Error Minimization in 3-Dimensional Model Reconstruction Using Sparse Bundle Adjustment and the Levenberg-Marquardt Algorithm on Stereo Camera Pairs

Luke Bonde, Allison Brumfield, Ye Yuan
Mathematics, Statistics, and Computer Science
Saint Olaf College
Northfield, Minnesota 55057
bonde, brumfield, yuan@stolaf.edu

Abstract

In computer vision and image processing, it is often desirable to reconstruct the 3D geometry of a scene using 2D images. Bundle adjustment is a common optimization algorithm often used as the final step of this type of 3D model reconstruction to simultaneously refine scene and camera parameters using the Levenberg-Marquardt non-linear least squares minimization algorithm to reduce reprojection error. sba, a C/C++ implementation of sparse bundle adjustment, was developed by Lourakis and Argyros for computational effectiveness by taking advantage of the sparseness of matrices involved in the Levenberg-Marquardt computation. We utilize sba to implement bundle adustment for a stereo camera model, in which pairs of cameras are described by both individual and joint parameters. However, sba is not designed for stereo models. This paper highlights the development of a stereo camera model in order in order to interface data collected by stereo camera pairs with sba to obtain optimal results for 3D reconstruction.

1 Introduction

In 3D reconstruction problems, bundle adjustment is most commonly used as a method for optimizing 3D structure and viewing parameters. This type of optimization method finds a solution by simultaneously optimizing both 3D structure and camera parameters by minimizing a cost function that describes the model's fitting error. Exploiting the sparseness of the matrix calculations, sparse bundle adjustment is a more computationally efficient method of calculating this type of minimization. An implementation of SBA was developed by Lourakis and Antonis, called sba [5]. Implemented as a C/C++ package which uses the Levenberg-Marquardt minimization algorithm with a sparse Jacobian, the package solves bundle adjustment problems efficiently for the single camera model.

However, for our specific 3D reconstruction problem, bundle adjustment and sba cannot be simply transfered to stereo camera pairs. The image data set we seek to optimize was acquired by dual cameras, marked as left and right and fixed on a stereo base. Such a camera pair can be modeled as stereo camera with fixed relative position between the optical centers of each camera. To use bundle adjustment method to improve our 3D reconstruction problem, we need to implement a solution using the sba package and an appropriate camera model.

We focus on simultaneously optimizing parameters describing camera pair motions and 3D structures by constructing a stereo camera model capable of interfacing with sba. The basis of our work in defining this model expands upon the stereo camera model published by Kutz et. al. [3]. We consider an additional rotation of the right camera not considered by Kurz et. al. and developing a method for sba to account for the situation when a 3D point is occluded in only one camera in the pair.

This paper focuses on the particulars of our implemented solution of bundle adjustment for a stereo camera pair model. Section 1 introduces background information for this project, including the pinhole camera model, Levenberg-Marquardt algorithm, bundle adjustment, and sba package; Section 2 details our implemented solution; Section 3 shows our results of our implemented minimization with a series of data sets; Section 4 discusses further work and improvements.

1.1 Projective Camera Model

The pinhole camera model formally describes the mathematical relationship between the world coordinates of a 3D point and its 2D projection onto the image plane of an ideal pinhole camera, where the theoretical camera aperture has an infinity small radius. There is a unique camera and respective camera matrix associated with every image of a 3D scene. A general description of the mathematical model is given as follows.

Let

$$K = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

be the *intrinsic camera parameter matrix*, where $\alpha = fm_u, \beta = fm_v$ represent the focal length in pixels. Here, f is the focal length in the camera coordinate system and m_u, m_v are scale factors derived from the image resolution relating pixels to the camera coordinate system. (u_0, v_0) is the principal point of the image, and γ is a skew factor, usually equal to 0. Suppose R is the 3×3 *rotation matrix* relating the orientation of the camera frame of reference to the world frame of reference and T is the 1×3 *translation matrix* representing the location of the camera's optical center with respect to the origin of the world coordinate system. Let I be the 3×3 identity matrix. Define the 3×4 camera matrix

$$C = KR(I|T)$$

which transforms 3D homogeneous coordinates in the world coordinate system to 2D points on the image plane of a camera. Given a homogeneous 3D point, $\mathbf{b}$, and its projection on the image plane, or feature points, $\mathbf{x}$,

$$\mathbf{b} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ and } \mathbf{x} = \begin{bmatrix} u \\ v \\ w \end{bmatrix},$$

the camera matrix C relates the points up to a scalar factor s, such that

$$s\mathbf{x} = C\mathbf{b}$$

Homogeneous coordinates are used to allow translations and rotation of vectors via matrix multiplication. For a more rigorous introduction to the camera matrix and homogeneous coordinates, see [2].

