Accurate Camera Calibration from Multi-View Stereo and Bundle Adjustment

Yasutaka Furukawa[1]
Department of Computer Science
and Beckman Institute
University of Illinois at Urbana-Champaign, USA[1]

Jean Ponce[2,1]
Willow Team
LIENS (CNRS/ENS/INRIA UMR 8548)
Ecole Normale Supérieure, Paris, France[2]

Abstract: *The advent of high-resolution digital cameras and sophisticated multi-view stereo algorithms offers the promises of unprecedented geometric fidelity in image-based modeling tasks, but it also puts unprecedented demands on camera calibration to fulfill these promises. This paper presents a novel approach to camera calibration where top-down information from rough camera parameter estimates and the output of a publicly available multi-view-stereo system [6] on scaled-down input images are used to effectively guide the search for additional image correspondences and significantly improve camera calibration parameters using a standard bundle adjustment algorithm [14]. The proposed method has been tested on several real datasets—including objects without salient features for which image correspondences cannot be found in a purely bottom-up fashion, and image-based modeling tasks—including the construction of visual hulls where thin structures are lost without our calibration procedure.*

1. Introduction

Modern *multi-view stereovision* (*MVS*) systems are capable of capturing dense and accurate surface models of complex objects from a moderate number of calibrated images: Indeed, a recent study has shown that several algorithms achieve surface coverage of about 95% and depth accuracy of about 0.5*mm* for an object 10*cm* in diameter observed by 16 low-resolution (640×480) cameras. Combined with the emergence of affordable, high-resolution (12Mpixel) consumer-grade cameras, this technology promises even higher, unprecedented geometric fidelity in image-based modeling tasks, but puts tremendous demands on the calibration procedure used to estimate the intrinsic and extrinsic camera parameters, lens distortion coefficients, etc.

There are two main approaches to the calibration problem: The first one, dubbed *chart-based calibration* or *CBC* in the rest of this presentation, assumes that an object with precisely known geometry (the chart) is present in all input images, and computes the camera parameters consistent with a set of correspondences between the features defining the chart and their observed image projections [3, 23]. It is often used in conjunction with positioning systems such as a robot arm [18] or a turntable [10] that can repeat the same motion with high accuracy, so that object and calibration chart pictures can be taken separately but under the same viewing conditions. The second approach to calibration is *structure from motion* (*SFM*), where both the scene shape (structure) and the camera parameters (motion) consistent with a set of correspondences between scene and image features are estimated [9]. In this process, the *intrinsic* camera parameters are often supposed to be known a priori [16], or recovered a posteriori through *auto-calibration* [22]. A final *bundle adjustment* (*BA*) stage is then typically used to fine tune the positions of the scene points and the entire set of camera parameters (including the intrinsic ones and possibly the distortion coefficients) in a single non-linear optimization [14, 21]. A key ingredient of both approaches to calibration is the *selection of feature correspondences* (*SFC*), a procedure that may be manual or (partially or totally) automated, and is often intertwined with the calibration process: In a typical SFM system for example, features may first be found as "interest points" in all input images, before a robust matching technique such as *RANSAC* [4] is used to simultaneously estimate a set of consistent feature correspondences *and* camera parameters. Some approaches propose to improve feature correspondences for robust camera calibration [17, 15]. However, reliable automated SFC/SFM systems are hard to come by, and they may fail for scenes composed mostly of objects with weak textures (e.g., human faces). In this case, manual feature selection and/or CBC are the only viable alternatives.

Today, despite decades of work and a mature technology, putting together a complete and reliable calibration pipeline thus remains a non-trivial procedure requiring much know-how, with various pitfalls and sources of inaccuracy. Automated SFC/SFM methods tend to work well for close-by cameras in controlled environments—though errors tend to accumulate for long-range motions, and they may be ineffective for poorly textured scenes and widely separated input images. CBC systems can be used regardless of scene texture and view separation, but it is difficult to design and build accurate calibration charts with patterns clearly visible

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

from all views. This is particularly true for 3D charts (which are desirable for uniform accuracy over the visible field), but remains a problem even for printed planar grids (the plates the paper is laid on may not be quite flat, laser printers are surprisingly inaccurate, etc.). In addition, the robot arms or turntables used in many experimental setups may not be exactly repetitive. In fact, even a camera attached to a sturdy tripod may be affected during experiments by vibrations from the floor, thermal effects, etc. These seemingly minor factors may not be negligible for modern high-resolution cameras,[1] and they limit the effectiveness of classical chart-based calibration. Of course, sophisticated setups that are less sensitive to these difficulties have been developed by photogrammeters [24], but they typically require special equipment and software that are unfortunately not available in many academic and industrial settings. Our goal, on the other hand, is to provide a flexible but high-accuracy calibration system that is affordable and accessible to everyone. To this end, a few researchers have proposed using scene information to refine camera calibration parameters: Lavest *et al.* propose in [13] to compensate for the inaccuracy of a calibration chart by adjusting the 3D position of the markers that make it up, but this requires special markers and software for locating them with sufficient sub-pixel precision. The calibration algorithms proposed by Hernandez *et al.* [11] and Wong and Cipolla [25] exploit silhouette information instead. They work for objects without any texture and are effective in wide-baseline situations, but are limited to circular camera motions.

