Improving the Numerical Stability of Structure from Motion by Algebraic Elimination

Mireille Boutin [a], Ji Zhang [b] and Daniel G. Aliaga [c]

[a] School of ECE, Purdue University, 465 Northwestern Av., West Lafayette, IN, USA;
[b]School of Mathematics, Purdue University, 150 N. University St., West Lafayette, IN, USA ;
[c]Dept. of Computer Sc., Purdue University, 250 N. University St., West Lafayette, IN, USA.

ABSTRACT

Structure from motion (SFM) is the problem of reconstructing the geometry of a scene from a stream of images on which features have been tracked. In this paper, we consider a projective camera model and assume that the internal parameters of the camera are known. Our goal is to reconstruct the geometry of the scene up to a rigid motion (i.e. Euclidean reconstruction.) It has been shown that estimating the pose of the camera from the images is an ill-conditioned problem, as variations in the camera orientation and camera position cannot be distinguished. Unfortunately, the camera pose parameters are an intrinsic part of current formulations of SFM. This leads to numerical instability in the reconstruction of the scene. Using algebraic methods, we obtain a basis for a new formulation of SFM which does not involve pose estimation and thus eliminates this cause of instability.

Keywords: Structure from motion, Euclidean reconstruction, pose estimation, elimination theory, invariants.

1. INTRODUCTION

Being able to accurately simulate large and complex 3D environments is a core challenge of today's computer technology. Indeed for reasons of cost and speed, and to improve the richness of the 3D models, there is a strong desire to replace manual model creation by an automatic acquisition and rendering system. But despite tremendous increases in computational power and storage space, current automated systems perform poorly, even for small and relatively simple environments.

Basically, what we expect from these systems is to be able to acquire the photogrammetric information (i.e. color, reflectance, texture,...) contained in a scene, and to recreate its effect in a picture, movie or other virtual environment. Being able to recreate these effects relies, in parts, on being able to reconstruct the geometry of the scene, i.e. the 3D positions and orientations of the structural elements contained in it. In this paper, we concentrate on this task. Our data acquisition system consists in an internally calibrated camera. We acquire a stream of images by either holding the camera in our hands while walking around the scene, or by placing the camera on a device navigating the scene. A set of feature points is then tracked on the pictures. In order to facilitate the tracking process, the camera movement is assumed to be relatively smooth. We are interested in reconstructing the geometry of the scene observed on the image stream. More precisely, we want to determine the 3D positions of the tracked features from their observed positions on the pictures. This is the well known problem of structure from motion (SFM).

SFM is, in itself, a very difficult problem. Despite decades of research, a satisfying solution still has not been found. In all current methods, the computations involved are often tedious and, in all cases, very sensitive. We attack these problems at the root by mathematically reformulating SFM in a better way. Our goal is to obtain a set of equations describing the geometry reconstruction process that can be solved robustly.

So why is SFM so difficult? One main reason is that, typically, the camera pose parameters are unknown. This is because measuring them requires a complicate setup, and precise estimates are difficult to obtain. So for each picture taken, the positions of the tracked features on the picture yield a set of equations involving the camera parameters and the reconstructed tracked feature positions in 3D. The camera parameters constitute a nuisance because they negatively impact the robustness of the reconstruction. Indeed, it has been shown that estimating the pose of a camera is an ill-conditioned problem.[1] This is due to an inherent confusion between the camera position and the camera orientation which simply cannot be resolved, regardless of the solution scheme used. This is bad news, since the geometric structure of a scene is linked to the camera pose estimates in a highly unstable manner.

There is, of course, a straightforward approach to getting rid of this cause of instability: to algebraically eliminate the camera pose parameters from the set of equations to be solved. For example, Tomasi and Shi[2] invented some SFM equations where the camera orientation parameters do not appear (using angles between camera rays) and used these equations to compute the so-called *direction of heading* of the camera. Numerical experiments demonstrated the robustness of this approach. Similarly, Tomasi[3] described the image changes through the angles between the projection rays and showed how these can be used to reconstruct both structure and motion in a two-dimensional world. Immunity to noise of this method was also noted in experiments, although the results were observed to be critically dependent on camera calibration. More than ten years have passed since these works of Tomasi and Shi have been published and still no complete mathematical framework for SFM without camera parameters (or even just without camera orientations) has been developed. So why was the idea of pose parameter elimination never exploited to its full extent? This is probably because it is easier said than done. Indeed, eliminating variables in a set of equations is difficult, especially when the number of unknowns in the equations is high, as is the case here. The following solves this problem with an effective, systematic method. It is based on a recent publication by Bazin and Boutin[4] which shows that a newly developed algebraic technique from invariant theory[5] can be used to remove extraneous variables in SFM. In Section 5, we present the results of some numerical experiments which indicate that variable elimination does indeed increase robustness. In fact, it appears that these modified equations provide a better framework for SFM refinement than the traditional bundle adjustment method,

