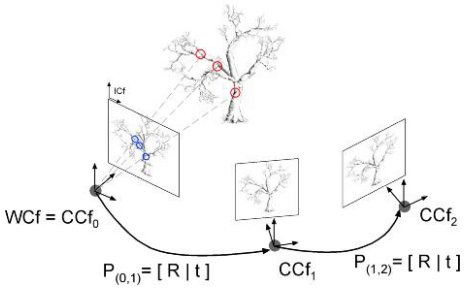

Fig. 1. Coordinate systems and conventions on camera motion.

Given that we need more than one image to reconstruct the scene, we consider the case where multiple cameras and images are present (See Figure 1). In particular, we are interested in the geometrical relation between different camera centers. We define the displacement between two CCs as a matrix P. This matrix consist of a rotation R and a translation t in the form $P = [R|t]$. For convenience, we always set the first image at the WCf. Therefore, the geometrical relation of an arbitrary camera with the first camera is just a rotation and a translation with respect to WCf.

In order to be able to project objects into the image plane we need to know the distance between the image plane and the CC. We need to consider that the image plane is a finite element consisting of pixels. Also, the origin of coordinates might not coincide with the Z axis of the CCf. Even further, the aspect ratio of the pixels might not be square. This is encoded in the calibration matrix K defined as:

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Together with the relative displacement of an arbitrary camera, they conform the Camera Matrix (CM) $P^\star$:

$$P^\star = K[R|t] \quad (2)$$

Considering these definitions we study the process of building a 3D model from a set or ordered images. During this process, a camera moves through the environment recording images. We assume that there is enough overlap between three consecutive frames. The process of building a 3D model out of the recorded images follows then the five steps of calibration, motion estimation, refinement, reconstruction and modeling.

B. The 3D reconstruction pipeline

In this section we briefly describe each of the steps in the 3D modeling pipeline (see Figure 2). We divide the steps in two mayor blocks. The first one comprises the methods and techniques required to obtain an accurate and consistent estimation of the relative position between the consecutive images. The second entitles the required techniques for obtaining the final 3D model.

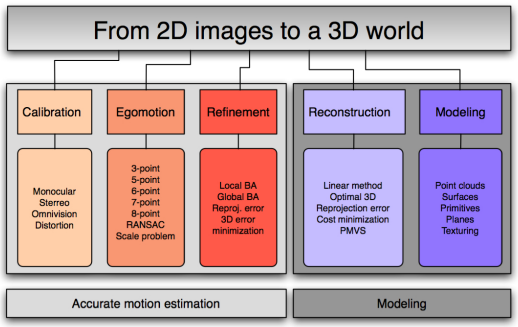

Fig. 2. 3D modeling pipeline. Some well known alternatives for each step are pointed out.

- **Calibration:** calibration is the process of obtaining the information, intrinsic parameters, that describes the camera. This includes parameters for radial distortion, focal length, skew parameters and center of projection.
- **Egomotion:** is the process of estimating the motion of the camera between frames based on the information that is depicted in the images (extrinsic parameters). In our work we only consider motion estimation between consecutive frames.
- **Refinement:** given that the images are recorded with noise and this noise is propagated to the motion estimation, an optimization step is desirable when computationally feasible in order to improve the estimates.
- **Reconstruction:** given a set of camera positions, calibration parameters and recorded images, reconstruction is the process of obtaining the 3D representation of the scene as a set of 3D points.

- **Modeling:** it is desirable and sometimes necessary to obtain an economical description of the scene. Modeling is the process of converting a sparse 3D point cloud into a set of higher order primitives such as planes, surfaces or pre-calculated models.

IV. Motion estimation and scale consistency

Motion estimation is the process of estimating the spatial geometrical relation between two or more given images. This relation can be estimated using image feature matches. For this to succeed, the intrinsic parameters of the camera need to be either known or estimated. We assume they can be estimated before hand and refined later if necessary. In this section we describe the basics of camera calibration and the process of estimating the motion between two images including the scale ambiguity.