In this article, we propose a very simple and efficient BA algorithm that does not suffer from these limitations and exploits top-down information provided by a rough surface reconstruction to establish image correspondences. Concretely, given a set of input images, possibly inaccurate camera parameters that may have been obtained by an SFM or CBC system, and some conservative estimate of the corresponding reprojection errors, the input images are first scaled down so these errors become small enough to successfully run the patch-based multi-view stereo algorithm (PMVS) of Furukawa and Ponce [8] that reconstructs a set of oriented points (points plus normals) densely covering the surface of the observed scene, and identifies the images where they are visible. The core component of our approach is its second stage, where image features are matched across multiple views using the estimated surface geometry and visibility information. Finally, matched features are input to the SBA bundle adjustment software of Lourakis and Argyros [14] to tighten up camera parameters. The proposed method has been tested on several real datasets—including objects without salient features for which image correspondences cannot be found in a purely bottom-up fashion, and image-based modeling tasks—including the construction of visual hulls where thin structures are lost without our bundle adjustment procedure (Section 4). Note that PMVS [6], SBA [14], and several CBC systems such as [3] are publicly available. Bundled with our software, which is also available online at http://www-cvr.ai.uiuc.edu/∼yfurukaw/research/pba, they make a complete software suite for high-accuracy camera calibration.

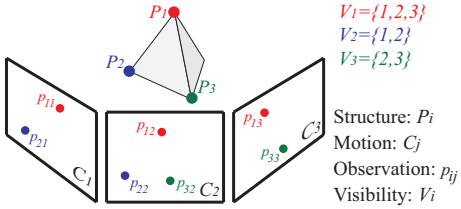


Figure 1. Notation: Three points P_1, P_2, P_3 are observed by three cameras C_1, C_2, C_3.

2. Imaging Model and Preliminaries

Our approach to camera calibration accommodates in principle any parametric projection model of the form $p = f(P,C)$, where P denotes both a scene point and its position in some fixed world coordinate system, C denotes both an image and the corresponding vector of camera parameters, and p denotes the projection of P in the image. In practice, our implementation is currently limited to a standard perspective projection model where C records five intrinsic parameters and six extrinsic ones. Distortion is thus supposed to be negligible, or already corrected, for example by software such as DxO Optics Pro [1].[2] Standard BA algorithms take the following three data as inputs: a set of n 3D point positions $P_1, P_2, \cdots, P_n$, m camera parameters $C_1, \ldots, C_m$, and the positions of the projections p_{ij} of the points P_i in the images C_j where they are visible (Fig. 1). They optimize both the scene P_i and camera parameters C_j by minimizing the sum of squared reprojection errors:

$$\sum_{i=1}^{n} \sum_{j \in V_i} (p_{ij} - f(P_i, C_j))^2, \qquad (1)$$

where V_i encodes visibility information as the set of indices of images where P_i is visible. Unlike BA algorithms, multi-

[1] For example, the robot arm (Stanford spherical gantry) used in the multi-view stereo evaluation of [18] has an accuracy of 0.01° for a 1m radius sphere observing an object about $15cm$ in diameter, which yields approximately $1.0[m] \times 0.01 \times \pi/180 = 0.175[mm]$ errors near an object. Even with the low-resolution 640×480 cameras used in [18], where a pixel covers roughly $0.25mm$ on the surface of an object, this error corresponds to $0.175/0.25 = 0.7$pixels, which is not negligible. If one used a high-resolution 4000×3000 camera, the positioning error would increase to $0.7 \times 4000/640 = 4.4$pixels.

[2] We will address this limitation in the near future. In the mean time, we believe that the experiments of Section 4 demonstrate the effectiveness of our current implementation in practical settings.

Input: Cameras parameters $\{K_j, R_j, t_j\}$ and expected reprojection error E_r.
Output: Refined cameras parameters $\{K_j, R_j, t_j\}$.

