Fast and globally convergent Structure and Motion estimation for General Camera Models

Gerald Schweighofer, and Axel Pinz
Institute of Electrical Measurement and Measurement Signal Processing,
Graz University of Technology *

Abstract

This paper presents a novel algorithm to solve the *Structure and Motion* problem. The novelty is in the use of a general camera model, which does not constrain the algorithm to a specific camera, and the use of the *Object Space Error for General Camera Models* as cost function. We show that, using this cost function, the structure and the translation part of the motion can be estimated from the rotation part of the motion in closed form. So only the rotation part of the motion needs to be optimized to estimate the minimum of the total cost function. This results in an iterative algorithm which has a theoretical speedup factor of 8 compared to the *bundle adjustment* method. We also prove the global convergence of the presented algorithm.

1 Introduction

It is widely accepted in the literature, that once a good initialization is obtained, bundle adjustment ([8, 2]) is the method of choice to minimize a given cost function to achieve high accuracy. However, bundle adjustment and other non-linear optimization algorithms will only converge to the *correct* solution when a good initial point is given. This is due to the fact that too many parameters need to be brought into a consistent form. One solution is to estimate a good initialization. The other way, which is not so common, is to reduce the number of parameters for the optimization step. This brings two advantages: First, convergence is faster, due to the smaller parameter space. Second, the finding of a consistent form becomes easier and so the initialization requirements are not as stringent as for bundle adjustment.

General Camera Models (GCM) were introduced to *Multi-View Geometry* by [6, 7]. The used model [1] allows to describe all cameras which capture light rays travelling along a straight line (e.g. catadioptric camera, central camera, camera looking at a reflective sphere, etc.). Based on *Plücker Coordinates* as a representation of the light rays, Sturm [7] derived *Essential* matrices for two view motion and different camera models. His assumption is, that all light rays (represented as a line), which measure the same point in space, intersect in a common point. Due to the nature of a real measurement, this assumption does not hold. Therefore, we present an iterative algorithm for *Structure and Motion* estimation based on a cost function for GCM (which includes the perspective camera).

*E-Mail: gerald.schweighofer@tugraz.at, axel.pinz@tugraz.at

BMVC 2006 doi:10.5244/C.20.16

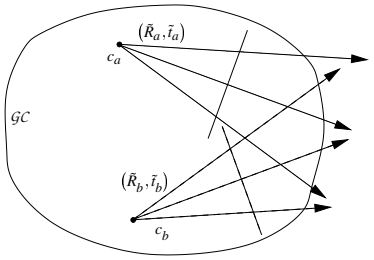


Figure 1: Cluster of perspective cameras.

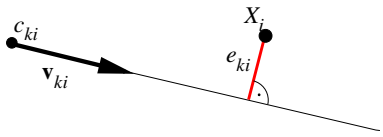


Figure 2: Scaled Object Space Error e_{ki}.

1.1 General Camera Model

The GCM described in [1] replaces the use of a pixel (which contains the position on the sensor and its intensity) as the atomic element with the *raxel*, which is the union of the pixel and the direction of the light ray. We represent this *raxel* by a point $\mathbf{c}$ (which could be the physical position of the pixel inside the camera, or for a perspective camera the center of projection) and a vector $\mathbf{v}$ which defines the ray. So a measurement with this sensor is the tuple ($\mathbf{c}$,$\mathbf{v}$). A simple example of such a GCM is shown in Figure 1. This *General Camera* $\mathcal{GC}$ consists of two rigidly linked perspective cameras (c_a and c_b). For such a system the measurement tuple ($\mathbf{c}$,$\mathbf{v}$) can be estimated by $(-\tilde{R}_a^T\tilde{t}_a,\ \tilde{R}_a^T\tilde{K}_a^{-1}\mathbf{p})$ for a measured point $\mathbf{p}$ in camera a (with $\tilde{R}_a$, $\tilde{t}_a$ as exterior and $\tilde{K}_a$ as interior calibration parameters). Similar equations can be given for all kinds of calibrated cameras to obtain the tuple ($\mathbf{c}$,$\mathbf{v}$) from a measurement.

1.2 Cost function

Using this tuple ($\mathbf{c}$,$\mathbf{v}$) as a measure of our camera we define here the cost function: **Object Space Error for General Camera Models**, which is based on the "Object Space Error" [4] , as

$$e_{ki} = \left\| \left(I - V_{ki}\right)\left(R_k\mathbf{X}_i + \mathbf{t}_k - \mathbf{c}_{ki}\right)\right\|^2 \quad \text{with} \quad V_{ki} = \frac{\mathbf{v}_{ki}\mathbf{v}_{ki}^T}{\mathbf{v}_{ki}^T\mathbf{v}_{ki}}. \tag{1}$$

The matrix V_{ki} is called *line of sight projection matrix* and encodes in a simple way the ray of our measurement.