1.2 Bundle Adjustment

Bundle adjustment (BA) is the process of jointly optimizing the location of the 3D points and camera parameters. Given feature points in several camera images of the same 3D scene, BA attempts to resolve discrepancies in camera and 3D point measurements. Lourakis and Argyos claim bundle adjustment is the most widely used optimization technique for a feature point based approach to 3D model reconstruction [5].

A discussion on the bundle adjustment algorithm present in [5] is presented in this paper. Assume n 3D points are visible in m images shown in fig. 1. Assume that all 3D points have an associated feature point in all images. Define $\mathbf{x_{ij}}$ to be the 2D projection of the ith

3D point onto image with associated camera j parameterized by vector $\mathbf{a_{ij}}$ describing the intrinsic and extrinsic camera parameters and the 3D point by vector $\mathbf{b_{ij}}$ describing position. BA attempts to accurately describe the location of the n 3D points and m cameras along with additional optical camera parameters by jointly minimizing the parameters in $\mathbf{a_{ij}}$ and $\mathbf{b_{ij}}$. To accomplish this, BA calculates the reprojection error of each 3D point projected onto each image plane. The reprojection error is the Euclidean distance between the 2D projection $\mathbf{x_{ij}}$ and the position of the feature point implied by $\mathbf{a_{ij}}$ and $\mathbf{b_{ij}}$. BA minimizes the total reprojection error, specifically

$$\min_{a_j, b_i} \sum_{i=1}^{n} \sum_{j=1}^{m} d(Q(\mathbf{a_j}, \mathbf{b_i}), \mathbf{x_{ij}})^2$$

where $Q(\mathbf{a_j}, \mathbf{b_j})$ is the 2D projection of 3D point i onto image j and $d(\mathbf{x}, \mathbf{y})$ is the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. Because of the flexibility in defining $\mathbf{a_{ij}}$, BA easily accommodates any monocular camera model parameters. If κ is the dimension of $\mathbf{a_{ij}}$ and λ the dimension of $\mathbf{b_{ij}}$, the total number of parameters to be minimized is $m\kappa + n\lambda$ and must be considered when scaling the number of parameters, images, and 3D points [5].

1.3 Levenberg-Marquardt Algorithm

Given a system of cameras and 3D points whose parameters not ideal from a lack of precision or accuracy, an optimization algorithm is required to refine these parameters. In particular, the algorithm should reduce the discrepancy between the known location of feature points and their projected locations, known as the reprojection error. Given that the total reprojection error is the sum of the square of distances between feature points and projected points, we require a non-linear least squares algorithm to find a minimum in the reprojection error function. The Gauss-Newton algorithm (GNA) is one such method specially designed to find the minimum sum of squared function values. Gradient descent is also a common method to find the local minimum of a function.

The Levenberg-Marquadt (LM) algorithm was originally proposed by Levenberg in 1944, and later by Marquart in 1963. It is an algorithm used to a local minimum of a multivariable function, which is in the form of the sum of squares of nonlinear real-valued functions. The LM algorithm combines GNA and the method of gradient descent: when the parameters are far from their optimal value, the LM algorithm uses a gradient descent approach; when the parameters are close their optimal value, the LM algorithm refines using the GNA. The LM algorithm is more robust than GNA because it finds a solution even when the parameters are far from optimal. However, LM algorithm can only find local minimum, which is not necessarily the desired global minimum. A brief formal description of the LM algorithm is given.