A. Camera Calibration and Radial Distortion

The process of estimating the intrinsic parameters is commonly referred as camera calibration. The set of camera parameters that are relevant for motion estimation are divided in two groups. The first group is related to camera calibration matrix K, where focal length, pixel aspect ratio and center of projection are estimated. The second group is the lens distortion parameters. This distortion can be irregular or follow a pattern. The most common type of distortion is the radial distortion where the distortion follows a radial pattern around the center of distortion.

A common method to estimate these parameters is using a calibration pattern. Given the some images of this pattern and knowing its dimensions the parameters can be estimated. For an in depth discussion of the calibration process we refer the reader to [15].

It is important to note that the radial distortion can be described with a mathematical model as a transformation function from distorted pixels to undistorted pixels or the other way around. Zhang [15] describes the radial distortion model as a function from correct image to the distorted image. Hartley et. al. [7] however define the function from the distorted image to the undistorted image. These two models are not the inverse of each other and only when the distortion is small, a taylor series approximation can be calculated with a small error.

FIT3D employs the calibration as estimated with [1] using Zhang method [15] . However, since the inverse radial distortion model is also useful, FIT3D provides the means to calculate the radial distortion from the distorted image to the undistorted image by means of selecting a set of straight lines in an image. After these lines are selected, the model up to fifth order is calculated and the image is corrected.

1) EXAMPLE: obtain radial distortion based on straight lines in the image: we employ a distorted image of a street where 3 straight lines consisting of 3 points are selected. From this information the distortion parameters are computed and the image is corrected.

Fig. 3. LEFT: distorted image with selection of straight lines. RIGHT: undistorted image.

```
>> [distParams,undistImg] =
getRadialDistortion('street.jpg',3,3);
```

B. Motion estimation

Having obtained the intrinsic calibration parameters and having compensated the images for radial distortion, we are now ready to estimate the spatial relation between a pair of images. There are a number of algorithms and techniques to do so [11] [7]. In FIT3D we focus on the use of the normalized 8-point algorithm [7] given that it is a linear method that yields up to 4 possible solutions. The correct solution is obtained by a voting mechanism where each image feature votes for the solution in which the reconstructed 3D feature is in front of both cameras. The 8-point algorithm is however very sensitive to images where all feature matches are projections of spatial points that lay in the same plane. For this situation we

take advantage of the frame-to-frame refinement (see Section IV-D).

Other well known alternatives can also be used for motion estimation. FIT3D is integrated with the Matlab version of the 5-point algorithm provided by Nister [11] and also provides methods for robust computation of the homography. RANSAC [5] is used for robust outlier rejection, a SIFT detector is used for selecting image features and SIFT descriptors are used for the matching process.

1) EXAMPLE: stitch two images using an Homography: The `stitchPano` function will stitch two images by selecting corresponding SIFT features and calculating the homography.

```
>> [panoramic,H] =
stitchImages('ind1.jpg','ind2.jpg',0.4);
>> imshow(panoramic);
```

Fig. 4. TOP: two input images. BOTTOM: stitched images.

2) EXAMPLE: 8-point algorithm motion estimation using RANSAC: we assume we have a set of matching features $X1$ in image 1 and $X2$ in image 2. We obtain the fundamental matrix using RANSAC, then obtain the 4 possible solutions for the camera motion and obtain the true solution using a voting system.