As an example the cost e_{ki} for one point X_i is shown in Figure 2. This cost measures the distance between a world point X_i and the projection of this point onto the *line of sight*. The index i represents one of n_i points, and the index k denotes one of n_k poses of our general camera.

2 Problem formulation

A moving general camera $\mathcal{GC}$ has measured structure points (X_i) for n_k different positions and orientations in space. The goal is to find from these corresponding measurements ($\mathbf{c}_{ki}$,$\mathbf{v}_{ki}$) the coordinates of the scene structure X_i and the position and orientation of the general camera (R_k,t_k) for all n_k poses.

Based on the *Object Space Error for General Camera Models* we define the solution to the *Structure and Motion* problem as a minimization of the *total cost* $E(\mathcal{R}, t, \mathcal{X})$

$$\arg\min_{\mathcal{R},t,\mathcal{X}} E(\mathcal{R}, t, \mathcal{X}) = \arg\min_{R_k,t_k,X_i} \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} e_{ki}(R_k, t_k, X_i), \tag{2}$$

writing only the unknown parameters (R_k,t_k,X_i) as parameters of the cost function (Equation 1). So the goal is to find those structure $\mathcal{X} = \{X_i, ..., X_{n_i}\}$ and motion ($\mathcal{R} = \{R_1, ..., R_{n_k}\}$, $t = \{t_1, ..., t_{n_k}\}$) parameters which minimize the total cost.

3 Derivation of the algorithm

To attain a simpler notation, the algorithm is derived with the assumption, that every camera sees every feature point once, but the algorithm is **not** limited to that case. The derivation is split into three parts. First, we present a parameterization for the optimal structure estimation ($\mathcal{X} := \mathcal{X}(\mathcal{R}, t)$) which is linear in the translation part of the motion (t). Second, using this parameterization of the structure we derive an optimal translation estimation ($t := t(\mathcal{R})$) which minimizes the total cost. Finally, we present the iterative step to improve the rotation part ($\mathcal{R}^{(\lambda+1)} := \mathcal{R}(\mathcal{R}^{(\lambda)}, t^{(\lambda)}, \mathcal{X}^{(\lambda)})$) of the motion.

3.1 Optimal structure estimation

Let us assume in this section that we know already the motion parameters R_k and t_k, which minimize Equation 2. We want to find the optimal reconstruction for the known motion parameters as

$$\arg\min_{X_i} \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} e_{ki}(R_k, t_k, X_i) \rightarrow \textit{we know } R_k \textit{ and } t_k \rightarrow \sum_{i=0}^{n_i} \arg\min_{X_i} \sum_{k=1}^{n_k} e_{ki}(X_i). \tag{3}$$

So solving for an optimal X_i is given by

$$\frac{\partial \left(\sum_{k=1}^{n_k} e_{ki}(\mathbf{X}_i) \right)}{\partial \mathbf{X}_i} = \sum_{k=1}^{n_k} \frac{\partial e_{ki}(\mathbf{X}_i)}{\partial \mathbf{X}_i} = 0. \tag{4}$$

Writing the cost function (Equation 1) as a dot product

$$\begin{aligned} e_{ki} &= \left[(I - V_{ki}) \left(R_k \mathbf{X}_i + \mathbf{t}_k - \mathbf{c}_{ki} \right) \right]^T \left[(I - V_{ki}) \left(R_k \mathbf{X}_i + \mathbf{t}_k - \mathbf{c}_{ki} \right) \right] \qquad (5) \\ &= \mathbf{X}_i^T R_k^T Q_{ki} R_k \mathbf{X}_i + 2\mathbf{X}_i^T R_k^T Q_{ki} \mathbf{t}_k - 2\mathbf{X}_i^T R_k^T Q_{ki} \mathbf{c}_{ki} - 2\mathbf{c}_{ki}^T Q_{ki} \mathbf{t}_k + \mathbf{t}_k^T Q_{ki} \mathbf{t}_k + \mathbf{c}_{ki}^T Q_{ki} \mathbf{c}_{ki}, \end{aligned}$$

where $Q_{ki} = \left(I - V_{ki} \right)^T \left(I - V_{ki} \right)$, and differentiating e_{ki} (Equation 5) with respect to $\mathbf{X}_i$ gives us

$$\frac{\partial e_{ki}}{\partial X_i} = 2R_k^T Q_{ki} R_k \mathbf{X}_i + 2R_k^T Q_{ki} \mathbf{t}_k - 2R_k^T Q_{ki} \mathbf{c}_{ki}. \tag{6}$$

