LUND UNIVERSITY

Efficient Structure and Motion: Path Planning, Uncertainty and Sparsity

Haner, Sebastian

2012

Link to publication

Citation for published version (APA):
Haner, S. (2012). *Efficient Structure and Motion: Path Planning, Uncertainty and Sparsity*. [Licentiate Thesis, Mathematics (Faculty of Engineering)]. Centre of Mathematical Sciences.

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

EFFICIENT STRUCTURE AND MOTION: PATH PLANNING, UNCERTAINTY AND SPARSITY

SEBASTIAN HANER

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Licentiate Theses in Mathematical Sciences 2012:2
ISSN 1404-028X

ISBN 978-91-7473-371-6
LUTFMA-2034-2012

© Sebastian Haner, 2012

Printed in Sweden by MediaTryck, Lund 2012

Preface

This thesis is based on the following papers:

- S. Haner and A. Heyden, "A Step Towards Self-calibration in SLAM: Weakly Calibrated On-line Structure and Motion Estimation", *Workshop on Mobile Vision*, San Francisco, 2010.
- S. Haner and A. Heyden, "Optimal View Path Planning for Visual SLAM", *Scandinavian Conference on Image Analysis*, Ystad, 2011.
- S. Haner and A. Heyden, "Covariance Propagation and Next Best View Planning for 3D Reconstruction", *European Conference on Computer Vision*, Florence, 2012.
- P. Piniés, L. M. Paz, S. Haner and A. Heyden, "Decomposable Bundle Adjustment using a Junction Tree", *International Conference on Robotics and Automation*, Minneapolis, 2012.

Acknowledgements

I would like to thank all my colleagues at the Centre for Mathematical Sciences for providing a most agreeable working environment, and especially all the Ph.D. students at the department for their help, inspiration and friendship during my time here. In particular, I am grateful to Johan Richter for sharing his knowledge of mathematics, life and everything, to my supervisor Anders Heyden for his positive attitude and support, and to Carl Olsson, for generously dispensing his infinite wisdoms.

Contents

Preface iii

Acknowledgements v

Introduction 1

1 Pinhole Camera Model 1
2 Maximum Likelihood Estimation 3
3 Triangulation 4
4 Camera Resectioning 4
5 Bundle Adjustment 6
    5.1 Robust cost functions 8
6 Covariance Estimation 10
References 13

A A Step Towards Self-calibration in SLAM: Weakly Calibrated On-line Structure and Motion Estimation 15

1 Introduction 17
2 Problem Formulation 18
3 The Parametrization 20
4 The Extended Kalman Filter 21
5 Experiments 23
6 Conclusions 26
References 28

B Optimal View Path Planning for Visual SLAM 31

1 Introduction 33

2 Problem Formulation 35
3 Cost Function 36
3.1 Calculating Covariance 37
3.2 Cost Function 38
3.3 Cost Function Properties 39
4 Proposed Algorithm 40
5 Experiments 41
6 Discussion 41
6.1 Computational Complexity 41
6.2 Extensions 44
7 Conclusion 45
References 46

C Covariance Propagation and Next Best View Planning for 3D Reconstruction 49
1 Introduction 51
2 Estimation from Uncertain Geometry 52
2.1 Pose Estimation 52
2.2 Triangulation 54
2.3 Complexity 55
3 Covariance Propagation 55
3.1 Selecting the Seed 55
3.2 Fixing the Gauge 56
4 Next Best View Planning 57
5 Reconstruction Pipeline 57
6 Experiments 58
7 Conclusion 66
References 66

D Decomposable Bundle Adjustment using a Junction Tree 69
1 Introduction 71
2 Decomposable Systems 73
3 Decomposable Structure of Bundle Adjustment 74
4 Decomposable Bundle Adjustment using a Junction Tree 76
4.1 Building a Junction Tree 78
4.2 Collect Evidence 79
4.3 Distribute Evidence 80

5 Results 80
5.1 Running Time Evaluation on Synthetic Experiments . . 81
5.2 Evaluation on Real Data 83
6 Parallelization 84
7 Conclusions 85
References 86

Introduction

This work is concerned with the so-called *structure-and-motion* problem in computer vision. Given two or more two-dimensional images of a scene, the task is to compute the three-dimensional shape of the scene (the structure) and the position and orientation of the camera when taking the images (the motion). This introduction will very briefly cover the basics of multiple-view geometry and reconstruction, and some of the standard algorithms and results used in the included papers.

