Self Calibration Without Minimization

Russell A. Manning   Charles R. Dyer

Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin 53706

Abstract

Camera calibration is a necessary step in performing 3D scene reconstruction from photographic images. In this paper we present a new algorithm for the metric self calibration of a general pinhole camera that does not require the global minimization of an objective function and can produce all legal solutions to the three-camera self-calibration problem in a single pass. In contrast, virtually all previous self-calibration algorithms rely on nonlinear global optimization unless special assumptions are made about the camera or its motion. The key drawbacks to global-optimization-based methods is that, for nontrivial error functions and noisy data, they have no clear stopping condition and require that the global optimum have a reasonably large attraction basin.

1 Introduction

A very common approach to problem solving in machine vision is to first create an objective function whose global minimum corresponds to a desired solution and then apply standard optimization techniques (such as Levenberg-Marquardt from random starting positions) to find the global minimum. There are significant problems with this paradigm. First, when the attraction basin for the global minimum is too small it can be difficult or impossible to locate; this is particularly true for objective functions with high dimensionality. Second, generic optimization algorithms only find local minima; in general, it is impossible to determine when a true global minimum has been located and thus it is impossible to know when to stop the search process.

In this paper we present a solution to the "camera self calibration" problem that does not require the global optimization of an objective function. A camera's *calibration* is the function that maps 3D world coordinates onto the 2D imaging plane of the camera. Camera *self calibration* is the process of determining the calibration function directly from views captured by the camera without any special knowledge of the scene (e.g., without using a pre-measured calibration target). Determining camera calibration is a necessary step in 3D scene reconstruction from photographs.

By avoiding global optimization our algorithm for self calibration avoids the problems listed above. First, there is no question of whether a local or global optimum has been found: the algorithm runs in a single pass and returns the solution. Second, the issue of a solution hiding in a small attraction basin is eliminated, although a related but much less severe problem replaces it as discussed in Section 2.

The specific calibration method given in this paper is for cameras that can be approximated as pinhole cameras and assumes unchanging internal parameters. Almost all previous self-calibration algorithms for general pinhole cameras have utilized the global minimization of a nontrivial error function to search for calibration. The trend began with the very first self-calibration algorithm [2], which used the Kruppa constraints as the basis for its error function. Many other researchers have followed this approach: Hartley [5], Heyden and Astrom [7], Triggs [12], Pollefeys and Van Gool [11, 10], Lourakis and Deriche [8], and Fitzgibbon and Zisserman [4] (in the context of autocalibration from multiple moving objects) to name just a few. Note that we are not considering methods (such as that of Hartley et al. [6]) that place restrictions on internal parameters or camera motions, many of which involve linear or quasi-linear objective functions; we are assuming fully-unconstrained pinhole cameras. It should also be noted that the method of [6] does not avoid the problems outlined at the beginning of this section: it cannot determine whether a local or global minimum to the objective function has been found and it does not avoid the problem of small attraction basins, although it does increase the likelihood of locating a small attraction basin by reducing the size of the overall search space.

2 Optimization and its limitations

Much of the field of artificial intelligence is devoted to search techniques, and one of the most common

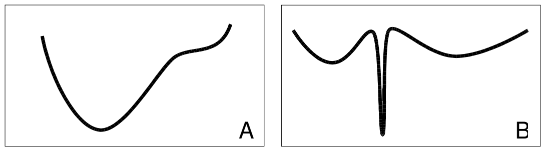

Figure 1: Optimization examples.

search techniques used in machine vision is the nonlinear optimization of a continuous objective function. This has the great benefit of standardization by making the search phase of an algorithm immediately understandable, and more importantly it allows machine vision algorithms to draw upon techniques from the entire field of numerical optimization, such as the often used Levenberg-Marquardt algorithm. However, it is well-recognized that simply phrasing a problem in terms of global optimization is not a panacea.