Build image pyramids for all the images.
Compute a level L to run PMVS: $L \leftarrow \max(0, \lfloor \log_2 E_r \rfloor)$.
Repeat four times
- Run PMVS on level L of the pyramids to obtain patches $\{P_i\}$ and their visibility information $\{V_i\}$.
- Initialize feature locations: $p_{ij} \leftarrow F(P_i, K_j, R_j, t_j)$.
- Sub-sample feature correspondences.
- *For* each feature correspondence $\{p_{ij} | j \in V_i\}$
  - Identify a *reference* camera C_{j_0} in V_i with the minimum foreshortening factor.
  - *For* each non-reference feature $p_{ij} (j \in V_i, j \neq j_0)$
    - *For* $L^* \leftarrow L$ down to 0
      - Use level L^* of image pyramids to refine p_{ij}: $p_{ij} \leftarrow argmax_{p_{ij}} \mathrm{NCC}(q_{ij}, q_{ij_0})$.
  - Filter out features that have moved too much.
- Refine $\{P_i, K_j, R_j, t_j\}$ by a standard BA with $\{p_{ij}\}$.
- Update E_r by the *mean* and *std* of reprojection errors.

Figure 2. Overall algorithm.

view stereo algorithms are aimed at recovering scene information alone given fixed camera parameters. In our implementation, we use the PMVS software [6, 8] that generates a set of *oriented* points P_i, together with the corresponding visibility information V_i. We have chosen PMVS because (1) it is one of the best MVS algorithms to date according to the Middlebury benchmarks [8, 18], (2) our method does not require a 3D mesh model but just a set of oriented points, which is the output of PMVS, and (3) –as noted earlier– PMVS is freely available [6]. This is also one of the reasons for choosing the SBA software [14] for bundle adjustment, the others being its flexibility and efficiency.

3. Algorithm

The overall algorithm is given in Fig. 2. We use the oriented points P_i $(i = 1, \ldots, n)$ and the corresponding visibility information V_i output by PMVS to form initial image correspondences p_{ij}. The parameters p_{ij} and V_i are then refined by our algorithm. Given these, it is possible to rely on SBA to improve the camera parameters. We focus in this section on how to initialize and refine feature correspondences.

3.1. Initializing Feature Correspondences

In practice, we have found PMVS to be robust to errors in camera parameters *as long as the image resolution matches the corresponding reprojection errors*—that is, when features to be matched are roughly within two pixels of the corresponding 3D points. Given an initial set of camera parameters, it is usually possible to obtain a conservative estimate of the expected reprojection error E_r by hand (e.g., by visually inspecting a number of epipolar lines) or automatically (e.g., by directly measuring reprojection errors associated with the features matched by an SFM system). Thus, we first build image pyramids for all the input images, then run PMVS on the largest pyramid level L smaller than $\log_2 E_r$. At this level, images are 2^L times smaller than the originals, with reprojection errors of at most about two pixels, and we run PMVS. We then project the points P_i output by this program into the images where they are visible to obtain an initial set of image correspondences $p_{ij} = f(P_i, C_j)$, with j in V_i. Depending on the value of L and the choice of the PMVS parameter ζ that controls the density of oriented points it constructs, the number of these points, and thus, the number of feature correspondences may become quite large. Dense reconstruction is not necessary for bundle adjustment, and we sub-sample feature correspondences for efficiency.[3] More concretely, we first divide each image into 10×10 uniform blocks, and randomly select within each block at most ε features. A feature correspondence will be used in the next refinement step if at least one of its associated image features p_{ij} was sampled in the above procedure. In practice, ε is chosen so that the number of feature correspondences is approximately one fifth of the original after this sampling step. Note that sub-sampling is performed in each block (as opposed to each image) in order to ensure uniformly distributed feature correspondences.

3.2. Refining Feature Correspondences

Due to errors in camera parameters and the use of low-resolution images in PMVS, the initial values of p_{ij} are not accurate. Therefore, the second step of the algorithm is to optimize the feature locations p_{ij} by comparing local image textures. Concretely, since we have an estimate of the surface normal at each point P_i, we consider a small 3D rectangular patch Q_i centered at P_i and construct its projection q_{ij} in the set V_i of images where P_i is visible. We automatically determine the extent of Q_i so its largest projection covers an image area of about $\delta \times \delta$ pixels (Fig. 3, we have used $\delta = 7$ throughout our experiments). In practice, as in [8], a patch Q_i is represented by a $\delta \times \delta$ grid of 3D points and the local image texture inside q_{ij} is, in turn, represented by a set of pixel colors at their image projections.