2. STANDARD APPROACHES TO SFM

In 3D vision text books (e.g. Ma et al.[6]), the equations describing SFM are typically introduced as

$$\begin{pmatrix} p_{ij} \\ 1 \end{pmatrix} = c_{ij} F_j \begin{pmatrix} P_i \\ 1 \end{pmatrix} \tag{1}$$

where p_{ij} represents the 2D coordinates of the 3D feature point P_i observed on picture j, c_{ij} is a constant, and F_j is a 3-by-4 matrix containing the camera parameters corresponding to picture j. Let us assume that the index i takes values from 1 to n, where n is the number of features tracked on the image, and that the index j takes values from 1 to J, where J is the number of pictures taken. The matrix F_j is commonly called the *fundamental matrix.* When the camera is internally calibrated, one can assume that the fundamental matrix takes the form

$$F_j = \begin{pmatrix} R_j & t_j \end{pmatrix},$$

where R_j is a 3D rotation matrix and t_j is a 3D translation vector. The solution of Equation (1), for $i = 1, \ldots, n$ and $j = 1, \ldots, J$, is only defined up to a Euclidean transformation. Indeed, if F_j and P_i is a solution, then

$$F_j \begin{pmatrix} & \tilde{R} & & \tilde{t} \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} & \tilde{R}^{-1} & & -\tilde{R}^{-1}\tilde{t} \\ 0 & 0 & 0 & 1 \end{pmatrix} P_j$$

is also a solution, for any 3D rotation matrix $\tilde{R}$ and any 3D translation vector $\tilde{t}$. A solution to (1) is thus called a *Euclidean* reconstruction (as opposed to projective reconstructions, to be defined shortly.)

One first approach to solving this equations consists in first solving for the fundamental matrices F_j's. The fundamental matrix is then plugged into the remaining equations which are solved for the P_i's. The structure of F_j is constrained by the fact that it contains a rotation matrix, and so, numerically, this constraint must be taken into account, which somewhat complicates the numerical solution process. Overall, this approach is not robust, as small errors in the fundamental matrix can yield big errors in the P_i's. In particular, a small error in the rotation matrix R yields a big error in P_i when that point is far away from the camera center. So one must estimate all the fundamental matrices with a very high accuracy. Unfortunately, this is impossible, as estimating F_j is equivalent to estimating the camera pose for picture j, a problem which was proven to be ill-conditioned by Fermüeller and Aloimonos.[1]

An alternative approach is the so-called *projective reconstruction*, which looks for an arbitrary 3-by-4 matrix M_j satisfying

$$\begin{pmatrix} p_{ij} \\ 1 \end{pmatrix} = c_{ij} M_j \begin{pmatrix} P_i \\ 1 \end{pmatrix}$$

(i.e. removing the constraints on the structure of the fundamental matrix.) Observe that, if M_j, P_j are a solution of the above equation, then for any non-singular matrix Q, M_jQ and $Q^{-1}P_j$ is also a solution of the above equations. This implies that the reconstruction obtained is only known up to a projective transformation. For this reason, a solution to this set of equations is commonly called a *projective* reconstruction. Removing the constraints on the structure of the fundamental matrix yields a set of multi-linear equations and thus facilitates the solution process. For example, one can solve for all M_j, P_j and c_{ij} using factorization methods[7–9] similar to Tomasi and kanade factorization algorithm.[10] Once the scene geometry is known up to a projective reconstruction, one upgrades to a Euclidean reconstruction by finding an appropriate projective transform (i.e. a non-singular 3-by-3 matrix) Q and applying it to all P_j's. This upgrade is possible when the internal parameters of the camera are unknown.[11] Unfortunately, this approach suffers from the same problem as the first approach, as solving for the M_j's is, in essence, equivalent to solving for the pose.

A third approach, initially proposed by Hartley,[12] is called *bundle adjustment*.[13] It is generally considered to be the most theoretically justified and accurate of all SFM methods at this point. It consists in solving for all unknown parameters (camera parameters and P_i's) simultaneously. This is done numerically by least square minimization. Obviously, this approach is computationally intense and may converge to the wrong solution (local minima), or even diverge. So it needs to be initialized with a good initial guess, which is typically provided by a method falling into one of the two above categories. In fact, because of its high precision, bundle adjustment almost always follows the reconstruction obtained with other methods. Unfortunately, the problem created by the need to estimate the pose remains in this approach as well, since the camera pose parameters are an intrinsic part of the equations to be minimized.

In the next section, we propose a new, improved basis of equations for SFM which aims to improve the numerical stability of the reconstruction. More precisely, we obtain a set of equations which is equivalent to the traditional SFM equations (i.e. the set is *complete* up to functional dependence) where the problematic camera pose parameters have been eliminated, using algebraic manipulation. We present numerical experiments demonstrating the improved stability in Section 5. Despite the fact that we used a very basic numerical scheme (i.e. not much tuning up, as opposed to the current methods for SFM which have been developed for decades) the numerical results clearly show that this is a better approach. In particular, we see that if the results of our numerical solution are used as an initial guess in the bundle adjustment method to attempt to refine them, we obtain no improvement at all. In fact, the refined results tend to worsen. We thus conclude that one should use our equations to design a better refinement step, to be used instead of bundle adjustment.