Using this result in Equation 4 gives us an equation for the optimal reconstruction of X_i:

$$\mathbf{X}_i = \tilde{x}_i + \sum_{k=1}^{n_k} \tilde{X}_{ki} \mathbf{t}_k, \tag{7}$$

with

$$\tilde{X}_{ki} = -\tilde{Y}_i R_k^T Q_{ki} \ , \ \tilde{x}_i = \tilde{Y}_i \sum_{k=1}^{n_k} R_k^T Q_{ki} \mathbf{c}_{ki} \ \text{ and } \ \tilde{Y}_i = \left(\sum_{\alpha=1}^{n_k} R_\alpha^T Q_{\alpha i} R_\alpha \right)^{-1}. \tag{8}$$

We see that the optimal structure (Equation 7), which minimizes Equation 2, is *linear* in the translation part of the motion (t_k's).

3.2 Optimal translation estimation

Using the parameterization of the optimal structure from Section 3.1 our optimization problem changes to a simpler one

$$\arg\min_{R_k,t_k,X_i} \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} e_{ki}(R_k,t_k,X_i) \xrightarrow{\text{use Equation 7}} \arg\min_{R_k,t_k} \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} e_{ki}(R_k,t_k,X_i(R_k,t_k)), \tag{9}$$

where we only need to optimize the motion parameters (R_k and t_k).
In this section we show that we can, using this formulation, estimate the optimal translation part of the motion w.r.t. the cost function (Equation 1) in a direct way. Again, we begin by writing the cost function as a dot product

$$\begin{aligned} e_{ki} &= \left[R_k \tilde{x}_i - \mathbf{c}_{ki} + \mathbf{t}_k + R_k \sum_{\alpha=1}^{n_k} \tilde{X}_{\alpha i} \mathbf{t}_\alpha \right]^T Q_{ki} \left[R_k \tilde{x}_i - \mathbf{c}_{ki} + \mathbf{t}_k + R_k \sum_{\alpha=1}^{n_k} \tilde{X}_{\alpha i} \mathbf{t}_\alpha \right] \quad (10) \\ &= (\tilde{x}_i^T R_k^T - \mathbf{c}_{ki}^T) Q_{ki} (R_k \tilde{x}_i - \mathbf{c}_{ki}) \\ &+ 2\mathbf{t}_k^T Q_{ki} (R_k \tilde{x}_i - \mathbf{c}_{ki}) + 2\left(\sum_{\alpha=1}^{n_k} \mathbf{t}_\alpha^T \tilde{X}_{\alpha i}^T \right) R_k^T Q_{ki} (R_k \tilde{x}_i - \mathbf{c}_{ki}) \\ &+ \mathbf{t}_k^T Q_{ki} \mathbf{t}_k + 2\mathbf{t}_k^T Q_{ki} R_k \left(\sum_{\alpha=1}^{n_k} \tilde{X}_{\alpha i} \mathbf{t}_\alpha \right) + \left(\sum_{\alpha=1}^{n_k} \mathbf{t}_\alpha^T \tilde{X}_{\alpha i}^T \right) R_k^T Q_{ki} R_k \left(\sum_{\alpha=1}^{n_k} \tilde{X}_{\alpha i} \mathbf{t}_\alpha \right). \end{aligned}$$

An optimal solution for all t_q ($q = 1 \ldots n_k$) is given by

$$\frac{\partial \left(\sum_{k=1}^{n_k} \sum_{i=1}^{n_i} e_{ki}(t_1 \ldots t_{n_k}) \right)}{\partial t_q} = \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} \frac{\partial e_{ki}(t_1 \ldots t_{n_k})}{\partial t_q} = 0 \quad \forall q = 1 \ldots n_k. \tag{11}$$