Consider finding the global minimums of the functions in Fig. 1(a) and Fig. 1(b). For functions like that in (a), this is a simple task for any nonlinear optimizer; at the very least, following local gradients will lead straight to the global minimum. The task can be much harder for functions like that in (b). In this case, a nonlinear minimizer is much more likely to get stuck in the two meaningless local minimums than to locate the "hidden valley" containing the global minimum.

For a one-dimensional function over a finite domain, the latter problem may still seem easy to overcome. The standard approach (because it is simple to implement) is to choose a random starting location, apply a nonlinear minimizer until a local minimum is reached, and then repeat this process until a "good" minimum is found (since it is usually impossible to know a priori what the global minimum should look like). The number of times this process must be repeated is related to the size of the "attraction basin" for the global minimum relative to the size of the search space. The *attraction basin* of a local minimum $\mathbf{x}$ is simply the set of all points from which the chosen nonlinear minimizer will converge to $\mathbf{x}$. Unfortunately, in many real problems the attraction basin of the global minimum can be very small relative to the search space size. This situation is often much worse for higher-dimensional search spaces where the likelihood of success depends on the volume of the attraction basin versus the volume of the overall search space.

Another troubling problem arises when the error function under consideration has more than one global minimum. While it is certainly possible to repeat the nonlinear optimization process until all global minimums have been located, this process could go on indefinitely depending on how small some of the attraction basins are. Furthermore, it may not be known a priori how many global minimums are supposed to exist, so that the algorithm must stop arbitrarily after "enough" minimums have been found.

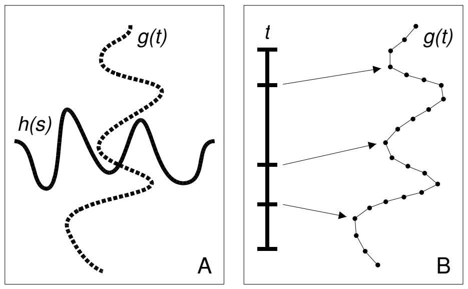

Figure 2: Curve-intersection problem.

Now consider the task illustrated in Fig. 2(a). This figure portrays two parametric curves $g(t)$ and $h(s)$, each of a single variable. The task is to determine where the curves intersect, and there may be an arbitrary number of intersection points. Because the curves are one-dimensional and exist in a two-dimensional search space, we will assume the mutual-intersection points are discrete and not consider the possibility of overlapping curve segments.

If we were to use a standard error-minimization approach, we might define a two-dimensional error function $f(s,t) = |g(t) - h(s)|$ representing the distance between points on the two curves. Then $f(s,t) \geq 0$ everywhere and clearly $f(s,t) = 0$ iff $g(t)$ and $h(s)$ represent the same position in the search space and the curves intersect. Unfortunately, due to numerous "near misses" between the two curves this error function has a shape like the function in Fig. 1(b) and applying nonlinear optimization methods to find the global minimums of f has the problems discussed earlier. It is also completely unknown a priori how many global minima f has, meaning the search process has no clear stopping point.

A better approach than using optimization for this problem is to simply "sketch" the curves g and h in the search space and then find the intersection points of the sketches directly. The intuition is that, when one looks at the illustration in Fig. 2(a) the mutual intersection point "pops out" instantly. A "sketch" (i.e., approximate surface) for each curve could be created by sampling the underlying parameters of g and h (i.e., the parameters s and t) at regular intervals and then connecting the images of successive samples using line segments, as illustrated in Fig. 2(b). The mutual-intersection points of g and h could then be found to a close approximation (depending on the sample rate of the underlying parameters) by intersecting the two piecewise-linear approximate surfaces. This process has the added benefit of determining all mutual-intersection points (again, depending on the granularity of the approximate surfaces). If desired, the approximate intersection points could be refined further by using nonlinear minimization on f; in this sense, the surface-fitting approach serves to locate attraction basins for the global minima of f.