Next, our problem is to refine feature locations by matching local image textures q_{ij} (for efficiency, we fix the shapes of the image patches q_{ij} and only allow the positions of their centers to change). Let us call the camera with the minimum foreshortening factor with respect to P_i the *reference* camera of P_i, and use j_0 to denote its index. We fix the lo-

[3] We could increase the value of ζ to obtain a sparser set of patches without sub-sampling, but, as detailed in [8], a dense reconstruction is necessary for this algorithm to work well and determine visibility information accurately.

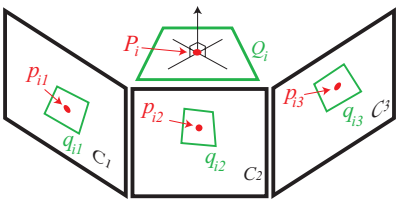


Figure 3. Given a patch (P_i, Q_i) and the visibility information V_i, we initialize matching images patches (p_{ij}, q_{ij}).

vase	*dino*	*face*	*spiderman*	*predator*
21 images	16 images	13 images	16 images	24 images
3*MP*	0.3*MP*	1.5*MP*	1*MP*	2*MP*

Figure 4. Top: Sample pictures for the five datasets used in the experiments. Bottom: The number of images and their approximate resolution (in megapixels) for each dataset.

cation p_{ij_0} in the reference camera and optimize every other element $p_{ij}, j \neq j_0$ one by one by maximizing the consistency between q_{ij_0} and q_{ij} in a multi-scale fashion. More concretely, starting from the level L of the image pyramids where PMVS was used, the conjugate gradient method is used to optimize p_{ij} by maximizing the normalized cross correlation between q_{ij_0} and q_{ij}. The process is iterated after convergence at the next lower level. After the optimization has been over at the bottom level, we check whether p_{ij} has not moved too much during the optimization. In particular, if p_{ij} has moved more than E_r pixels from its original location, it is removed as an outlier and V_i is updated accordingly. Having refined feature correspondences, we then use the SBA bundle adjustment software [14] to update the camera parameters. In practice, we repeat the whole procedure (PMVS, multi-view feature matching, and SBA) four times to tighten up the camera calibration, while E_r is updated to be the mean plus three times the standard deviation of reprojection errors computed in the last step. Note that L is fixed across iterations instead of recomputed from E_r for efficiency, since PMVS runs slowly with a small value of L.

4. Experimental Results and Discussions

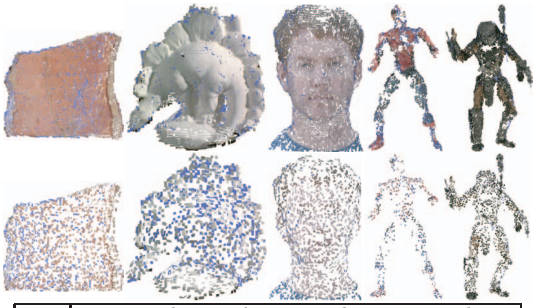
4.1. Datasets

The proposed algorithm has been implemented in C++ and tested on five real datasets, with sample input images shown in Fig. 4, along with the number of images and their (approximate) resolution. The *vase* dataset has been calibrated by a local implementation of a standard automated SFC/SFM/BA suite as described in [9]. This software has failed on all other datasets except for *predator*, for which 14 out of the 24 images have been calibrated successfully. It is of course possible that a different implementation would have given better results, but we believe that this is rather typical of practical situations when different views are widely separated and/or textures are not prominent, and this is a good setting to exercise our algorithm. The *spiderman* dataset has been calibrated using a planar checkerboard pattern and a turntable with the calibration software from [3], and the same setup has been used to obtain a second set of camera parameters for the *predator* dataset. The *face* dataset was acquired outdoors, without a calibration chart, and textures are too weak for typical automated SFC/SFM algorithms to work. This is a typical case where, in post-production environments for example, feature correspondences would be manually inserted to calibrate cameras. This is what we have actually done for this dataset. The *dino* dataset is part of the Middlebury MVS evaluation project, and it has been carefully calibrated by the authors of [18]. Nonetheless, this is a very interesting object lacking in salient features and a good example to test our algorithm. Therefore, we have artificially added Gaussian noise to the camera parameters so that reprojection errors become approximately six pixels, yielding a challenging dataset.