Differentiating e_{ki} from Equation 10 w.r.t. t_q gives (after a few lines of math)

$$\begin{aligned} \left. \frac{\partial e_{ki}}{\partial t_q} \right|_{q \neq k} &= 2\tilde{X}_{qi}^T R_k^T Q_{ki} R_k \left(\sum_{\alpha=1}^{n_k} \tilde{X}_{\alpha i} \mathbf{t}_\alpha \right) + 2\tilde{X}_{qi}^T R_k^T Q_{ki} (R_k \tilde{x}_i - \mathbf{c}_{ki}) + 2\tilde{X}_{qi}^T R_k^T Q_{ki} \mathbf{t}_k \quad (12) \\ \left. \frac{\partial e_{ki}}{\partial t_q} \right|_{q = k} &= 2\tilde{X}_{ki}^T R_k^T Q_{ki} R_k \left(\sum_{\alpha=1}^{n_k} \tilde{X}_{\alpha i} \mathbf{t}_\alpha \right) + 2\tilde{X}_{ki}^T R_k^T Q_{ki} (R_k \tilde{x}_i - \mathbf{c}_{ki}) + 2\tilde{X}_{ki}^T R_k^T Q_{ki} \mathbf{t}_k \\ &+ 2Q_{ki} (R_k \tilde{x}_i - \mathbf{c}_{ki}) + 2Q_{ki} \mathbf{t}_k + 2Q_{ki} R_k \left(\sum_{\alpha=1}^{n_k} \tilde{X}_{\alpha i} \mathbf{t}_\alpha \right). \quad (13) \end{aligned}$$

The combination of Equations 11-13 gives us n_k equations of the form

$$\frac{\partial \left(\sum_{k=1}^{n_k} \sum_{i=1}^{n_i} e_{ki}(t_1 ... t_{n_k}) \right)}{\partial t_q} =$$

$$= \sum_{k=1}^{n_k} (\sum_{i=1}^{n_i} \tilde{X}_{qi}^T R_k^T Q_{ki}) \mathbf{t}_k) + \sum_{i=1}^{n_i} Q_{qi} \mathbf{t}_q + \sum_{i=1}^{n_i} Q_{qi} (R_q \tilde{x}_i - \mathbf{c}_{qi}) = 0. \quad (14)$$

By stacking all n_k equations together we obtain a simple linear system of equations

$$A_t \mathbf{t} = \mathbf{b}_t, \quad (15)$$

where $\mathbf{t} = [t_1^T t_2^T ... t_{n_k}^T]^T$ is a vector of the unknown translation part of the motion. A solution for $\mathbf{t}$ can easily be obtained by $\mathbf{t} = A_t^{-1} \mathbf{b}_t$.

3.3 Iteration Step

In the previous section we have shown that given the rotation part of the motion, we can estimate the translation part of the motion in a direct step Equation 15. Furthermore we can use this optimal translation part of the motion to estimate the optimal structure from Equation 7. Let us rewrite the optimization problem (Equation 1) with this knowledge as

$$\arg\min_{\mathcal{R}} E(\mathcal{R}) = \arg\min_{\mathcal{R}} \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} \left\| \left(I - V_{ki} \right) \left(R_k \mathbf{X}_i^*(\mathcal{R}) + \mathbf{t}_k^*(\mathcal{R}) - \mathbf{c}_{ki} \right) \right\|^2 =$$

$$\arg\min_{\mathcal{R}} \sum_{k=1}^{n_k} \sum_{i=1}^{n_i} \left\| R_k \mathbf{X}_i^*(\mathcal{R}) + \mathbf{t}_k^*(\mathcal{R}) - \mathbf{c}_{ki} - V_{ki} \mathbf{q}_{ki}(\mathcal{R}) \right\|^2, \quad (16)$$

with

$$\mathbf{q}_{ki}(\mathcal{R}) = R_k \mathbf{X}_i^*(\mathcal{R}) + \mathbf{t}_k^*(\mathcal{R}) - \mathbf{c}_{ki}. \quad (17)$$

The $\mathbf{t}_k^*$ are the result of the optimal translation estimation and $\mathbf{X}_i^*$ are the result of the optimal structure estimation using $\mathbf{t}_k^*$. $\mathcal{R}$ represents all rotation matrices $\{R_1, ..., R_{n_k}\}$ which are used to estimate $\mathbf{X}_i^*$ and $\mathbf{t}_k^*$. It is not possible to solve for all $R_k \in \mathcal{R}$ in closed form. However, it is possible to give an iterative way to estimate all R_k (this is inspired by [4]): First, assume that the λ^{th} estimate of $\mathcal{R}$ is $\mathcal{R}^{(\lambda)} = \{R_1^{(\lambda)}, ..., R_{n_k}^{(\lambda)}\}$, $\mathbf{X}_i^{*(\lambda)} = \mathbf{X}_i^*(\mathcal{R}^{(\lambda)})$ and $\mathbf{q}_{ki}^{(\lambda)} = \mathbf{q}_{ki}(\mathcal{R}^{(\lambda)})$. The next estimate of the $R_k^{(\lambda+1)}$ is estimated as the solution of the *absolute orientation problem* [3]