3. AN ALGEBRAIC METHOD FOR VARIABLE ELIMINATION

Eliminating the camera pose parameters in the SFM equations is more difficult than one could think, a priori. For example, it is obvious that the SFM equations can be viewed as a set of polynomial equations, namely

$$\begin{aligned} \begin{pmatrix} p_{ij} \\ 1 \end{pmatrix} - c_{ij} F_j(c_\theta, s_\theta, c_\phi, s_\phi, c_\psi, s_\psi) \begin{pmatrix} P_i \\ 1 \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} && (2) \\ c_\theta^2 + s_\theta^2 - 1 &= 0 && (3) \\ c_\phi^2 + s_\phi^2 - 1 &= 0 && (4) \\ c_\psi^2 + s_\psi^2 - 1 &= 0 && (5) \end{aligned}$$

where the sine and cosine of the 3D rotation angles θ, ϕ, ψ have been replaced by the variables $c_\theta, s_\theta, c_\phi, s_\phi, c_\psi, s_\psi$ and where an additional set of constraints specified by Equations (3)-(5) has been added. One would thus think that the symbolic elimination tools developed for the case of polynomial equations (e.g. several symbolic algebra packages which can compute Groebner basis) would be well suited for eliminating the nuisance parameters in this case. Unfortunately, the set of equations we are dealing with in the case of SFM is so big and involves so many variables that the programs always seem to run out of memory before the computations finish (we tried both Singular[14] and Macaulay,[15] unsuccessfully) Also, by restricting ourselves to polynomial functions, we are likely to end up with equations of a higher polynomial degree than we began with. Indeed, since division by a variable is not allowed, the more variables are eliminated, the more the degrees of the polynomials in the basis tend to increase. This approach to variable elimination thus has the undesired likely potential of increasing the complexity and the numerical instability of the problem.

In contrast with commutative algebraic approaches, the computational approaches developed in the context of differential geometry are not restricted to polynomial equations, which gives them distinct advantages (and also disadvantages, but this is beyond the scope of this paper...) The computational methods we are interested in come from invariant theory, and so a few, simple definitions must first be given. First, we need the concept of a group of transformations, which is merely a set of transformations on a space which satisfies some properties. More precisely, we demand that the set of transformations be such that

1. (closure) for any two transformations in the transformation group, there exists a third transformation such that successively applying the first two transformations is equivalent to applying the third transformation;

2. (identity element) there exists a transformation which does nothing to the space;

3. (inverse elements) for any transformation in the group, there exists another transformation such that, applying these two transformations successively is the same as doing nothing on the space.

We also need the concept of *orbit* passing through a point. Given a group of transformations on a space, an orbit passing through a point is the set of all points which can be obtained by applying a transformation contained in the group to this point. Finally, we need to define the concept of an invariant. Given a transformation group on a space, an invariant is a real valued function which takes constant values on the orbits of the transformation group. In other words, an invariant is a function whose valued is unchanged by applying a group transformation to its argument.

We begin with a short example which summarizes how we use invariants to eliminate variables. Suppose one is interested in finding the value of an unknown vector $\mathbf{x}$ and that this vector is known to satisfy a set of equations. Now assume that among these equations, there is one that can be written as $f(x,\theta)=0$, where θ is also unknown (i.e. θ is a nuisance parameter.) In this case, we might be tempted to try to eliminate θ from this equation, especially if it is a parameter which is hard to estimate numerically. For example, $\mathbf{x}$ could be a two-dimensional vector and the equation f could be

$$\mathbf{x}=\left(\begin{array}{cc}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{array}\right)\left(\begin{array}{c}k_1\\ k_2\end{array}\right),$$

where k_1, k_2 are some constants defined by the data obtained in an experiment. Eliminating θ in the above case is very easy, because the equation satisfied by $\mathbf{x}$ simply says that $\mathbf{x}$ is on a circle passing through the point (k_1, k_2). Since a circle is made of all those points which lie at a fix distance from the origin, we can replace the above equation by

$$|\mathbf{x}|=|(k_1,k_2)|\,,$$

thus eliminating the parameter θ.

So what have we done, in this example? We have observed that what our initial equation was saying was that all $\mathbf{x}$ satisfying our equation lies on the orbit of a group action, namely that of the group of rotations in the plane. The distance from a point to the origin is an invariant under rotations in the plane. Therefore, the point we are looking for, $\mathbf{x}$, satisfies an equation of the type $|\mathbf{x}|=c$, where c is a constant (because an invariant takes a constant value on an orbit). Now since the (known) vector (k_1, k_2) also lies in this orbit, we know the value of the constant, because $|(k_1,k_2)|=c$ too.