Let f be a function that maps a parameter vector, $\mathbf{p} \in \mathbb{R}^m$, which contains all the parameters in the model, to a predicted measurement vector $\widehat{\mathbf{x}} \in \mathbb{R}^n$ such that $\widehat{\mathbf{x}} = f(\mathbf{p})$. An initial

estimate for $\mathbf{p}$ and $\widehat{\mathbf{x}}$ are needed to find an optimal estimate for the parameter vector which best satisfied the mapping f, which is achieved by minimizing the squared reprojection error $\epsilon^T \times \epsilon$, where $\epsilon = \mathbf{x} - \widehat{\mathbf{x}}$ for all $\mathbf{p}$ and $\mathbf{x}$. For small values of $\|\delta_\mathbf{p}\|$, f can be approximated by

$$f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p}$$

where $\mathbf{J}$ is the Jacobian matrix of the multivariable function f with respect to $\mathbf{p}$.

This algorithm is an iterative process, starting with $\mathbf{p}_0$ and converging toward a local minimizer for the function f. For each iteration, it finds the value $\delta_\mathbf{p}$ which minimizes the quantity $\|\mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p})\| \approx \|\mathbf{x} - f(\mathbf{p}) - \mathbf{J}\delta_\mathbf{p}\| = \|\epsilon - \mathbf{J}\delta_\mathbf{p}\|$. The minimum can be achieved when $\mathbf{J}\delta_\mathbf{p} - \epsilon$ is orthogonal to the column space of $\mathbf{J}$, which is equivalent to the problem of solving the *normal equations* $\mathbf{J}^T(\mathbf{J}\delta_\mathbf{p} - \epsilon) = \mathbf{0}$, and solve for $\delta_\mathbf{p}$

$$\mathbf{J}^T\mathbf{J}\delta_\mathbf{p} = \mathbf{J}^T\epsilon.$$

The LM method actually solves a variation of the equation above called *the augmented normal equations*

$$\mathbf{N}\delta_\mathbf{p} = \mathbf{J}^T\epsilon, \mathbf{N} \equiv \mathbf{J}^T\mathbf{J} + \mu\mathbf{I},\ \mu > 0$$

where μ is called the damping term. If the value of $\delta_\mathbf{p}$ leads to a smaller value of $\epsilon^T\epsilon$, the update in $\mathbf{p}$ is accepted and the process is repeated with a smaller damping value μ. Such process repeats until a value of $\delta_\mathbf{p}$ which decreases the error is found. Solving the above equation repeats for different decreasing μ until an acceptable value of $\mathbf{p}$ is found related to one iteration in the method [5].

1.4 Sparse Bundle Adjustment

Sparse Bundle Adjustment (SBA) is a variation of BA using the LM algorithm. We provide a brief, informal description of SBA; a formal description can be found in [5].

The standard LM solution is elegant in theory, but computationally slow. As briefly explained in Section 1.3, the LM algorithm involves the solution of linear systems know as the normal equations. These linear systems are repeatedly solved in the LM algorithm and each computation of the solution to a dense linear system has complexity $O(N^3)$ in the number of unknown parameters [5]. From this, it is clear that general purpose LM code implementations are very computationally demanding when used for a large number of parameters. This remains true if other non-linear least squares algorithms other than LM are used. The situation is further complicated by the fact that the size of the Jacobian matrix also increases in size with the number of parameters. Thus, when performing large computations using the Jacobian and a large number of parameters, the code implementation risks thrashing, which is the process of wasting CPU cycles to write and read pages to and

from disk. Fortunately, the normal equations matrix has a sparse block structure due to the lack of interaction among all camera and 3D point parameters other than the camera and 3D point being considered. Thus, significant computational benefits are gained by employing a sparse variant of the LM algorithm which explicitly takes advantage of the normal equations zeros pattern by avoiding storing and operating on zero elements.