$$R_k^{(\lambda+1)} = \arg\min_{R} \sum_{i=1}^{n_i} \left\| R \mathbf{X}_i^{*(\lambda)} + \mathbf{t}_k^{*(\lambda)} + \left(-\mathbf{c}_{ki} - V_{ki} \mathbf{q}_{ki}^{(\lambda)} \right) \right\|^2 \quad \text{subject to } R^T R = I. \quad (18)$$

This constrained problem can be solved using quaternions [3] or singular value decomposition (SVD) [4]. In this formulation $\mathbf{X}_i^{*(\lambda)}$ represents the actual estimated structure points. $V_{ki} \mathbf{q}_{ki}^{(\lambda)}$ represents the projection of the transformed structure points with the actual pose of the k^{th} sensor ($R_k^{(\lambda)}$,$t_k^{(\lambda)}$) to the *line of sight* V_{ki}.

3.4 Summary of the Algorithm

The complete algorithm works as follows:

1. Start with an initial set of rotations $\mathcal{R}^{(\lambda)}$, $\lambda = 1$.
2. Estimate the parameterization of the structure ($\tilde{X}_{ki}$ and $\tilde{x}_i$) using Equation 8.
3. Estimate an optimal translation $\mathbf{t}_k^{*(\lambda)}(\mathcal{R})$ using Equation 15 and ($\tilde{X}_{ki}$ and $\tilde{x}_i$).
4. Estimate an optimal structure $\mathbf{X}_i^{*(\lambda)}$ using Equation 7.
5. Solve the *absolute orientation problem* in Equation 18 for each camera position k to get better estimates of the rotations $\mathcal{R}^{(\lambda+1)}$.
6. Repeat steps 2-5 until convergence.

4 Proof of global convergence

To prove the global convergence of the proposed algorithm we need to show that (based on the Global Convergence Theorem [5] and the proof in [4]):

1. The algorithm is closed (definition given in [4]).
2. All $\{\mathcal{R}_k^{(\lambda)}\}$ are contained in a compact set.
3. The algorithm decreases the total cost unless a solution is reached.

The output of the algorithm ($\mathcal{R}^{\lambda}$) are a set of solutions to the *absolute orientation algorithm*. Since the output of the *absolute orientation algorithm* is closed (proof based on a closed mapping of the SVD in [4]) and the input parameters are continuous (steps 2-4 of the algorithm) the first condition is met. Since the algorithm always generates a set of rotation matrices and the set of orthogonal matrices is compact the second criterion is also fulfilled. To prove the third criterion we use the fact that [4]

$$\begin{aligned}\sum_{i=1}^{n_i}\left|\left|\mathbf{q}_{ki}^{(\lambda+1)}-V_{ki}\mathbf{q}_{ki}^{(\lambda+1)}\right|\right|^2 &= \sum_{i=1}^{n_i}\left|\left|\mathbf{q}_{ki}^{(\lambda+1)}-V_{ki}\mathbf{q}_{ki}^{(\lambda)}+V_{ki}\mathbf{q}_{ki}^{(\lambda)}-V_{ki}\mathbf{q}_{ki}^{(\lambda+1)}\right|\right|^2 \\ &= \sum_{i=1}^{n_i}\left|\left|\mathbf{q}_{ki}^{(\lambda+1)}-V_{ki}\mathbf{q}_{ki}^{(\lambda)}\right|\right|^2-\sum_{i=1}^{n_i}\left|\left|V_{ki}\mathbf{q}_{ki}^{(\lambda+1)}-V_{ki}\mathbf{q}_{ki}^{(\lambda)}\right|\right|^2, \qquad (19)\end{aligned}$$

which gives us a relation beetween the total cost of two iteration steps. Writing the total cost $E(\mathcal{R}^{(\lambda+1)})$ as a function of the $\mathbf{q}_{ki}^{\lambda+1}$ and by the result of Equation 19 gives us