This is the idea behind using invariants to eliminate variables. We first need to find a group action for which the set of all possible values of our unknowns is an orbit. Note that this is not necessarily the case, in general. For this to work, the equations we begin with must define a group of transformation of the type

$$\mathbf{x} = \mathbf{g} * \mathbf{k},$$

where $*$ denotes the application of the transformation g onto a vector $\mathbf{k}$ of known quantities. For any invariant I of this group action, we obtain a new equation

$$I(\mathbf{x}) = I(\mathbf{k}),$$

where the group parameters have been eliminated. Now different invariants may lead to *redundant* equations, (e.g. the distance to the origin, and twice the distance to the origin, which is also in invariant in the example discussed above.) So it is essential to use a set of *independent* invariants. Moreover, some invariants may lead to simpler equations than others (e.g. the distance to the origin is easier to deal with than the exponential of the distance to the origin, which is also an invariant, in the example discussed above.) So it is instructive to consider the set of equations defined by a so-called *generating* set of invariants of the group action. By definition, any invariant is a function (locally) of the invariants contained in a generating set. A set of generating invariants thus provide a basis for formulating all other possible mathematical frameworks where the nuisance parameters do not appear. In particular, we may want to look for combinations of the generating invariants which lead to new invariants satisfying some desirable properties (e.g. simple dependence on certain variables or multi-linearity.)

So for eliminating variables in the case where the unknowns lie on the orbit of a group of transformations, we simply need to obtain a generating set of functionally independent invariants of this group of transformations. Fels and Olver[5] have recently developed a systematic symbolic computation method for obtaining such a set of invariants in the case of a regular Lie group action on a manifold. A non-technical summary of this method can be found in the paper by Bazin and Boutin.[4] The reader interested in this computational method can also refer to a book by Olver[16] for a more detailed, yet accessible, presentation.

4. A BASIS FOR A NEW MATHEMATICAL FORMULATION OF SFM

As explained in the previous section, the invariants of a group of transformation can sometimes be used to eliminate nuisance parameters. However, this method requires that the equations dealt with take a form which is compatible with that of a group transformation parameterized by the nuisance parameters. So we first need to show how the SFM equations can be viewed as group transformation equations where the group parameters include the camera orientation and where the set of all possible reconstructions P_i form an orbit.

To do this is slightly non-trivial and requires some imagination. Indeed, suppose we are given a picture of 3D scene features P_i, $i = 1, \ldots, n$, taken with an internally calibrated camera at time j. We track this feature on the image and obtain the 2D coordinates p_{ij}, for $i = 1, \ldots, n$. The equations relating p_{ij} to P_i are the three equations contained in 1. But these three equations are *not* a group transformation equations, as the fundamental matrix transforms a 4-dimensional vector into a 3-dimensional vector. (Recall that a group transformation is a transformation of a space, not a mapping from one space to another.) So we need to rethink a bit what this equation is telling us.

We begin by observing that one possibility for the reconstruction of P_i is simply $(p_{ij}, 1)^T$, where T denotes the transpose of a matrix. This would be the case, for example, if the camera center position was the origin, if the camera plane lied on the $z = 1$ plane, and if, somehow, the point P_i lies directly on the camera plane. (All right, this is more like a limit case of a solution than a possible solution, but let us not be overly zealous here.) This canonical camera position, orientation and 3D point reconstruction is merely one possibility among infinitely many others, which we now need to express as an orbit of a group of transformation. The set of all possible points P_i can lie anywhere along the rays of light connecting the camera center C_j and the actual 3D feature coordinates. This means that one group parameter, say λ_{ij}, can be a real number which translates P_i along the ray of light. Now the actual ray of light defined by $P_i - C_j$ could be any rotation and translation of the canonical ray of light. So the other group parameters can be taken as 3D rotations and translations of the rays of light. In summary, we can write

$$\begin{aligned} C_j &= R_j \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + t_j && (6) \\ P_i &= R_j \left[\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \lambda_{ij} \left(\begin{pmatrix} p_{ij} \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right) \right] + t_j, && (7) \end{aligned}$$

with R_j is a 3D rotation matrix, t_j a 3D translation vector and λ_{ij} a real number. From these equations, we see that all possible $(C_j, P_1, \ldots, P_n)$ lie on the orbit of a group transformation. The group parameters are R_j, t_j and λ_{ij}, for $i = 1, \ldots, n$ and $j = 1, \ldots, J$. Under this group of transformations, $(C_j, P_1, \ldots, P_n)$ and $((0,0,0)^T, (p_{1j}, 1)^T, \ldots, (p_{nj}, 1)^T)$ lie in the same orbit.