1.5 *sba* a C/C++ Package for Sparse Bundle Adjustment

sba is a generic sparse bundle adjustment C/C++ package available under GNU General Public License and the first and only free software released of its kind. It is generic in the sense that the user has full control over the definition of parameters describing the cameras and the 3D structure: In this way *sba* will support any camera model with arbitrary parameters [5]. The user must supply *sba* with an appropriate method of calculating the estimated feature point reprojections and the proper Jacobian for use in the LM minimization. Techniques for calculating the Jacobian include: by hand, Mathematica or similar software, and automatic differentiation techniques such as finite differences.

sba exploits the specific sparseness of the Jacobian in the LM algorithm when applied to bundle adjustment to achieve computational gain. It is capable of handling large reconstruction problems. As noted in the *sba* documentation, with a system of 54 cameras, 5207 3D points, and 24609 feature points resulting in 15999 variables to be minimized, *sba* optimized the system in approximately 7 seconds. This time trial was performed on a machine running Linux with an Intel P4@1.8Ghz with unoptimized BLAS, a package of basic linear algebra subprograms [5].

sba requires LAPACK and BLAS for linear algebra functionality and no other 3rd party libraries. Downloads and installation instructions for Linux/Unix and Windows are available at `http://users.ics.forth.gr/~lourakis/sba/index.html#download`.

2 Sparse Bundle Adjustment of Stereo Camera Pairs

2.1 Previous Work on Stereo Bundle Adjustment

Considering the high demand for tools processing stereoscopic image sequences, especially for the step of camera motion estimation, Kurz et. al. develop a new stereo camera model for bundle adjustment. The stereo camera model was designed to be applicable to a wide range of cameras in today's movie production. They first decomposed the camera matrix

A for a metric camera:

$$A = K \begin{bmatrix} I & | & 0 \end{bmatrix} \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix}$$

where C is the position of camera center, R is the rotation matrix, and K is a calibration matrix. They further apply this decomposed projection model to a standard stereo camera setup. Instead of treating left camera and right camera as separate entities, they design a combined camera model considering the motion shared by both left and right cameras. Assuming the relative position offset is constant, since the baseline between the cameras is not changed, they propose two decompositions for the left and the right camera matrices

$$A_L = K_L \begin{bmatrix} R_L & | & 0 \end{bmatrix} \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix}$$
$$A_R = K_R \begin{bmatrix} R_R & | & -R_R C_R \end{bmatrix} \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix}$$

where subscripts L and R denote parameters that are exclusive to left or right camera [3].

To conduct bundle adjutment, they use LM algorithm with their new stereo camera model treating camera j in sba as representing a stereo pair such that $\vec{a}_j$ hold pair parameters. They validate their stereo camera model by synthetic experiments on rendered sequences of images, and on a variety of real-world video sequences. The results of their synthetic experiments show that their stereo bundle adjustment can process image sequences with more accuracy and efficiency compared to conventional bundle adjustment [3].

2.2 Stereo Camera Model

We consider a stereo camera model based on dual cameras generally mounted on a rig or mechanism. Unlike the monocular projective camera model, certain forces directly influence both cameras of the pair simultaneously: A rotation or translation of the stereo camera pair or rig alters the position and rotation of both cameras. We define a stereo camera pair model that considers parameters that apply to the pair and also those that apply to either the left or right camera individually based on the work of Kurz et. al.

For an accurate representation of a stereo image pair we recommend the use of the following pair parameters

- R_P, the rotation of the stereo pair given by a rotation matrix. Can be equivalently be expressed as Euler angles $\alpha_P, \beta_P, \gamma_P$ or the quaternion q_P.
- $t_P = \{x_P, y_P, z_P\}$, the translation of the pair or rig expressed in the coordinate system of the 3D scene.
- fx_c, fy_c, the focal length of the camera expressed in pixels.

- $u0_c, v0_c$, the principal point of the virtual image for a given camera expressed in pixels. and the individual camera parameters, subscripted $c \in \{L, R\}$ to specify the individual parameters for the left and right camera respectively.

  - $t_c = \{x_c, y_c, z_c\}$, translation of the cameras optical center specifying the cameras placement on the rig. We define the optical center of the left camera to be the measurement center of the stereo pair such that $t_L = \{0, 0, 0\}$ and t_R is the translation of the right camera relative to the left.
  - R_c, the rotation matrix for the orientation of the camera. Or equivalently, the Euler angles $\alpha_c, \beta_c, \gamma_c$ or the quaternion q_{AR}. Again, the orientation of the left camera can be defined as the default orientation of the stereo pair such that $\alpha_c = 0, \beta_c = 0, \gamma_c = 0$. The right camera then has a relative rotation to the left camera.
  - s, a scale factor that has the ability to distort the focal length of the right camera an additional amount, which accounts for any variation in the focus of the right camera.