3 The search for calibration

The self-calibration method used in this paper is of a type called "stratified self calibration." An overview of how this class of techniques works can be found in many sources, such as [11] and [1]. Because our focus in this paper is on the *method* of solution, we will describe the problem as the generic search for a point labeled $\hat{\mathbf{a}}$. Briefly, our stratified self calibration algorithm works as follows: First, a series of views is captured by a camera with fixed internal parameters. The views are assumed to have different optical centers and orientations. Next, a common projective basis is determined for all views. This can be complished using fundamental matrices and corresponding points between views. Third, "affine" calibration is determined by locating a point $\hat{\mathbf{a}}$ in a particular search space; it is this search process that our algorithm performs without minimizing a nonlinear objective function. Lastly, once $\hat{\mathbf{a}}$ is found the cameras can be placed in the same "metric" basis, meaning the camera views are now calibrated up to an overall scale factor.

Our method for finding $\hat{\mathbf{a}}$ works by locating the mutual intersection point of a series of screw-transform manifolds [9]. A "screw-transform manifold" is a surface in the calibration search space containing all legal calibrations corresponding to a given fundamental matrix. Because $\hat{\mathbf{a}}$ represents a legal calibration for all fundamental matrices arising from a given camera, it will be contained in *every* screw-transform manifold. Thus we can locate $\hat{\mathbf{a}}$ by finding the mutual intersection point(s) of a sufficient number of manifolds. In the case of stratified self calibration, the search space for $\hat{\mathbf{a}}$ is 3-dimensional and each screw-transform manifold is 2-dimensional; thus three or more manifolds are needed. Since a fundamental matrix exists for each pair of views, 3 views are necessary at a minimum.

In general, screw-transform manifolds are continuous and gently curving; they can be approximated well by using a series of linear patches. A visualization of several intersecting manifolds is given in Fig. 3. Note the use of triangular patches to represent the surfaces, and consider the analogy given in Fig. 2(b) where a 1-dimensional manifold is represented by line segments. In the example of Fig. 3, there is a single mutual intersection point which was located using the SURFIT algorithm described in Section 4. Note that once this point is found by the algorithm there is no question that it is a solution: it is a point physically lying on each manifold (as approximated by the triangular patches). Thus our approach ultimately either finds a legal calibration or knows that it has failed. Contrast this with generic optimization algorithms that can only determine if they have found a local minimum, which may or may not be the desired global optimum.

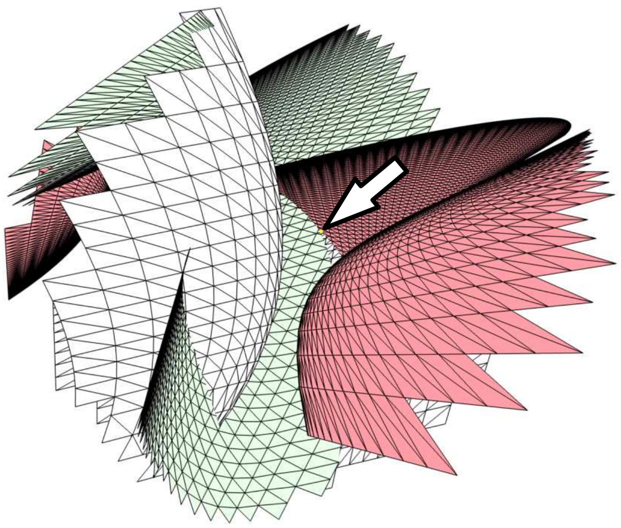

Figure 3: The mutual intersection point $\hat{\mathbf{a}}$ of 3 screw-transform manifolds. Each manifold is represented by a different color and $\hat{\mathbf{a}}$ is indicated by the arrow.

4 The surface-fitting (SURFIT) algorithm for manifold intersection

In the previous section we discussed how self calibration from monocular image sequences can be accomplished by finding the mutual intersection point of a series of manifolds. We now present a fast, reliable algorithm for determining such mutual intersection points. The method can be applied to general manifold-intersection problems, although we will concentrate on its specific application to self calibration.