1 Pinhole Camera Model

In this work, as in most of the computer vision literature, the camera is modeled as an old-fashioned pinhole camera, where light from the scene passes through a focal point (the camera center) and falls on the image plane, forming an inverted image. Geometrically, undoing this inversion is equivalent to placing the image plane in front of the camera center (see Figure 1). If the camera center is at the origin and the optical axis is aligned with the z-axis of the coordinate system, the image projection of the point X can be computed as $x = (fX_1/X_3, fX_2/X_3)^\top$. In the general case, with camera center position C and orientation R, the scene point must first be transformed into the camera coordinate system, $X^{\mathrm{c}} = R(X - C)$. It is convenient to express x and X in *homogeneous coordinates*, where an extra dimension is added to the representation, so that e.g. $X = (X_1, X_2, X_3, 1)^\top$. Each point is now represented by an equivalence class where $X \sim \bar{X}$ if $X = \lambda\bar{X}$ for some $\lambda \neq 0$. With this convention, point projection can be described by the linear equation

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = KPX = \begin{pmatrix} f & \alpha & u \\ 0 & \gamma f & v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & -RC \end{pmatrix} X\,. \tag{1}$$

The actual projection is then obtained from the representative of x with third coordinate equal to one, i.e. $(x_1/x_3, x_2/x_3, 1)^\top$. The matrix K encodes the *intrinsic* parameters of the camera that map the projected point to actual pixel coordinates. The aspect-ratio parameter γ allows for non-square pixels, the skew parameter α for non-perpendicular image coordinate axes, and the offset (u, v) accounts for the fact that the pixel coordinate system origin is at a corner of the image and not at the principal point. In most applications we can safely assume $\gamma = 1$ and $\alpha = 0$, and that the principal point is in the center of the image.

Real camera lenses also exhibit varying degrees of geometric distortion. In the papers that follow, it will be assumed that such distortions have been compensated for in the input to the algorithms. Except for in Paper A, it will also be assumed that all the intrinsic parameters f, γ, α, u and v are known; when this is the case, we say the camera is *calibrated*, and it is convenient to work with the normalized image coordinates $K^{-1}x$. We shall then simply assume the projection model $x = PX$.

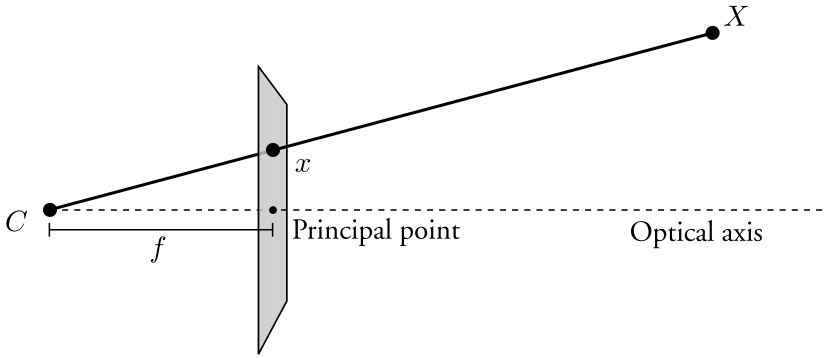


Figure 1: Pinhole camera model.