We can also write equivalent equations in projective coordinates. Let us fix $w_0 \neq 0$ and $w_i \neq 0, w_0$. Then there exists W_0, W_i such that

$$\begin{aligned} \begin{pmatrix} W_{0j}C_j \\ W_{0j} \end{pmatrix} &= \begin{pmatrix} & R_j & & t_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} W_iP_i \\ W_i \end{pmatrix} &= \begin{pmatrix} & R_j & & T_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ w_{0j} \end{pmatrix} + \bar{\lambda}_{ij} \left[\begin{pmatrix} w_{ij}p_{ij} \\ w_{ij} \\ w_{ij} \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ w_{0j} \end{pmatrix} \right] \end{aligned}$$

This formulation actually better, since the corresponding invariants take polynomial form, up to a minor change of variables. In contrast, the Euclidean formulation leads to rational invariants.[4] We thus work with the projective formulation, even though this forces us to deal with more variables.

Using Fels-Olver moving frame method, we obtained functionally independent invariants forming a generating set. They give us the following equations, for each picture j:

$$\begin{aligned} \frac{W_{ij}}{W_{ij} - W_{0j}} \frac{W_{1j}}{W_{1j} - W_{0j}} Q_{ij} \cdot Q_{1j} &= cst_{1ij}, \text{ for } i = 3, \ldots, n \\ \frac{W_{ij}}{W_{ij} - W_{0j}} \left(\frac{W_{1j}}{W_{1j} - W_{0j}} \right)^2 \frac{W_{2j}}{W_{2j} - W_{0j}} Q_{1j} \times Q_{ij} \cdot Q_{1j} \times Q_{2j} &= cst_{2ij}, \text{ for } i = 2, \ldots, n, \\ \frac{W_{ij}}{W_{ij} - W_{0j}} \frac{W_{1j}}{W_{1j} - W_{0j}} \frac{W_{2j}}{W_{2j} - W_{0j}} Q_{ij} \cdot Q_{1j} \times Q_{2j} &= cst_{3ij}, \text{ for } i = 1, \ldots, n, \end{aligned}$$

where $Q_{ij} = P_i - C_j$ and the values of the constants are given by the canonical solution, as

$$\begin{aligned} \frac{w_{ij}}{w_{ij} - w_{0j}} \frac{w_{1j}}{w_{1j} - w_{0j}} q_{ij} \cdot q_{1j} &= cst_{1ij}, \text{ for } i = 3, \ldots, n, \\ \frac{w_{ij}}{w_{ij} - w_{0j}} \left(\frac{w_{1j}}{w_{1j} - w_{0j}} \right)^2 \frac{w_{2j}}{w_{2j} - w_{0j}} q_{1j} \times q_{ij} \cdot q_{1j} \times q_{2j} &= cst_{2ij}, \text{ for } i = 2, \ldots, n, \\ \frac{w_{ij}}{w_{ij} - w_{0j}} \frac{w_{1j}}{w_{1j} - w_{0j}} \frac{w_{2j}}{w_{2j} - w_{0j}} q_{ij} \cdot q_{1j} \times q_{2j} &= cst_{3ij}, \text{ for } i = 1, \ldots, n, \end{aligned}$$

where $q_{ij} = (p_{ij}, 1)^T - (0,0,0)^T = (p_{ij}, 1)^T$. Note that these equations do not involve the camera orientation. By independence, none of these equation is redundant. And because our set of invariant is a generating set, any other SFM equation which is independent of the camera orientation is a direct consequence of the above equations. It is unclear which equation, among all the consequences of what we wrote will lead to the best numerical results. However, a slight modification allows us to simplify considerably the equations to be solved.

Indeed, instead of fixing the value of $w_{j0}, \ldots w_{jn}$, and considering $W_{j0}, \ldots, W_{jn}$ as unknowns, it is best to fix the value $W_{j0}, \ldots W_{jn}$, and to consider $w_{j0}, \ldots, w_{jn}$ as unknowns. For example, we can set $W_{j0} = 2$ and $W_{ji} = 1$, for $i = 1, \ldots, n$. Setting $W_{j0} = 2$ also forces $w_{j0} = 2$, as well. Also, to obtain a set of polynomial equations, we then set

$$\gamma_{ij} = \frac{w_{ij}}{w_{ij} - w_{0j}} = \frac{w_{ij}}{w_{ij} - 2}.$$

This gives us the following system of equations for SFM.

$$\begin{array}{rcl} (P_i - C_j) \cdot (P_1 - C_j) & = & \gamma_{ij}\gamma_{1j}k_{1ij}, \text{ for } i = 3, \ldots, n, \\ (P_1 - C_j) \times (P_i - C_j) \cdot (P_1 - C_j) \times (P_2 - C_j) & = & \gamma_{ij}\gamma_{1j}^2\gamma_{2j}k_{2ij}, \text{ for } i = 2, \ldots, n, \\ (P_i - C_j) \cdot (P_1 - C_j) \times (P_2 - C_j) & = & \gamma_{ij}\gamma_{1j}\gamma_{2j}k_{3ij}, \text{ for} i = 1, \ldots, n \end{array} \quad (8)$$

where the value of the constants k's are given by the canonical solution as

$$\begin{array}{rcl} (p_{ij}^T, 1) \cdot (p_{1j}^T, 1) & = & k_{1ij} \\ (p_{1j}^T, 1) \times (p_{ij}^T, 1) \cdot (p_{1j}^T, 1) \times (p_{2j}^T, 1) & = & k_{2ij} \\ (p_{ij}^T, 1) \cdot (p_{1j}^T, 1) \times (p_{2j}^T, 1) & = & k_{3ij} \end{array}$$