Our algorithm is conceptually simple: Create a piecewise-linear approximation to each manifold and then find the mutual intersection points of the approximate surfaces directly. We term this the *surface-fitting* (SURFIT) algorithm. This approach is possible because the screw-transform manifold corresponding to fundamental matrix $\mathbf{F}$ is parameterized by a function $\Psi_{\mathbf{F}}$: for every 2 real numbers κ and θ, $\Psi_{\mathbf{F}}(\kappa, \theta)$ is a position on the manifold for $\mathbf{F}$. The function $\Psi_{\mathbf{F}}$ induces a "grid" coordinate system on the manifold. A specific definition of $\Psi_{\mathbf{F}}$ is given in [9], but for the purposes of this paper it is only necessary to understand that $\Psi_{\mathbf{F}}$

is a function that parameterizes the manifold.

The first step in creating a piecewise-linear approximation of $\Psi_{\mathbf{F}}$ is to think of the parameters κ and θ as lying in a restricted range like $[0, 1]$. In Fig. 4, `Kappa` and `Theta` denote the functions that convert the parameters from the range $[0, 1]$ to a range suitable for use with $\Psi_{\mathbf{F}}$. For example, `Theta` could represent scaling by 2π since θ is an angle. Once the parameters have been bounded, the next step is to create a grid of vertices on the manifold surface that samples the restricted range; psuedocode for this task is given in Fig. 4. The end result is a collection of triangles (2 for each grid square) that form a piecewise-linear approximation to the surface $\Psi_{\mathbf{F}}(\Re^2)$.

```
/* Input: fundamental matrix f, integer sampling rate
m; Output: piecewise-linear surface approximating
screw-transform manifold corresponding to f */
FUNCTION CreateSurface(f, m)
1   grid is an array[0..m][0..m] of 3-vectors
2   for i := 0 to m
3      for j := 0 to m
4         grid[i][j] :=
             Psi( f, Theta(i/m), Kappa(j/m) )
5   return triangles induced by grid
```

Figure 4: CreateSurface function.

To find $\hat{\mathbf{a}}$, we must be able to intersect the approximate surfaces. Since the surfaces are two-dimensional but exist in three-dimensional space, it takes the intersection of at least three surfaces to arrive at a set of distinct points. However, because of noise in the original data and due to the approximate nature of the piecewise-linear surfaces, the intersection of four or more surfaces will almost certainly be empty. This is not the case when intersecting only three surfaces: if three triangles in $\Re^3$ intersect at a point, then they will still intersect at a point even if each triangle is repositioned slightly. Thus if the screw-transform manifolds are smooth relative to the sample rate (so that the piecewise-linear approximations are reasonably good) and if noise levels in the original data are small, we would expect the mutual intersection points returned by the SURFIT algorithm to be close to the true mutual intersection points. Experimental evidence (Section 5, Fig. 7) indicates this is the case.

Thus the goal now is to find the mutual intersection points of three of the approximate manifolds. Naively, the task might seem impossible: Assuming a 50×50 grid has been used to approximate each manifold (as was used in most of the experiments of Section 5), there are $50 \times 50 \times 2 = 5000$ triangles for each manifold and each triplet of triangles (one from each surface) must be tested for mutual intersection, leading to $5000^3 = 125000000000$ tests. However, we can work in stages, first finding which pairs of triangles in surface 1 and surface 2 intersect, and then testing each intersecting pair against each triangle in surface 3 (Fig. 5). Using the latter approach, there are now $(2m^2)^2 + O(2m^2)$ tests, where the sample rate m is 50 in this case.

```
/* Input: three surfaces represented as collections
of triangles; Output: set of all mutual intersection
points of the surfaces */
FUNCTION MutualIntersections(triangles1,
    triangles2, triangles3)
 1  pairwise is a set of triangle pairs
 2  pairwise = [ ]
 3  foreach t1 in triangles1
 4     foreach t2 in triangles2
 5        if Intersect(t1, t2) then
 6           pairwise := pairwise + [(t1,t2)]
 7  mutual is a set of points
 8  mutual = [ ]
 9  foreach t3 in triangles3
10     foreach (t1,t2) in pairwise
11        if Intersect(t1,t2,t3) then
12           mutual := mutual + [intersection
                point of t1, t2, and t3]
13  return mutual
```

Figure 5: MutualIntersections function.