$$\begin{aligned}E(\mathcal{R}^{(\lambda+1)}) &= \sum_{k=1}^{n_k}\sum_{i=1}^{n_i}\left\|R_k^{(\lambda+1)}\mathbf{X}_i^*(\mathcal{R}^{(\lambda+1)})+\mathbf{t}_k^*(\mathcal{R}^{(\lambda+1)})+\mathbf{c}_{ki}-V_{ki}\mathbf{q}_{ki}(\mathcal{R}^{(\lambda+1)})\right\|^2 \qquad (20)\\ &= \sum_{k=1}^{n_k}\sum_{i=1}^{n_i}\left\|\mathbf{q}_{ki}^{(\lambda+1)}-V_{ki}\mathbf{q}_{ki}^{(\lambda+1)}\right\|^2 \xrightarrow{\text{use of Equation 19}} \\ &= \sum_{k=1}^{n_k}\sum_{i=1}^{n_i}\left|\left|\mathbf{q}_{ki}^{(\lambda+1)}-V_{ki}\mathbf{q}_{ki}^{(\lambda)}\right|\right|^2-\sum_{k=1}^{n_k}\sum_{i=1}^{n_i}\left|\left|V_{ki}\mathbf{q}_{ki}^{(\lambda+1)}-V_{ki}\mathbf{q}_{ki}^{(\lambda)}\right|\right|^2. \qquad (21)\end{aligned}$$

The first term in the bottom line of Equation 21 represents the total cost after solving the *absolute orientation algorithm*. Because the *absolute orientation algorithm* Equation 18 gives us always the best possible solution, it holds that

$$\sum_{i=1}^{n_i} \left|\left| \mathbf{q}_{ki}^{(\lambda+1)} - V_{ki}\mathbf{q}_{ki}^{(\lambda)} \right|\right|^2 \leq \sum_{i=1}^{n_i} \left|\left| \mathbf{q}_{ki}^{(\lambda)} - V_{ki}\mathbf{q}_{ki}^{(\lambda)} \right|\right|^2. \tag{22}$$

Finally we obtain

$$E(\mathcal{R}^{(\lambda+1)}) \leq E(\mathcal{R}^{(\lambda)}) - \sum_{k=1}^{n_k}\sum_{i=1}^{n_i} \left|\left| V_{ki}\mathbf{q}_{ki}^{(\lambda+1)} - V_{ki}\mathbf{q}_{ki}^{(\lambda)} \right|\right|^2. \tag{23}$$

Since the last term of Equation 23 is always positive, this means that the total cost $E(\mathcal{R}^{(\lambda)})$ decreases at every iteration until a *stable* point is reached. If this term is equal to zero it can be shown that this is equivalent to $E(\mathcal{R}^{(\lambda+1)}) \equiv E(\mathcal{R}^{(\lambda)})$, which is a *stable* point.

5 Time Complexity

In this section we derive the time complexity of the proposed algorithm by estimating the complexity of each step of the algorithm (Section 3.4):

- Step 2: Estimating the matrix $\tilde{Y}_i$ needs $O(n_k)$ time per point, and so $\tilde{X}_{ki}$. $\tilde{x}_i$ needs $O(n_k + n_k)$. The time complexity to obtain all $\tilde{X}_{ki}$ is $\mathbf{O(n_i n_k)}$.
- Step 3: Building the Matrix A_t needs stacking of Equation 14 n_k times, which itself needs $O(n_k n_i)$. So the time for building A_t needs $O(n_i n_k^2)$. Building b_t takes $O(n_k n_i)$ time. Since we need to take the inverse of the A_t (size: $3n_k$x$3n_k$) this step takes $\mathbf{O_{inv}(3n_k)} + \mathbf{O(n_i n_k^2)}$ time. (The time complexity of the inverse of a matrix of size n x n will be denoted by $O_{inv}(n)$.)
- Step 4: The estimation of the structure from Equation 7 takes for each structure point $O(n_k + 1)$ time, which leads to $\mathbf{O(n_k n_i)}$ for all structure points.
- Step 5: Solving the *absolute orientation problem* n_k times requires $\mathbf{O(n_k n_i)}$ time, because the complexity of the *absolute orientation problem* is linear with the number of points.

We see that Step 3 of the proposed algorithm is the most time consuming, and so the complexity of one iteration of the proposed algorithm is $\mathbf{O_{inv}(3n_k)} + \mathbf{O(n_i n_k^2)}$.

In Section 3 the algorithm was derived based on the assumption, that every camera sees every feature point once. This was done to allow a simple notation for easy understanding of the algorithm. But the algorithm is not limited to that case. To extend the algorithm in a way, that some points $\tilde{X}_i$ are not seen by some cameras, only the corresponding part of the sums needs to be removed. In the case that a point $\tilde{X}_i$ is seen twice by one camera (this is possible because we are dealing with generalized cameras), only the corresponding part of the sums needs to be added.
We introduce n_m as the total number of measurements[1]. Using n_m, and following the same complexity estimation procedure as above (Step 2-5), we obtain the complexity of

[1] In the simple case, where each point is measured once by each camera, n_m would be equal to $n_i n_k$.