- R_{AR}, a rotation matrix for the additional rotation of the right camera relative to the left camera. This additional roation allows for a slight displacement of the right camera when surveyors were capturing images. Or equivalently, the Euler angles $\alpha_{AR}, \beta_{AR}, \gamma_{AR}$ or the quaternion q_{AR}.

The camera calibration matrix is defined in the same way for the left camera as in the monocular camera model to map the camera coordinate system to the pixel coordinate system of the virtual image seen by that camera.

$$K_L = \begin{bmatrix} fx_L & 0 & u0_L \\ 0 & fy_L & v0_L \\ 0 & 0 & 1 \end{bmatrix}$$

The additional scale factor is included with the focal length such that the camera calibration matrix for the right camera is adjusted as follows:

$$K_R = \begin{bmatrix} s * fx_R & 0 & u0_R \\ 0 & s * fy_R & v0_R \\ 0 & 0 & 1 \end{bmatrix}$$

For each camera we also define a 3×4 projective matrix which maps the homogeneous coordinate of a 3D point in the 3D world coordinate system into the camera coordinate system for each camera.

$$A_L = \begin{bmatrix} R_L R_P & | & -R_L R_P t_P + t_L \end{bmatrix}$$

$$A_R = \left[\begin{array}{c|c} R_R R_P R_{AR} & R_R R_P R_{AR}(-t_P + R_P^T R_R^T t_R) \end{array} \right]$$

The rotations are applied in the following order: any additional rotation of the right camera that does not move the optical center, the rotation of the pair, and finally the rotation of the specific camera on the rig. The use of all three rotations allows for minute adjustment of specific rotations. The pair translation t_P, must be rotated into the camera coordinate system using the composed rotation before being applied for each camera.

The projection of a 3D point b_i to feature point $p_{i,k,x}$ onto the virtual image plane of the x camera in stereo pair k is given by the matrix equation where $x \in \{L, R\}$

$$p_{i,k,x} = K_x A_x \left[\begin{array}{c} b_i \\ 1 \end{array} \right]$$

2.3 Bundle Adjustment and Stereo Camera Pairs

Define a set of K stereo camera pairs with associated images $I_{k,x}$ and projection matrices $A_{k,x}$ for $1 \leq k \leq K and x \in \{L, R\}$ which view n 3D points p_i and feature points $p_{i,k,x}$ for $1 \leq i \leq n$. Bundle adjustment for stereo camera pairs is defined as

$$\min \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{x} d(p_{i,k,x}, A_{k,x} P_i)^2$$

which demonstrates the extension of bundle adjustment to stereo camera pairs mathematically [2].

2.4 sba and Stereo Camera Pairs

As mentioned earlier, the sba package created by Lourakis is intended to optimize systems of cameras in which each camera can move independently of each other camera (monocular vision). However, with stereo vision, cameras are translated as pairs, and therefore an implementation of the optimization for a stereo model should account for mathematical relationship between the left and the right camera of a pair. The sba package is flexible enough to allow for this variation.

Considering potential implementations of the interface between the previous stereo model and the sba package, one potential intuitive solution is to associate a parameter vector a_j for each camera j in the system as with the monocular model. However, since pairs of cameras share 3 rotation and 3 translation parameters, the left and right camera parameter

would have 6 duplicate parameters which must always be equivalent. This constraint would be difficult to implement without modifying the sba package to not alter one duplicate parameter without changing the other. Given that we want to allow the right camera of any pair to have additional rotation parameters, this would mean the size of parameter vectors for right and left cameras would be theoretically different and would also require special compensation within sba. Therefore, we consider another solution.