We can still improve this number greatly by using bounding boxes. Specifically, a bounding box is determined for clumps of nearby triangles on each approximate surface. Next, when intersecting surface 1 with surface 2, these bounding boxes are compared against each other to determine whether any triangle in the clump on surface 1 could possibly intersect with any triangle in the clump on surface 2. Only if the bounding boxes intersect are the triangle-triangle intersection tests performed for each pair of triangles drawn from the two clumps. The vast majority of bounding box tests will yield no intersection, greatly improving speed.

Choosing the size of each bounding-box clump is important. If the box is too large then many bounding-box intersection tests will succeed, leading to many triangle intersection tests. The extreme case is when the clump size equals the entire approximate surface, in which case nothing has been gained. The other extreme has each clump consisting of one triangle, in which case bounding box tests will still help somewhat (since they are faster than triangle-intersection tests). For 50×50 grids we tested a variety of clump sizes for speed and

```
/* Input: set of fundamental matrices fmats and
integer sampling rate srate; Output: best affine
calibration point */
FUNCTION Surfit(fmats, srate)
 surfaces is a set of approximate manifolds
 surfaces = [ ]
 foreach f in fmats
    surfaces := surfaces +
       [ CreateSurface( f, srate ) ]
 bestGoodness := -1
 begin RANSAC loop:
    choose m1, m2, m3 from surfaces
    mutual := MutualIntersections(m1,m2,m3)
    foreach p in mutual
       if ( bestGoodness = -1 ) or
          ( goodness(p) > bestGoodness )
          bestGoodness := goodness(p)
          bestIntersection := p
 return bestIntersection
```

Figure 6: SURFIT with RANSAC.

settled on a clump size of 2×2 grid squares (consisting of 8 triangles)

The worst case for bounding-box intersection tests in $\Re^3$ is 6 floating-point comparisons. However, the process can short circuit after any failed test, and we found each bounding-box test took only about 1.5 floating-point comparisons on average. Thus the intersection process, using 50×50 grids and a 2×2 grid-square clump size, reduces to $(25^2)^2 = 390625$ bounding-box intersections each requiring on average 1.5 floating-point comparisons. These tests typically generated a small number of triangle-triangle intersection tests, and thus the overall run time of the SURFIT algorithm is extremely fast (Fig. 9). Note that, in order for bounding boxes to work effectively, the search space must be normalized (at the beginning of the algorithm in Fig. 5) so that the approximate surfaces are as separated as possible. We normalized the search space so that the three surfaces coincided roughly with the xy-plane, xz-plane, and yz-plane, respectively. The ability to normalize the search space is made possible by the explicit form of the manifold surfaces; this is a great strength of our approach.

When more than 3 fundamental matrices can be determined from the original monocular image sequence, the robust statistical technique of RANSAC [3] can be employed to utilize the extra information and eliminate outlying fundamental matrices. The full SURFIT algorithm with RANSAC is given in Fig. 6. Note that, before starting the main RANSAC loop, one approximate screw-transform manifold surface is generated from each available fundamental matrix, and it is only necessary to do this once. Thereafter, during each pass of the RANSAC loop, three manifolds are drawn from all available manifolds and their mutual intersection points are determined. Thus, if n fundamental matrices are available and k iterations of RANSAC are performed, the total run time is $O(n) + O(k)$.