Algorithm	Time Complexity
Bundle Adjustment	$O_{inv}(6n_k + 3n_i) + O((6n_k + 3n_i)n_m)$
Bundle Adjustment (sparse, Schur complement)	$O_{inv}(6n_k) + O((6n_k)^2 n_i + n_m)$
Proposed Algorithm	$O_{inv}(3n_k) + O(n_m n_k)$

Table 1: Time Complexity of one iteration for different Algorithms.

one iteration of the proposed algorithm w.r.t. n_m as: $\mathbf{O_{inv}(3n_k)} + \mathbf{O(n_m n_k)}$. This means that for $n_k \ll n_i$ it is linear in the number of total measurements and in the case $n_i \ll n_k$ it is cubic (time complexity of the inverse) in the number of camera positions.

In Table 1 we compare the time complexity of our algorithm with two other known iterative methods for structure and motion estimation:

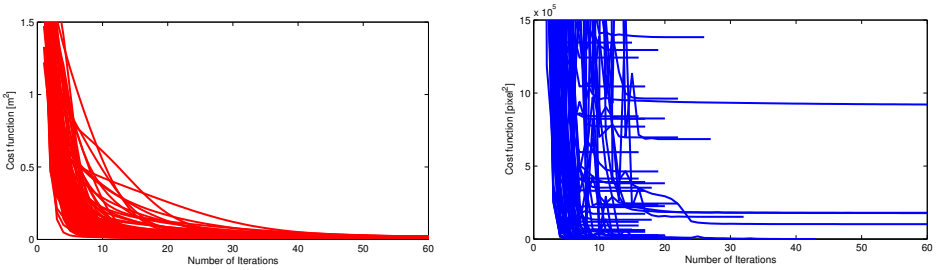
- Bundle adjustment is known as non-linear optimization over all parameters. In our case we have $3n_i$ structure parameters and $6n_k$ (3 for each rotation and translation part of a camera) motion parameters. The iterative step of bundle adjustment consists of the following steps: First, the *Jacobian matrix J*, which is the derivation of all measurements w.r.t. all parameters $(6n_k + 3n_i)$, is estimated. Second, this matrix is used to estimate the update step which is based on the *normal* equation: $\delta p = (J^T J)^{-1} J^T \varepsilon$. The most time consuming parts are the estimation of the product $J^T J$ and the calculation of the inverse of it. So the time complexity of one iteration is $O_{inv}(6n_k + 3n_i) + O(2(6n_k + 3n_i)n_m)$ [8, 2].

- An extension to bundle adjustment is the *sparse* version of it. Because most of the elements of the *Jacobian J* are zero, one can use structure information to compute the update step in a faster way. The estimation of $J^T J$ needs then only $O(n_m)$. Further the estimate of an intermediate matrix S of size $6n_k$ x $6n_k$ needs $O((6n_k)^2 n_i)$ and the final inverse of this $O_{inv}(6n_k)$. So the complete time complexity of this algorithm is $O_{inv}(6n_k) + O((6n_k)^2 n_i + n_m)$ [8, 2].

Comparing the algorithms in Table 1 we see that the most time consuming part of all algorithms is the computation of the inverse of a matrix. The time complexity is cubic in the size of the matrix. A reduction of the size in this step by a factor of two (as we have with our proposed algorithm w.r.t. the sparse bundle algorithm) gives a theoretical speedup factor of 8 of this step. The estimation of the precise improvement of our method would require to count all operations of both algorithms, but the order of improvement is about a factor of 8.

6 Experiments

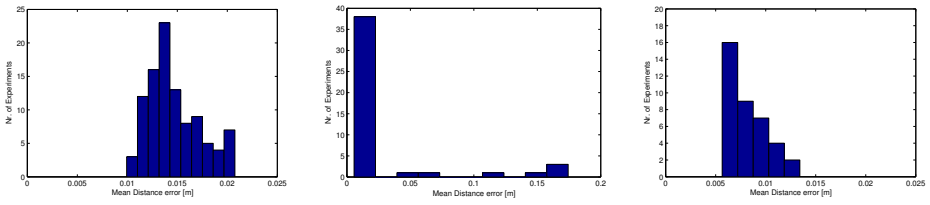
We tested the proposed algorithm on simulated data. The setup was as follows: Points are randomly distributed in a cube of three meter side length. The general camera used in the experiments was a cluster of two perspective cameras, as described in Section 1.1. The general camera was placed randomly at 4 positions ($n_k = 4$) around the point cloud at different distances (approx. 3 m). From the point cloud 20 points ($n_i = 20$), which are visible in all cameras, were selected. These points are projected into the cameras. Gaussian noise with a *sigma* of 2 pixel is added to all image points. From the noisy measurements