Our proposed solution is to use the parameter vector a_k to represent all of the variable parameters for a pair k of cameras. This implementation does not include duplicate parameters, nor does it theoretically require camera parameter vectors of varying size. The visibility mask is defined as a vector indicating which camera pairs can see which 3D points and $Q(a_k, b_i) = \hat{x}_{ik}$ is defined as the projection of 3D point i onto both cameras in a pair k. Thus the number parameters for our measurement vector is $|x_{ik}| = 4$; one feature point corresponding to each camera in the pair. For the visibility mask, we say that a pair k can see a 3D point i if at least one of the cameras of the pair k has a feature point projected from the 3D point i. In this case, the corresponding i, k entry in the visibility mask is set to 1.

However, we are required to consider the case in which the left camera of a pair can see a 3D point, but the right camera cannot (or vice versa). In the previous case, we set the missing feature point of the right camera to some default value, such as -1. Thus, in the case that left camera of a pair k can see 3D point i, but the right camera cannot, we set the corresponding measurement vector to $\mathbf{x}_{i,k} = \{x_L, y_L, -1, -1\}$. This structure for $\mathbf{x}_{i,k}$ creates a problem for our projective function Q. The problem is non-existent with monocular vision, as the visibility mask alone was sufficient to determine whether the projection function should be called for a given camera, but in the stereo model, the visibility mask is not sufficient because it does not indicate whether one or both of the cameras can see that 3D point. To solve this problem, we create a C++ object to keep track of special cases in which one camera of a pair cannot see a 3D point, but the pair itself can. This object is passed into the projective function, which returns -1, -1 for the entry in $\hat{\mathbf{x}}$ for the camera in the pair with the occluded 3D point. Since both $\mathbf{x}$ and $\hat{\mathbf{x}}$ to have the same default value for missing feature points, it will have no effect on error calculation. The function to compute the Jacobian similarly accounts for these special cases, and the entries in the Jacobian for the corresponding 3D point and camera are explicitly set to 0 to indicate no relationship.

2.5 Lens Distortion Model

Although the pinhole camera model provides a good mathematical model for describing the aperture of practical camera, its simplification of the aperture as an infinitesimally small point removes the effects of lens distortion. For a more accurate model of our problem, we account for lens distortion in our bundle adjustment process. For a 2D point on the image,

lens distortion is expressed as

$$u_d = u + h_u(u, v)$$
$$v_d = v + h_v(u, v)$$

where u and v are distortion-free image coordinates, u_d and v_d are corresponding distorted coordinates. Radial distortion is caused by the curvature of the lens and tangential distortion occurs when lenses are not perfectly parallel to the image plane. The Brown-Conrady model is a very popular model, which uses power series to describe both radial and tangential distortion. Thus distorted image points are

$$u_d = u(1 + K_1 r^2 + K_2 r^4 + ...) + P_1(r^2 + 2u^2 + 2P_2 uv)$$
$$v_d = v(1 + K_1 r^2 + K_2 r^4 + ...) + P_1(r^2 + 2v^2 + 2P_2 uv)$$

where r is the distance from principal point (u_0, v_0) to the given distortion-free point (u, v), K_i is a radial distortion parameter and P_i is a tangential distortion parameter. Since the first two terms r^2 and r^4 are predominant, we include two parameters to describe each distortion in our model.

For our convenience, we utilize pre-existing functions to correct the lens distortion at the beginning of bundle adjustment.The distortion-correcting function takes distortion parameters from the input file, estimates undistortion parameters with Levenberg-Marquardt algorithm and uses the estimated parameters to correct the distorted points. Since the pre-existing functions calculate with units in meters, we need to convert our feature points from pixels to meters

$$(u_{meter}, v_{meter}) = (u_{pixel}, v_{pixel})/(f_u, f_v) - (u_0, v_0)$$
$$(u_{pixel}, v_{pixel}) = ((u_{meter}, v_{meter}) + (u_0, v_0))(f_u, f_v)$$

where (f_u, f_v) is the focal length and (u_0, v_0) is the principal points. We chose not to optimize the lens distortion parameters for each camera in order to reduce the size of our parameter matrix $\mathbf{p}$ and the number of feature points required for optimization, and as a result decided to undistort our feature points initially with the method described above and store the undistorted values in the measurement $\mathbf{x}$ before computing the bundle adjustment.