Probably due to the use of a rather inaccurate planar calibration board, and a turntable that may not be exactly repetitive, careful visual inspection reveals that *spiderman* and *predator* contain some errors, in particular, for points far away from the turntable where the calibration board was placed. The calibration of *face* is not tight either, because of the sparse manual feature correspondences (at most a few dozens among close-by views) used to calibrate the cameras. The *vase* dataset has relatively small reprojection errors with many close-by images for which SFM algorithms work well, but some images contain large errors because of the use of a flash and the limited depth of field, and errors do accumulate. Note that since silhouette information is used both by the PMVS software and the visual hull computations described in the next section, object silhouettes have been manually extracted using PhotoShop for all datasets except *dino*, where background pixels are close to black, and thresholding followed by morphological operations is sufficient to obtain the silhouettes. Note that the silhouette extraction is not essential for our algorithm, although it helps the system to run and converge more quickly. Furthermore, the use of PMVS is not essential either and can be replaced by any another multi-view stereo system.

4.2. Experiments

The two main parameters of PMVS are a correlation window size γ, and a parameter ζ controlling the density of the reconstruction: PMVS tries to reconstruct at least one patch in every $\zeta \times \zeta$ image window. We use $\gamma = 7$ or 9 and $\zeta = 2$ or 4 in all our experiments. Figure 5 shows for



	vase	*dino*	*face*	*spiderman*	*predator*
E_r	12	7	8	7	7
L	3	2	3	2	2
N_p	9926	5912	7347	3344	12760
N_t	1310	1763	1997	840	3587

Figure 5. Top: Patches reconstructed by PMVS at level L of the pyramid. Center: Subsets of these patches that have successfully generated feature correspondences after sub-sampling. Bottom: Statistics of the matching process.

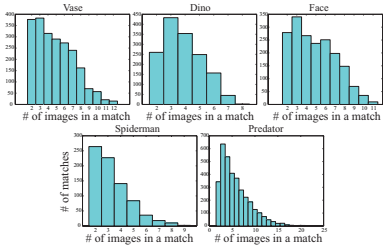


Figure 7. Histograms of the number of images in which features are matched.

each dataset a set of patches reconstructed by PMVS (top row), and its subset that have successfully generated feature correspondences after sub-sampling (second row). Figure 5 (bottom row) gives some statistics on the matching procedure. There, as usual, E_r denotes a conservative estimate of the expected reprojection errors in pixels, and L denotes the level of image pyramids used by PMVS to reconstruct a set of patches. The number of patches reconstructed by PMVS is denoted by N_p, and the number of patches that successfully generated feature correspondences after sub-sampling is denoted by N_t. Examples of matched 2D features for each dataset are shown in Fig. 6. By taking into account the surface orientation and the visibility information estimated by PMVS, the proposed method has been able to match features in many views taken from quite different angles even when image textures are very weak. In the examples shown in Fig. 6, the numbers of images where features are matched varies from 6 to 17. Histograms are given in Fig. 7.

It is impossible to give a full quantitative evaluation of our results given the lack of ground truth 3D data. We can, however, demonstrate that our camera calibration procedure does its job as far as improving the reprojection errors of the patches associated with the established feature correspondences is concerned. Figure 8 shows the mean and standard deviation of these reprojection errors at each iteration of our algorithm for every dataset. The right-most graph shows the number of 2D features matched and used to refine camera parameters. Note that six, instead of four, iterations have been performed to create the figure (the two extra iterations show a decrease in error but do not seem to affect the quality of our reconstructions in practice). The mean reprojection error decreases from 2-3pixels before refinement to about 0.25 to 0.5 pixels for most datasets after six iterations. Incorporating radial distortion in the parameters refined by our algorithm might allow us to go even further. As noted earlier, we plan to do so in the near future.

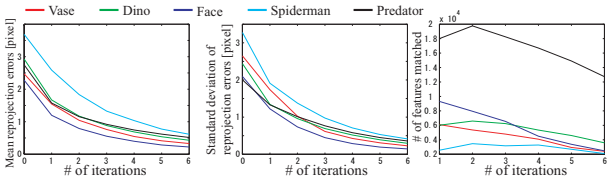


Figure 8. The mean and standard deviation of reprojection errors in pixel for each dataset at each iteration. The right-most graph shows the total number of matched 2D features per iteration.

We have used a couple of different methods to qualitatively assess the accuracy of the estimated camera parameters. First, epipolar geometry has been used to check the consistency between pairs of images (Fig. 9). More concretely, for a pair of images, we draw pairs of epipolar lines in different colors to see if corresponding epipolar lines of the same color pass through the same feature points in the two images. Several images in the *vase* dataset contained large errors before refinement (approximately six pixels in some places) because of the limited depth of field and an exposure difference due to the use of a flash. The *spiderman* and *predator* datasets also contain very large errors, up to seven (or possibly more) pixels for points far from the ground plane where the calibration chart is located. In each case, the proposed method has been able to refine camera parameters to sub-pixel level precision. Inconsistencies in the *dino* dataset introduced by the added noise have also been corrected by our method despite its weak texture.