Fig. 5. TOP: Three views of a street. BOTTOM: calculated camera trajectory.

```
>> [F,inliers] = ransacF(X1,X2,K);
>> Pall = getCameraMatrix(F);
>> X1i = X1(inliers,:);
>> X2i = X2(inliers,:);
>> P = getCorrectCameraMatrix(X1i,X2i);
```

C. Linear scale estimation

Given the nature of the pinhole camera model, there is a scale ambiguity in the estimation of the motion and the reconstructed 3D scene. The global scale cannot be recovered unless information about the real world of the relation between the real world and the camera is established. Additionally, if the motion is estimated in a frame-to-frame basis considering only the image features, there is a scale ambiguity between the estimated translations. In the literature, this frame-to-frame scale ambiguity is solved by computing the motion of a third camera given the reconstructed structure from frames one and two. This technique, commonly refered as the PnP problem [3], is the common choice and there exist algorithms for estimating the full motion with as little as 3 points. Employing a PnP algorithm for estimating the motion of the third camera has some implications in terms of the error that is propagated. When using a PnP algorithm, the image error is first propagated to the reconstructed 3D structure, and then to the motion estimation. Also, this requires 3-frame matches for the full motion estimation.

FIT3D provides an alternative linear technique [4] with the same computational complexity but considerable improvement in terms of error propagation. This method computes the motion of the third camera given only image features between images two and three. This yields a scale free translation vector. Then the scale is computed with a linear function. With this method, the error in the rotation and translation direction is only propagated from the image features, avoiding the propagation on the 3D reconstruction. Only in the computation of the scale, the error is propagated through the 3D structure. FIT3D also provides an implementation of the P6P solution for computing the motion of the third camera.

1) EXAMPLE: calculate scale with linear: For this example, we have 3 estimated camera poses $P1$, $P2$ and $P3$ along with 3-frame image feature correspondences $X1$, $X2$ and $X3$. The first camera is at the origin ($P1 = [I|0]$) and the other to cameras were calculated on a frame-to-frame basis. We triangulate the image features

between cameras $P1$ and $P2$, resulting in the set of space points $X3D$. The scale of the third camera is computed linearly.

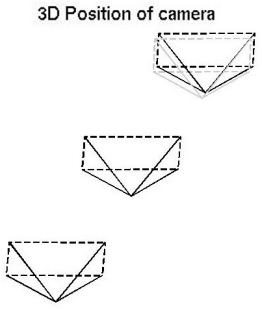

Fig. 6. Black: camera before scale adjustment. Grey: camera after scale adjustment.

```
>> [x3d] = findTriangulationLM(X1, X2, P1,
P2, K1, K2);
>> [scale,P3scaled] =
findScaleLinear(P3,x3d,X2,X3);
```

D. Iterative refinement

The process of estimating the motion between multiple cameras usually involves error propagation. In our setup, the error is propagated from the sensor of the camera, to the feature localization and matching and then to the motion estimation algorithm and scale adjustment. Therefore it is common to obtain a scale consistent motion estimate that is slightly deviated from the desired result. The process of optimizing the estimate is commonly known at Bundle Adjustment [13]. It involves the refinement of the motion estimate using a numerical iterative approach. This procedure is typically performed through the minimization of a cost function.

Depending on the scope of the refinement procedure, this can be characterized as either *local refinement* or *global refinement*. Local refinement aims at optimizing the motion estimate at a local level of only a few camera poses. Global refinement on the other hand aims at taking advantage of the complete set of constraints in order to optimize the complete set of camera poses. FIT3D provides method for both local and global refinement with two very distinct flavors.

Local Refinement

We understand the local refinement as the optimization of both camera motion estimate P and camera calibration matrix K using as little as two cameras. The idea is to refine the motion estimate locally as the number of parameters to refine is very small and the procedure can be performed very fast. This type of refinement is robust and can usually cope with certain degeneracies where the simple use of the 8-point algorithm will fail.

FIT3D provides the functionality to refine the motion estimate of a camera and its camera calibration matrix given a set of matching features and the previous camera motion estimate. For this local refinement procedure we employ the common technique of minimizing the reprojection error of the triangulated 3D structure.

1) EXAMPLE: refine locally: in this example we iteratively refine the second camera pose and internal parameters to minimize the reprojection error in both the first and second camera. P1 is assumed to be at $[I|0]$.