3 Results

We fabricated a perfect data set with 4 pairs of cameras, 12 3D points, and 75 feature points using $Mathematica$ to accurately determine the feature point location for a defined set of cameras and 3D points. To generate a baseline data set on which to test our minimization implementation, we used Mathematica to calculate the location of feature points in our synthetic images. We defined 12 points in 3D space along with four camera pairs rotated to

view some or all of the 3D points. We had 10 variable parameters for each pair of cameras: 3 for translation, 3 for rotation, 3 for additional rotation applied to the right camera, and a scale factor. Using the projection function defined in Section 2.1 for use with our stereo camera model, we used Mathematica to calculate the feature points of the 3D points for each virtual image of the 8 cameras. The mean reprojection error (calculated by finding the mean Euclidean distance of feature points and their reprojections) on this data set was .003 pixels before running the optimization. After optimization, the mean reprojection error was reduced to .0028, which indicates that our optimization program does not move the local minimum once one is found. This data set required virtually no computation time to optimize.

Next, we generated an imperfect data set by reducing the precision of camera and 3D points parameters in our perfect data set while the locations of the feature points were left unchanged. This would indicate the scenario in which the measurements of the known variables are precise. The initial reprojection error for this data set was 45.88 pixels, and our optimization program decreased the reprojection error to .0028 pixels, the same as it was in the perfect data set. However, the locations of the cameras and 3D points differ slightly from the perfect data set, which indicate that there are multiple local minima in the error function. This data set also required virtually no processing time.

We applied our optimization to a real data set with 71 pairs of cameras, 64 3D points, and 948 feature points. It is important to note that the known measurements (the feature points) were imperfect in this data from lens distortion and human measurement error. To correct for the lens distortion, we use the Brown-Conrady model with three terms to approximate the radial distortion component. This was implemented as described in Section 2.4. The initial reprojection error in this data set was 4.95. After sba optimization, which took 13.14 seconds, the reprojection error decreased to 3.92, which is a 20.8% decrease. It is important to note that this data had also already been optimized by a LM minimization algorithm. This indicates that our optimization improved upon a previously optimized data set.

To simulate more realistic conditions, we reset the positions of the camera pairs from the previous data set to be their (pre-optimized) estimated positions. Also, we rounded the Euler angles of the pair to the nearest $k * \frac{\pi}{4}$ for all integers k, set the additional rotation parameters (Euler angles) to 0, and set the scale factor to 1. This situation represents a scenario in which camera pair parameters are estimated. The initial reprojection error for this set was 143.4 pixels, and the optimization reduced this value to 3.92 pixels in 13.60 seconds.

Finally, to mimic realistic data we tested our implementation on a data set with both altered locations and orientations of the cameras as above and also truncated locations of 3D points to the nearest tenth of a meter. The initial reprojection error for this set was 142.29 pixels, and the optimization reduced this value to 3.92 pixels. The process took 12.64 seconds.

A summary table of the initial and resulting mean projection error and timings can be found

in fig. 3.

4 Conclusion and Future Work

Both the effectiveness of the optimization and the speed of the computation indicate that our design has practical applications to 3D reconstruction. However, there some improvements that can be made to our design.

First, there is an opportunity to improve the Jacobian evaluation. For the projection function we define for the stereo camera model of our implementation, we use Mathematica to analytically calculate the Jacobian required by *sba*. With the large number of parameters and terms in this projection function the Jacobian has approximately 50,000 terms in the matrix. Improvements in calculations using the Jacobian by *sba* can be achieved by simplifying the equations used to describe the Jacobian of a specific camera-point combination to group like terms and computations. Also, parallelism can be applied to the execution of the Jacobian function to improve computation speed. Lourakis suggests use with Parallel Linear Algebra for Scalable Multi-core Architecture (PLASMA) [5]. This example of data parallelism could potentially be optimized by other parallelism packages as well.

Second, the design of our interface between sba and the stereo model implements an additional visibility mask to account for left-right visibility issues, as mentioned in Section 2.3. Therefore, it would be desirable to refine the design such that the additional visibility mask is not required, thus removing redundant computation even though this involves changing how the projective and Jacobian functions are executed by sba.