Next, we have tested the ability of our algorithm to recover camera parameters that are highly consistent across widely separated views. We use the *spiderman* and *predator* datasets in this experiment (Fig. 10) since parts of these objects are as thin as a few pixels in many images. Recovering such intricate structures normally requires exploiting silhouette information in the form of a visual hull [2] or a hybrid model combining silhouette and texture information [7, 10, 19, 20]. In turn, this requires a high degree of geometric consistency over the cameras, and provides a good testing ground for our algorithm. We have used the EPVH software of Franco and Boyer [5] to construct polye-

Figure 6. A set of matching 2D features is shown for each dataset. The proposed method is able to match features in many images even without salient textures due to the use of surface geometry and visibility information estimated by the multi-view stereo algorithm.

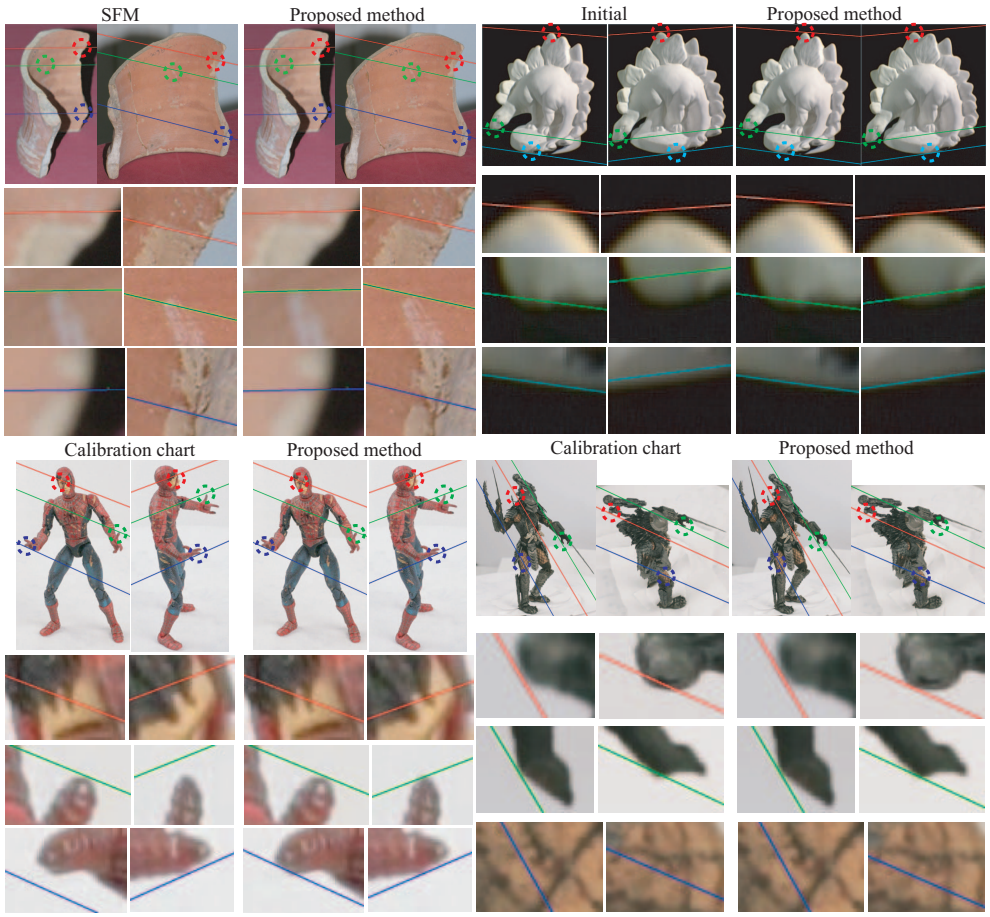


Figure 9. Epipolar lines are used to assess the improvements in camera parameters. A pair of epipolar lines of the same color must pass through same feature points.

dral visual hulls in our experiments, and Fig. 10 shows that thin, intricate details such as the fingers of *spiderman* and the blades of *predator* are successfully recovered with refined camera parameters, and competely lost otherwise.