Note that the camera orientation parameters have been eliminated. The unknowns are the camera center positions for each picture C_j, for $j = 1, \ldots, J$, and each of the 3D features positions P_i, for $i = 1, \ldots, n$. There are thus 3n-3J unknowns. The number of equations is $3J(n-3)$. So for n and J big enough, we can attempt to solve these equations, or a subset of these equations, numerically. In a generic case, the solution is unique when more equations than unknowns are taken into account.

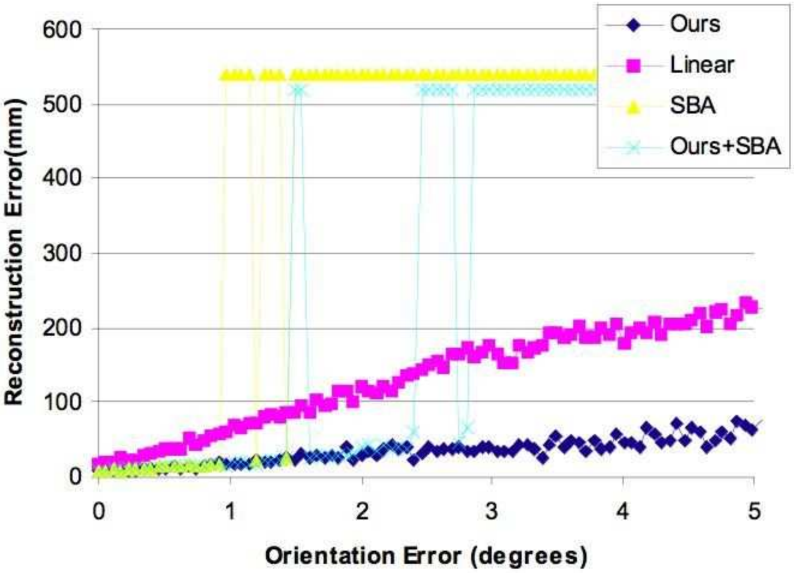

Figure 1. Ground Truth Comparison of Orientation Error Sensitivity. Using a chessboard equipped with a mechanically tracked arm, we obtained a ground truth reconstruction and compared it with the results of different SFM methods. The results displayed were obtained using a linear reconstruction method (SVD), least square minimization of a subset of our equations, a sparse bundle adjustment method with the linear reconstruction as initial guess, and the same sparse bundle adjustment method with our results as initial guess. The vertical axis has been cropped for better display.

5. NUMERICAL EXPERIMENTS

At this point, we are unsure which equations among (8), or which combination of equations, would be best to for SFM, from a numerical point of view. Also, it is unclear whether it is worth it to attempt to remove the camera center from the equations as well. This is the subject of ongoing research. However, the following experiments demonstrate that removing the camera orientation from the equations to be solved leads to a SFM formulation which is less sensitive to pose estimation errors. As pose estimation errors are an issue which cannot be resolved, this thus shows that our formulation has a definite advantage over other mathematical formulations.

Let us suppose we are given a slightly erroneous estimates of the camera pose for each picture. These estimates can be obtained with several different methods, but this is irrelevant here, since we merely want to demonstrate the robustness of our formulation to pose estimation errors. We are mostly interested in errors in the orientation since this is the focus of our mathematical formulation. So we shall vary the amount of error in the camera orientation and observe the effect on a solution obtained with our equations. Note that the camera orientation tends to be the most problematic pose parameter, as it is much more difficult to measure/estimate accurately than the camera position.

Starting with an estimate for the camera pose, the simplest way to estimate the 3D position of the tracked features consists in using SVD to solve Equation (1) for all j. As we already mentioned, even though the equations to solve are linear, small errors in the camera pose can lead to big errors in the reconstruction estimate. What happens when we take this estimate and use it as an initial guess in a least square minimization of a subset of our equations? For example, we can take the following 9 degree two equations from the set 8.

$$\begin{aligned}
(P_1 - C_j) \cdot (P_1 - C_j) &= \gamma_{1j}^2 k_{1ij}, \text{ for } j = 1,2,3 \\
(P_2 - C_j) \cdot (P_1 - C_j) &= \gamma_{2j}\gamma_{1j} k_{1ij} \text{ for } j = 1,2,3 \\
(P_1 - C_j) \times (P_2 - C_j) \cdot (P_1 - C_j) \times (P_2 - C_j) &= \gamma_{1j}^2 \gamma_{2j}^2 k_{2ij}, \text{ for} j = 1,2,3.
\end{aligned}$$

If we simply assume that the camera center position is equal to the estimate and optimize the values of the unknowns $\gamma_{11}, \gamma_{12}, \gamma_{13}, P_1$ and P_2, then the improvement in the results are surprising good.