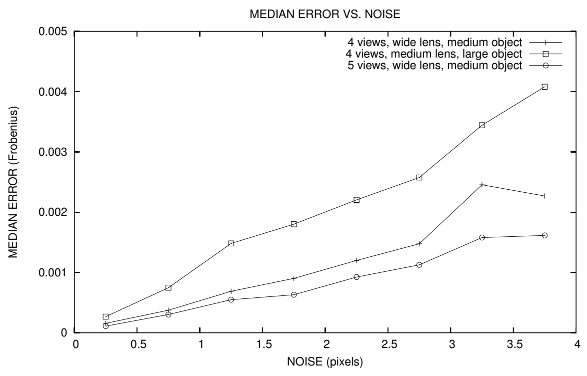

Figure 7: Relationship between noise and error in three different data sets.

5 Experimental results

Experiments using both real and simulated data were performed to answer the following questions:

(1) Does SURFIT work with noise-free data (i.e., is the algorithm correct)? How does performance degrade as noise increases?

(2) How fast does SURFIT run?

(3) Does the use of more than 3 views improve results? By how much?

(4) How many points are contained in the mutual intersection of 3 modulus-constraint manifolds?

(5) Does the method of this paper work with views taken by a real camera (which does not necessarily follow a pinhole model)? How does the reconstruction look?

Questions (1)-(4) were investigated with synthetic data, using synthetic images of dimension 1000×1000 pixels. Timings were performed on a 533 MHz Pentium II. The error measure was the Frobenius norm between the calculated and true internal calibration matrices (after both were normalized).

5.1 Answer 1: Algorithm correctness

The first question is answered by the graphs in Fig. 7 and Fig. 8. Fig. 7 shows error decreasing towards 0 as noise decreases towards 0, indicating that SURFIT would work perfectly in the absence of noise. Also in

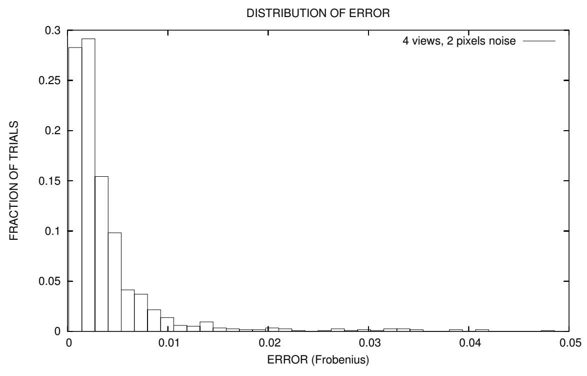

Figure 8: Error distribution for the surface-fitting algorithm applied to synthetic data with noise radius 2 pixels.

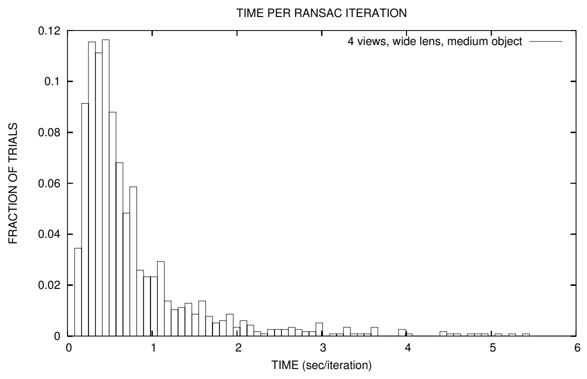

Figure 9: Distribution of run times.