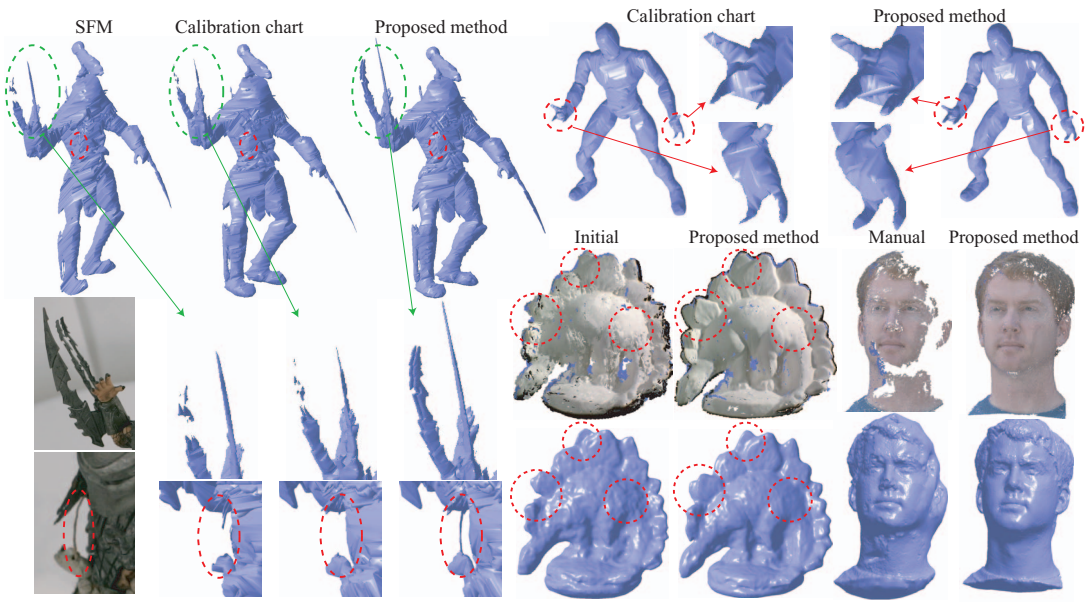


Figure 10. Visual hull models are used to assess the accuracy of camera parameters for *spiderman* and *predator*. Intricate structures are reconstructed only from the camera parameters refined by the proposed method. For *dino* and *face*, a set of patches reconstructed by PMVS and a 3D mesh model extracted from these patches are used for the assessment. See text for more details.

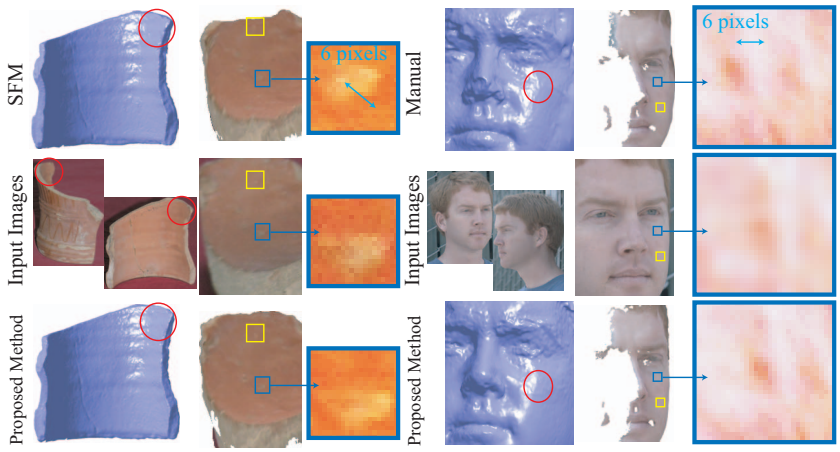


Figure 11. Inconsistencies in widely separated cameras (accumulation errors) are often not recognizable from 3D mesh models reconstructed by a MVS algorithm. For further assessments, we pick a pair of separated cameras shown in the middle row, texture-map the surface from the right image, render it to the left, and compare the rendered model with the left image. The rendered and the input images look the same only if camera parameters and the reconstructed model are accurate. The top and the bottom rows show rendered images and the reconstructed 3D mesh model before and after the refinement, respectively. The amount of errors with the initial camera parameters (calibrated by SFM for *vase* and manual feature correspondences for *face*) is roughly six pixels for both datasets, which are very large.

For *dino* and *face*, we have used PMVS to reconstruct a set of patches that are then converted into a 3D mesh model using the method described in [12] (Fig. 10, bottom right). The large artifacts at the neck and the chin of the shaded *face* reconstruction before refinement are mainly side effects of the use of visual hull constraints in PMVS (patches are not reconstructed outside the visual hull [7]), exacerbated by the fact that the meshing method of [12] extrapolates the surface in areas where data is not present. Ignoring these artifacts, the difference in quality between the reconstructions before and after refinement is still obvious in Fig. 10, near the fins of the dinosaur, or the nose and mouth of the face for example. In general, however, the accumulation of errors due to geometric inconsistencies among widely separated cameras is not always visually recognizable in 3D models reconstructed by multi-view stereo,