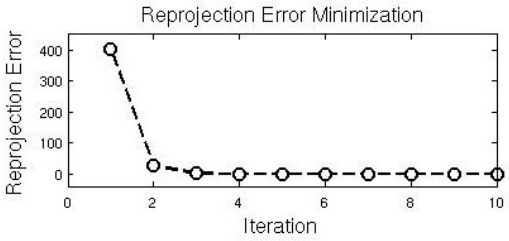

Fig. 7. Reprojection error during local refinement.

```
>> [P2refined,K2refined] =
bundleAdjust(P2,X1,X2,K2);
```

Global Refinement

For the global iterative refinement we employ a different approach than the standard Bundle Adjustment where the camera poses and/or the 3D structure is optimized minimizing the reprojeciton error. Instead, we take advantage of 3-frame feature matches to define a technique [4] that aims at optimizing both the camera poses and the 3D structure with a single error distance minimization.

Given that we can obtain 3-frame feature matches and we have obtained a scale consistent motion estimate, we can now obtain the triangulated 3D structure between two consecutive frames. For the sake of simplicity we only consider the 3 frame case now. Given 3 frames we can therefore obtain two different but corresponding sets of 3D points in space if we obtain the spatial triangulation of only the 3-frame feature matches.

As we know the point to point correspondence, we devise a cost function that minimizes the squared sum of the point-to-poin distance in space. This technique presents computational advantages than the standard Bundle Adjustment approach since only the parameters of the camera poses are explicitly optimized while the 3D structure is implicitly optimized.

2) EXAMPLE: global refinement: given the errors in the reconstructed structure, we apply our global refinement scheme based on the minimization of the distance between corresponding pointclouds.

Fig. 8. Top: Images used to obtain a scaled consistent motion. Bottom-Left: resulting 3D structure before refinement. Bottom-Right: resulting structure after refinement. Motion was altered to produce a noisy pointcloud.

```
>> [Psrefined] = bundleAdjustNFrames3D(
threeFrameMatches, Ks, Ps, iterations,
maxDistance);
```

V. 3D MODEL RECONSTRUCTION

Up to this point we have shown how to use FIT3D to obtain an accurate scale consistent motion estimation for a set of consecutive cameras. This however is not enough since we would also like to obtain a 3D representation of the depicted scene. For this we need to first obtain a set of 3D points in space that represent the key features in the images and finally obtain a set of primitives to represent those points.

A. 3D triangulation

Obtaining a set of 3D points from image feature matches and camera poses is not a trivial problem due to the error in image recording and the propagation of the error through the motion estimate and refinement. The straight forward method to obtain the triangulation of an image feature is tracing a ray from the camera centers to the features in the image plane and calculating the intersection in space. The error propagation however makes this method unrealistic since most likely the rays will not intersect. There are a number of solutions to this problem [7] from which we choose a linear method due to its simplicity and speed.

1) EXAMPLE: obtain 3D pointcloud: we use a linear triangulation method to obtain the set of 3D points corresponding to the selected image features (see Figure 8, bottom-left).

```
>> [P2refined,K2refined] =
findTriangulationLM(P2,X1,X2,K2);
```

B. Modeling

Having obtained a set of 3D points that represent the scene we need to simplify the set of points to a set of textured primitives.

Primitive fitting

The first step is to find a set of primitives to describe the point cloud. For this purpose we choose a robust RANSAC approach with a novel sampling technique. Having the complete set of 3D points, we sample the set and choose one seed point and the closest n points. Then, using RANSAC, we fit a plane through those points and reject the outliers. If a plane is found, then the points are discarded from the complete set and the process continues. The method stops once a certain number of planes are found or the set of points is exhausted.