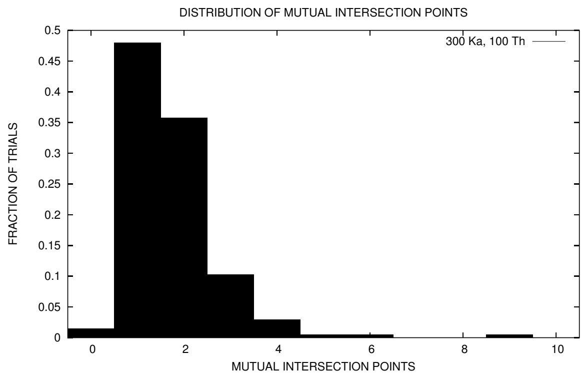

Figure 10: Distribution of the number of mutual intersection points of three modulus-constraint manifolds.

this graph we see noise reaching almost 2.4 pixels before error rises above 0.001 in the 5-camera, wide-field-of-view case, indicating that SURFIT combined with RANSAC performs very well even in the presence of noise. Fig. 8 shows the distribution of calibration error for several hundred trials with 2 pixels of added noise and a medium-size field of view, again showing strong clustering towards small errors.

5.2 Answer 2: Algorithm speed

A stratified self-calibration algorithm has several stages, each requiring some time to perform. Our research only concerns the stage where projective reconstruction is upgraded to affine, and so we only give run times for this part of our implementation. The per-surface running time of the preprocessing phase (in which the approximate surfaces are generated) was 0.955 seconds for a 50×50 grid. The only remaining phase of our algorithm that takes appreciable time is the RANSAC loop. Since the number of RANSAC iterations can vary depending, for example, on how many fundamental matrices are available, we only give timings for individual iterations (Fig. 9) The histogram shows that typical iterations take 0.25-0.75 seconds. When 4 camera views are used for calibration, there are at most ${}_4C_2 = 6$ fundamental matrices and at most ${}_6C_3 = 20$ RANSAC iterations. One could thus expect a run time of 5-15 seconds.

5.3 Answer 3: Advantage of extra views

How much does the use of more than 3 views improve calibration? This question is answered by Fig. 7. To generate this graph, hundreds of trials were run, each having a different noise radius chosen randomly between 0 and 4 pixels. Two of the data sets used 4 cameras while the third set used 5 cameras. The two 4-camera data sets differed in field of view and retinal object size. For each data set, the graph shows median error for every trial with noise radius in range 0 to 0.5 pixels, 0.5 to 1.0 pixels, and so forth.

The graph shows that using 5 views produces notably better results than using 4 views, ceteris paribus. It also shows that using a wide field of view greatly lowers the error from using a medium-size field of view. We believe the latter improvement arises because a wider field of view produces a better fundamental matrix calculation, and our calibration technique relies entirely upon fundamental matrices.

5.4 Answer 4: Number of mutual intersection points

Question 4 is answered by Fig. 10. The figure shows the distribution of the number of mutual intersection points found during 204 trials of a data set with 3 cameras, no noise, roughly 90° field of view, and using 100 samples of θ and 300 samples of κ (an extremely high sampling rate so that the approximate surfaces would be a close match to the true manifolds). The graph shows that roughly 85% of trials yielded either 1 or 2 mutual intersection points. Roughly 2% found no intersection point; this was probably due to the finite granularity of the surfaces that were fitted to each manifold. Few trials yielded more than 3 mutual intersec-

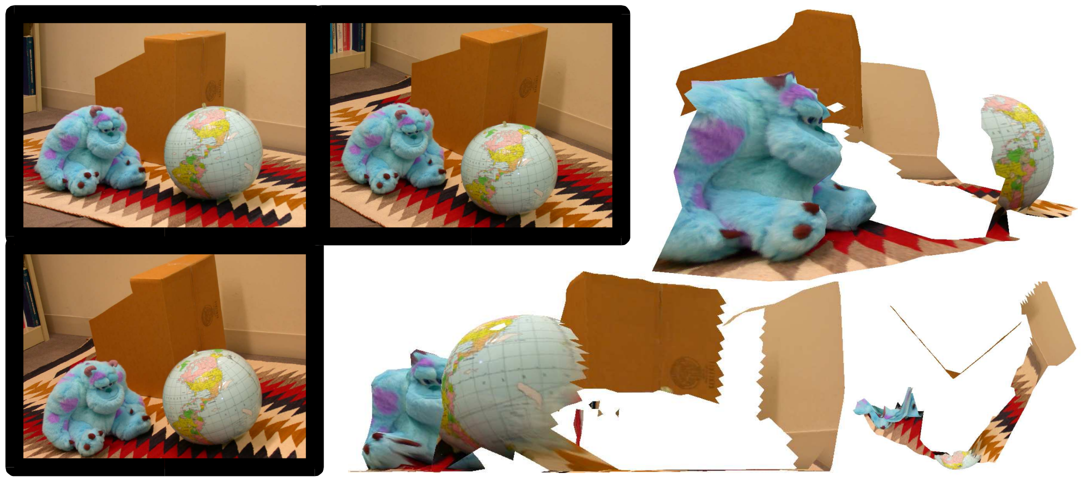
Figure 11: Reconstruction from three real camera views (shown with black borders). An overhead view is in the lower right.