Table 1. Running time in minutes of the three steps of the proposed algorithm for the first iteration.

	vase	*dino*	*face*	*spiderman*	*predator*
PMVS	1.9	0.40	0.65	0.34	1.9
Match	1.1	0.66	0.96	0.24	1.6
BA	0.17	0.13	0.17	0.03	0.38

because detailed local reconstructions can be obtained from a set of close cameras, and wide-baseline inconsistencies turn out as low-frequency errors. In order to assess the effectiveness of our algorithm in handling this issue, we pick a pair of widely separated cameras C_1 and C_2, map a texture from one camera C_1 onto the reconstructed model, render it as seen from C_2, and compare the rendered model with the input image associated with C_2. The two images should look the same (besides exposure differences) when the camera parameters and the 3D model are accurate. Figure 11 illustrates this on the *vase* and *face* datasets: Mesh models obtained again by combining PMVS [8] and the surface extraction algorithm of [12] are shown for both the initial and refined camera parameters. Although the reconstructed *vase* models do not look very different, the amount of *drifting* between rendered and input images is approximately six pixels for initial camera parameters. Similarly, for the *face* model, the reconstructed surfaces at the left cheek just beside the nose look detailed and similar to each other, while the rendered image is off by approximately six pixels as well. In both cases, the error decreases to sub-pixel levels after refinement. Note that reducing low-frequency errors may not necessarily improve the appearance of 3D models, but is essential in obtaining *accuracy* in applications where the actual model geometry, and not just their overall appearance, is important (e.g., engineering data analysis or high-fidelity surface modeling in the game and movie industries).

Finally, the running time in minutes per iteration of the three steps (PMVS, feature matching, bundle adjustment) of the proposed algorithm on a Dual Xeon 3.2GHz PC is given in Table. 1. As shown by the table, the proposed algorithm is efficient and takes at most a few minutes to refine camera parameters per iteration.

Acknowledgments: This paper was supported in part by the National Science Foundation under grant IIS-0535152, the INRIA associated team Thetys, and the Agence Nationale de la Recherch under grants Hfimbr and Triangles. We thank S. Sullivan, A. Suter, and Industrial Light and Magic for the face data set and support of this work. We also thank Jerome Courchay for the vase data set and SfM software.

References

[1] DxO Labs. DxO Optics Pro (http://www.dxo.com).

[2] B. Baumgart. *Geometric modeling for computer vision*. PhD thesis, Stanford University, 1974.

[3] J.-Y. Bouguet. Camera calibration toolbox for matlab.

[4] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6), 1981.

[5] J.-B. Franco and E. Boyer. Exact polyhedral visual hulls. In *BMVC*, 2003.

[6] Y. Furukawa and J. Ponce. PMVS (http://www-cvr.ai.uiuc.edu/∼yfurukaw/research/pmvs).

[7] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. In *ECCV*, pages 564–577, 2006.

[8] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, 2007.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[10] C. Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *CVIU*, 96(3), 2004.

[11] C. Hernández Esteban, F. Schmitt, and R. Cipolla. Silhouette coherence for camera calibration under circular motion. *PAMI*, 29, 2007.

[12] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symp. Geom. Proc.*, 2006.

[13] J.-M. Lavest, M. Viala, and M. Dhome. Do we really need an accurate calibration pattern to achieve a reliable camera calibration? In *ECCV*, 1998.

[14] M. Lourakis and A. Argyros. SBA: A generic sparse bundle adjustment C/C++ package based on the Levenberg-Marquardt algorithm.

[15] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. pages 1–8, 2007.

[16] D. Nister. An efficient solution to the five-point relative pose problem. *PAMI*, 26(6), 2004.

[17] L. Quan. Invariants of six points and projective reconstruction from three uncalibrated images. *PAMI*, 17(1), 1995.

[18] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *CVPR*, 2006.

[19] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *ICCV*, 2005.

[20] S. Tran and L. Davis. 3d surface reconstruction using graph cuts with surface constraints. In *ECCV*, 2006.

[21] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. *Bundle Ajustment — A Modern Synthesis*. Vision Algorithms: Theory and Practice. 2000.

[22] W. Triggs. Auto-calibration and the absolute quadric. In *CVPR*, June 1997.

[23] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras. *J. Robot. Autom.*, 3(4), 1987.

[24] V. Uffenkamp. State of the art of high precision industrial photogrammetry. In *Third International Workshop on Accelerator Alignment*, Annecy, France, 1993.

[25] K. K. Wong and R. Cipolla. Reconstruction of sculpture from its profiles with unknown camera positions. *IEEE Trans. Im. Proc.*, 2004.