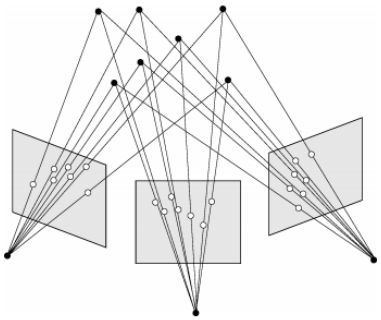

Figure 1: Schematic illustration of $n = 7$ 3D points projecting onto $m = 3$ images used in bundle adjustment [5].

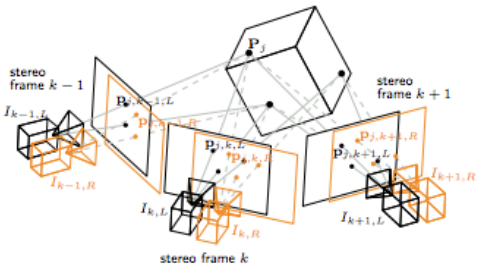

Figure 2: Stereo camera pair visualization of 3D cube from 3 separate pair translations and rotations. The relative position and orientation of the camera pairs remains constant. Solid lines are the projection of the 3D points p_j onto the left stereo camera and dashed lines are the projection onto the right stereo camera [3].

Data Set	**Initial MRE**	**Final MRE**
Small, accurate	0.00302	0.00286
Small, approximate	45.8836	0.00286
Large, optimized	4.94586	3.91972
Large, approx. cameras	143.412	3.91972
Large, approx. cameras and 3D points	142.291	3.91972

Figure 3: Mean Reprojection Error (MRE) results of stereo camera pair SBA minimization on a series of data sets ranging in size and 3D point and camera location accuracy. The small files have precise feature point location whereas the larger data sets have imperfect feature point measurements.

References

[1] H. Gavin, "The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems", Department of Civil and Environmental Engineering, Duke University, September 24, 2013. [Online] Available: http://people.duke.edu/~hpgavin/ce281/lm.pdf

[2] P. Hillman,"White paper: Camera calibration and stereo vision, Square Eyes Software, October 2005". [Online] Available: http://www.peterhillman.org.uk/downloads/whitepapers/calibr ation.pdf

[3] C. Kurz, T. Thormahlen, and H. Seidel, "Bundle Adjustment for Stereoscopic 3D", Max Planck Institute for Computer Science (MPII), 2011. [Online] Available: http://www.mpi-inf.mpg.de/~ckurz/papers/Kurz_ MIRAGE2011.pdf

[4] M. Lourakis, "A brief description of the levenberg-marquardt algorithm implemented by levmar", Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH), February 11, 2005. [Online]. Available: http://users.ics.forth.gr/~ lourakis/ levmar/levmar.pdf

[5] M. Lourakis and A. A. Argyros, "SBA: a software package for generic sparse bundle adjustment", ACM Trans. Math. Softw. 36, 1, Article 2 (March 2009). [Online] Available: http://users.ics.forth.gr/~lourakis/sba/sba-toms.pdf

[6] A. Majumder, "The Pinhole Camera", University of Califormia, Irvine, Feb. 2010. [Online] Available: http://www.ics.uci.edu/~majumder/vispercep/cameracalib.pdf

[7] L. Ming, "Correspondence analysis between the image formation pipelines of graphics and vision,"Max-Planck-Institute for Computer Science. [Online] Available: http://campar.in.tum.de/twiki/pub/Chair/TeachingWs05ComputerVision/CorrespondenceAnalysisBetweenTheImageFormationPipelinesOfGraphicsAndVision_Ming.pdf

[8] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, "Bundle adjustment a modern synthesis" Springer Berlin Heidelberg, Jan. 2000. [Online] Available: http://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Triggs 00.pdf

[9] Z. Zhang, "A flexible new technique for camera calibration", IEEE Trans. Pattern Anal. Mach. Intell., vol.22, no.11, pp.1330 1334, Nov. 2000. [Online]. Available: http://dx.doi.org/10.1109/34.888718