tion points. These results suggest that the 3-view self calibration problem has either 1 or 2 solutions for cameras in general arrangement.

5.5 Answer 5: Real camera performance

A side benefit of recovering a metric reconstruction of camera matrices through self calibration is the ability to perform metric scene reconstruction when feature-point correspondences are available. We applied the techniques of this paper to build a model of a real scene from three photographs. The original photographs and some views of the scene reconstruction are shown in Fig. 11. [[REVIEWERS: A movie of this reconstruction can be found at http://www.cs.wisc.edu/∼rmanning/3dpvt.html]] Despite using only three closely-spaced views to perform the scene reconstruction, the result is very good.

6 Concluding remarks

It is common practice in machine vision to try to solve problems through generic global optimization. However, such solutions are only reliable if the attraction basin for the target global extremum is large enough to be found easily. Furthermore, there is usually no clear stopping point for global optimization because a better local minimum might be found with more search. In this paper, we have formulated the monocular camera self-calibration problem in a way that avoids global optimization. Instead, calibration is determined in a single pass as the physical mutual intersection point(s) of several surfaces. We have given an efficient algorithm called SURFIT for locating these mutual intersection points and demonstrated the performance of the algorithm through experiments.

References

[1] O. D. Faugeras. Stratification of 3-dimensional vision: Projective, affine, and metric representations. *Journal of the Optical Society of America*, 12(3):465–484, 1995.

[2] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera self-calibration: theory and experiments. In *Proc. European Conference on Computer Vision*, pages 321–334, 1992.

[3] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[4] A. W. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3-d reconstruction of independently moving objects. In *Proc. European Conference on Computer Vision*, pages 891–906. Springer-Verlag, June 2000.

[5] R. Hartley. Euclidean reconstruction from uncalibrated views. In A. Zisserman and D. Forsyth, editors, *Applications of Invariance in Computer Vision, LNCS 825*, pages 237–256. Springer-Verlag, 1994.

[6] R. Hartley, E. Hayman, L. de Agapito, and I. Reid. Camera calibration and the search for infinity. In *Proc. Seventh Int. Conf. Computer Vision*, pages I:510–517, 1999.

[7] A. Heyden and K. Astrom. Euclidean reconstruction from constant intrinsic parameters. In *Proc. Int. Conf. on Pattern Recognition*, pages 339–343, 1996.

[8] M. Lourakis and R. Deriche. Camera self-calibration using the kruppa equations and the svd of the fundamental matrix: The case of varying intrinsic parameters. Technical Report 3911, INRIA, March 2000.

[9] R. Manning and C. Dyer. Stratified self calibration from screw-transform manifolds. In *Proc. European Conference on Computer Vision*, volume 4, pages 131–145, 2002.

[10] M. Pollefeys, R. Koch, and L. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. Sixth Int. Conf. Computer Vision*, pages 90–95, 1998.

[11] M. Pollefeys and L. Van Gool. A stratified approach to metric self-calibration. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 407–412, 1997.

[12] W. Triggs. Autocalibration and the absolute quadric. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 609–614, 1997.