Our numerical experiments were done on streams of images with a set of features tracked using the Kanade-Lucas-Tomasi automatic automatic feature tracking software package.[17] For example, we took a chessboard dataset, captured using an in-house acquisition system, and varied the orientation error from zero to 5 degrees. We use a mechanically tracked arm (Microscribe Arm G2LX manufactured by Immersion Corporation[18]) to obtain very precise measurements of the camera pose (fraction of a degree precision) and chess board position (millimeter precision) to be used as ground truth in this experiment. The reconstruction error was quantified using Euclidean norm. The results are plotted in Figure 1. (Note that the reconstruction error axis was cropped; the points which lie on a vertical line above the graph correspond to diverging results.) These results show that even this simple approach significantly improves the numerical reconstruction when the pose estimate is imprecise. This figure also displays the results of taking the pose and geometry estimates obtained with SVD and using them as an initial guess in the bundle adjustment method. We used a publicly available sparse bundle adjustment method.[19] As can be seen in Figure 1, when the angle error is small, the quality of the reconstruction obtained with bundle adjustment is equivalent to that of our method. But as soon as the angle error approaches one degree, bundle adjustment tends to diverges, while our method consistently gives good results.

One can argue that there are better methods for obtaining pose and reconstruction, and that these would lead better estimate than our method. But fact is, most existing methods are too noise sensitive and thus involve a refinement phase using bundle adjustment. As a convincing argument, we saw that bundle adjustment does not improve our method. The results of taking the chessboard dataset reconstruction of our method and refining it with bundle adjustment, with varying camera orientation estimates error, is plotted in Figure 1. Not only does bundle adjustment not improve our results, but as as soon as the error approaches 1.5 degree, the refinement stage leads to diverging results.

A more visual illustration of the numerical stability of our results is given by the giraffe dataset results shown in Figure 2. In this experiment, we obtained an initial approximation for the pose using triangulation of four