1) EXAMPLE: fit planes: in this example we use the data set of the alley to find planar patches in the 3D point cloud (see Figure 9). Each planar patch is assigned a different color.

```
>> [planes] = fitPlanesRANSA( X3D,
minPoints, maxPlanes, ransacThreshold,
ransacIterations);
```

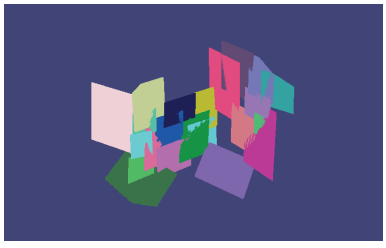

Fig. 9. Colored planar patches.

Texture generation

Having obtained a set of planes that describe the reconstructed scene, we need to apply a realistic texture and trim the borders of the planes.

For this purpose, we use the convex hull of the set of points to define the limit of the planar patch. For the texture generation, we project the texture of the image that recorded this part of the scene from the most orthogonal position. This method helps in avoiding visual shadows and occlusions.

2) EXAMPLE: apply texture: having obtained the set of planar patches, the most frontal view among the recorded images is used to reproject the texture (see Figure 10).

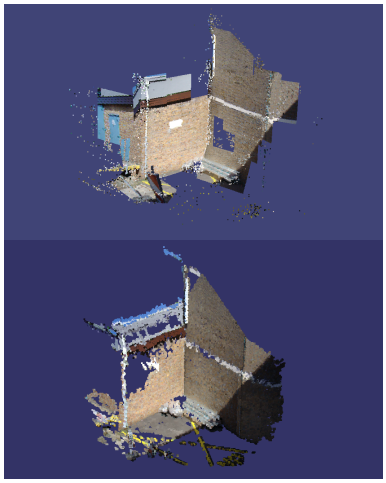

Fig. 10. Top: Textured planar patches. Bottom: colored point cloud output using PMVS2.

VI. Conclusion

In this paper we have presented some of the functionality of our FIT3D Toolbox for 3D reconstruction from a set of ordered images. We believe that FIT3D fills a gap in the set of research tools that will help other scientist to develop and apply their methods and techniques in a more global context. FIT3D allows them to obtain a full 3D model with little effort in the integration. As compared to other relevant tools, FIT3D presents a set of novel approaches for both robust monocular visual motion estimation and refinement by point cloud distance minimization. Even tough FIT3D provides functionality for global refinement, the robust motion estimation techniques employed are in most cases sufficient and do not require the global refinement step. Only for those more difficult data sets it needs to be applied. Also, FIT3D is provided in an educative format where special care has been taken in building Matlab functions that are easy to use and understand. Along with the toolbox a set of example scripts and data sets are provided. In the future we plan to extend the Toolbox with error analysis methods and eliminate the constraint of ordered images using topological mapping techniques.

Given the extent of FIT3D only a small portion of the functionality has been presented here. Additional functionality is provided for: mass feature extraction, feature matching, feature tracking, error measures, interfaces for PMVS2, motion sampling, simulators for ground truth generation and evaluation, and a long etcetera.

References

[1] J. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/.

[2] Frank Dellaert, Steven Seitz, Chuck Thorpe, and Sebastian Thrun. Structure from motion without correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00)*, June 2000.

[3] Daniel DeMenthon and Larry S. Davis. Exact and approximate solutions of the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(11):1100–1105, 1992.

[4] Isaac Esteban, Leo Dorst, and Judith Dijk. Closed form solution for the scale ambiguity problem in monocular visual odometry, 2010.

[5] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. pages 726–740, 1987.

[6] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[8] P. D. Kovesi. MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>.

[9] M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[10] D. Lowe. Distinctive image features from scale-invariant keypoints. In *Int. J. of Computer Vision*, 2004.

[11] D. Nister. An efficient solution to the five-point relative pose problem. In *CVPR*, 2003.

[12] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[13] Bill Triggs, Philip Mclauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – a modern synthesis, 2000.

[14] A. Vedaldi. An open implementation of the SIFT detector and descriptor. Technical Report 070012, UCLA CSD, 2007.
[15] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision*, 1999.