(a) (b)

Figure 3: **Convergence of the Algorithm:** (a) proposed algorithm (b) bundle adjustment.

the input parameters for the proposed algorithm were generated (V_{ki} and $\mathbf{c}_{ki}$).
The first experiment shows the convergence of the proposed algorithm. We randomly generated a set of n_k rotation matrices, for which the maximum difference between the ground truth rotations and the randomly generated ones is 45°. We repeated this process 100 times. In Figure.3(a) we see the cost function printed w.r.t. the number of iterations. We see that in all experiments our algorithm converges to the *same* minimum and that the total cost is in 87% below $0.2m^2$ (this equals a mean error of $5cm$) after 10 iterations. In Figure 3(b) we see the convergence of the bundle adjustment method, which optimizes all parameters (the used cost function for the bundle adjustment is the image-space error). We see that in 62 out of 100 experiments, either the convergence fails, or the algorithm converges to a wrong solution. This happens because the initial parameters are too far from the correct values (initial rotation is 45° off), and so the method is trapped in a local minimum.
The second experiment shows the accuracy of the proposed algorithm. In Figure 4(a) we plotted the histogram of the reached accuracy after 60 iterations of our algorithm. The accuracy was measured as the mean distance between the reconstructed points and the ground truth. As a comparison we show in Figure 4(b) the histogram of the reached accuracy of the bundle adjustement algorithm. We see that in only 38 out of 100 experiments the algorithm converges to the *correct* minimum. Only in these 38 cases the initial parameters are close enough to the *correct* solution, to let the *bundle adjustement* converge correctly. The accuracy of these 38 cases is shown in Figure 4(c). The mean accuracy of our proposed algorithm is about $1.5cm$, while the mean accuracy of the bundle adjustement method is about $0.9cm$.



(a) (b) (c)

Figure 4: **Histograms of the reached accuracy:** (a) proposed algorithm (b) bundle adjustment (c) zoomed into correct solutions of bundle adjustment.

7 Conclusion

We presented an iterative structure and motion algorithm which has the following properties:

- It works for general camera models.
- It works on sparse data. (i.e. not all points have to be visible in all cameras).
- We proved the global convergence of the algorithm.
- The time complexity analysis shows a theoretical speedup of a factor of 8 compared to the sparse bundle adjustment method.
- Experiments showed that the algorithm converges in a few iteration steps to a minimum of the cost function.

We have shown that using the *Object Space Error for General Camera Models* it is possible to split the optimization process into two stages. First, the optimal structure and the optimal position of the cameras can be estimated in closed form from the rotational part of the motion. Second, the rotational part of the motion can be optimized using the well known *absolute orientation algorithm*. Doing this, we have reduced the most time intensitive part (inverse of a matrix) by a factor of 8 w.r.t. bundle adjustment. Further it is shown by the experiments (and also by the proof of global convergence) that the proposed algorithm converges from a broader range of initializations than bundle adjustment. This is achieved because fewer parameters are involved in finding a consistent solution.

8 Acknowledgments

This work was supported by the Austrian Science Foundation (FWF, project S9103-N04 and P15748).

References

[1] Michael D. Grossberg and Shree K. Nayar. A general imaging model and a method for finding its parameters. In *ICCV*, pages 1100–1105, 2001.

[2] Hartley and Zisserman. *Multiple View Geometry*. Cambridge University Press, 2000.

[3] Berthold K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4:629–642, April 1987.

[4] C.P. Lu, G.D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *PAMI*, 22(6):610–622, June 2000.

[5] D.G. Luenberger. *Linear and Nonlinear Programming*, chapter 6. second ed. Reading, Mass.:Addison Wesly, 1948.

[6] Robert Pless. Using many cameras as one. In *CVPR '03*, pages 587–593, 2003.

[7] Peter Sturm. Multi-view geometry for general camera models. In *CVPR*, pages 1100–1105, Washington, DC, USA, 2005. IEEE Computer Society.

[8] Bill Triggs, P. McLauchlan, Richard Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.