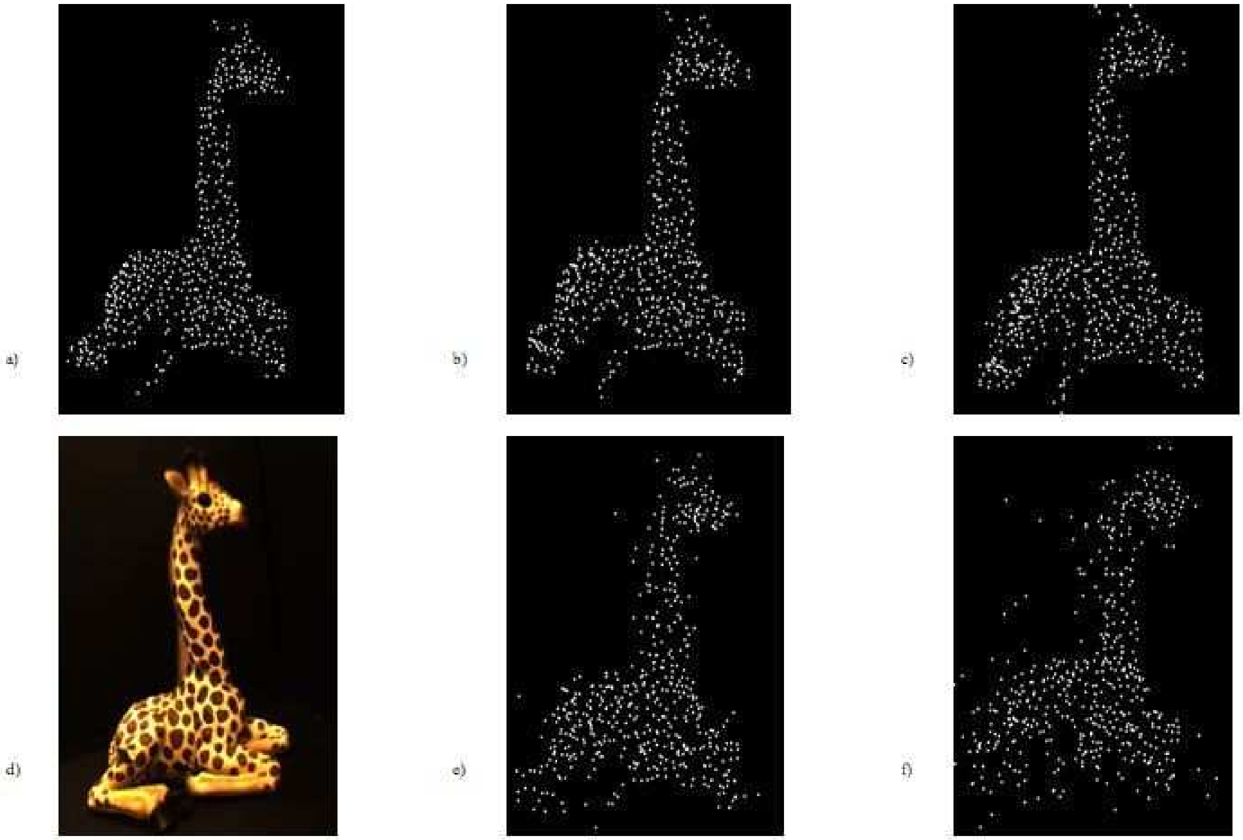

Figure 2. Visual Comparison of Orientation Error Sensitivity We show several reconstructions of a small giraffe (d) under different amounts of orientation error. (a-c) show the 3D feature points reconstructed using a least square minimization of a subset of our equations with zero degrees, six degrees and 12 degrees added to the camera orientation estimate. The reconstruction using SVD degrades significantly when 6 degrees are added to the camera orientation. Sparse bundle adjustment using the results of SVD as initial guess degrade when 12 degrees are added to the camera orientation estimate.

tracked landmarks on the giraffe. We then added varying amounts of errors in the camera orientation estimate. In parts a) b) and c), we display the results obtained with our method when the added orientation errors are zero, six and twelve degrees respectively. This is contrasted with the tracked feature reconstruction using SVD, in part e), which degrades when the added error is six degrees. The results of the SVD were also used as initial guesses for the bundle adjustment method. As seen in part f), the results diverge when the added error is 12 degrees.

We thus conclude that removing the camera orientation from the mathematical formulation of SFM formulation greatly diminishes the sensitivity to pose estimation errors. In particular from the chessboards experiments, we conclude that if a SFM solution method leads to a better estimate of the camera center and 3D tracked features, it is most likely better to refine this estimate using a least square minimization of our equations rather than the bundle adjustment method. But again, we want to emphasize that the particular set of equations chosen and that the numerical scheme used is far from optimal. In particular, we did not attempt to refine the camera center estimates using our equations. But, surely, these preliminary results hold the promise of obtaining a very robust solution of SFM with some more work.

Acknowledgments

This work was funded in parts by NSF grant 0434398.

REFERENCES

1. C. Fermüller and Y. Aloimonos, "Observability of 3D motion," *Int. J. Comput. Vision* **37**(1), pp. 43–63, 2000.
2. C. Tomasi and J. Shi, "Direction of heading from image deformations," in *CVPR93*, pp. 422–427, 1993.
3. C. Tomasi, "Pictures and trails: A new framework for the computation of shape and motion from perspective image sequences," pp. 913–918, IEEE Computer Society Press, (Los Alamitos, CA, USA), June 1994.
4. P.-L. Bazin and M. Boutin, "Structure from motion: a new look from the point of view of invariant theory," *SIAM J. Appl. Math.* **64**(4), pp. 1156–1174, 2004.
5. M. Fels and P. J. Olver, "Moving coframes. I. a practical algorithm," *Acta Appl. Math.* **51**, pp. 161–213, 1998.
6. Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, SpringerVerlag, 2003.
7. P. F. Sturm and B. Triggs, "A factorization based algorithm for multi-image projective structure and motion," in *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, pp. 709–720, Springer-Verlag, (London, UK), 1996.
8. S. Mahamud and M. Hebert, "Iterative projective reconstruction from multiple views," in *Proceedings of IEEE conference on Computer Vision and Pettern Recognition*, **2**, pp. 430–437, 2000.
9. X. Xu and K. Harada, "Sequential projective reconstruction with factorization," *MG&V* **12**(4), pp. 477–487, 2003.
10. C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *Int. J. Comput. Vision* **9**(2), pp. 137–154, 1992.
11. O. D. Faugeras, "What can be seen in three dimensions with an uncalibrated stereo rig," in *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pp. 563–578, Springer-Verlag, (London, UK), 1992.
12. R. I. Hartley, "Euclidean reconstruction from uncalibrated views," in *Proceedings of the Second Joint European - US Workshop on Applications of Invariance in Computer Vision*, pp. 237–256, Springer-Verlag, (London, UK), 1994.
13. B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pp. 298–372, Springer-Verlag, (London, UK), 2000.
14. V. L. G.-M. Greuel and H. Schönemann, "Singular::Plural 2.1," A Computer Algebra System for Noncommutative Polynomial Algebras, Centre for Computer Algebra, University of Kaiserslautern, 2003. `http://www.singular.uni-kl.de/plural`.
15. D. R. Grayson and M. E. Stillman, "Macaulay 2, a software system for research in algebraic geometry."
16. P. J. Olver, *Classical invariant theory*, vol. 44 of *London Mathematical Society Student Texts*, Cambridge University Press, Cambridge, 1999.
17. J. Shi and C. Tomasi, "Good features to track," tech. rep., Ithaca, NY, USA, 1993.
18. Immersion Corporation. http://www.emicroscribe.com.
19. M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm," Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.