2 Maximum Likelihood Estimation

Solving structure-and-motion is an inverse problem where the loss of depth information in projections from 3D to 2D needs to be overcome. The problem is complicated further by the fact that image measurements are never exact, but exhibit a certain degree of noise. Given a probabilistic model of this noise, it can be argued that the best solution to the problem is the structure and motion which maximizes the probability of observing the measured data, the *maximum likelihood estimate*. Encoding the structure and motion in a parameter vector θ, the solution we seek is

$$\theta^* = \arg\max_{\theta} \ \mathcal{L}(\theta)\,, \tag{2}$$

where

$$\mathcal{L}(\theta) = \mathcal{L}(\theta \,|\, \tilde{x}) = p(\tilde{x} \,|\, \theta) \tag{3}$$

is the likelihood function and $\tilde{x}$ a vector of measured image projections. If $\Pi(\theta)$ gives the expected noise-free projections given camera and point parameters in θ, the *reprojection error* may be defined as $r(\theta) = \tilde{x} - \Pi(\theta)$ and $p(\tilde{x} \,|\, \theta) = D\big(r(\theta)\big)$ where D is the probability density function of the noise distribution. In the case of zero-mean Gaussian noise, $D = \mathcal{N}(\mathbf{0}, \Sigma)$, the likelihood function becomes

$$\mathcal{L}(\theta) \propto e^{-\frac{1}{2} r(\theta)^\top \Sigma^{-1} r(\theta)}\,, \tag{4}$$

and equivalently, taking the logarithm,

$$\theta^* = \arg\min_{\theta} \ r(\theta)^\top \Sigma^{-1} r(\theta)\,. \tag{5}$$

Under the common assumption $\Sigma = \sigma^2 I$, this reduces further to just minimizing the sum of squares of the reprojection errors, i.e. a quadratic cost function. For other noise distributions, corresponding cost functions are similarly derived.

Unfortunately, finding the maximum likelihood solution is in general difficult, and often structure-and-motion must be solved as a sequence of subproblems. These consist of triangulation, resectioning and bundle adjustment, which we describe below.

3 Triangulation

Triangulation is the problem of finding the scene point X given its projection in two or more images and the corresponding camera poses. If the camera parameters and image projections are known precisely, the rays passing through camera centers $C^{(i)}$ and corresponding image points $x^{(i)}$ intersect in a single point. If there is noise in the image measurements, the rays will most likely not intersect, but we may try to find a point X that best explains the observations, a maximum likelihood estimate. As shown in Section 2, this involves minimizing the distance between the computed projection of X in the images and the measured values $x^{(i)}$, a non-linear function with many local optima. In general this can only be accomplished using iterative optimization techniques. To find an initial estimate to start the iterative algorithm, or in the noiseless case, one may use a simple linear solution. In homogeneous coordinates we have $x^{(i)} = P^{(i)}X$, $i = 1, \ldots, n$, which translates to the projections

$$\begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \end{pmatrix} = \frac{1}{P_3^{(i)}X} \begin{pmatrix} P_1^{(i)}X \\ P_2^{(i)}X \end{pmatrix} \tag{6}$$

where the subscript indicates the row of the vector or matrix. Each image gives two linear equations in X,

$$\begin{pmatrix} x_1^{(i)}P_3^{(i)} - P_1^{(i)} \\ x_2^{(i)}P_3^{(i)} - P_2^{(i)} \end{pmatrix} X = \mathbf{0}\,, \tag{7}$$

which may be solved in a least-squares sense using the singular value decomposition. This method is fast and handles an arbitrary number of views, but it minimizes an algebraic error rather than the geometric reprojection error, which may produce poor results. For the case of only two or three views there exist methods which can compute the maximum likelihood triangulation under the assumption of Gaussian measurement noise, essentially in closed form [2, 5, 1]. In Paper C we revisit the triangulation problem and show how to deal with uncertainty not only in the image measurements but also in the camera parameters.

4 Camera Resectioning

Camera resectioning is the problem of determining the camera parameters given scene points $X^{(i)}$ and corresponding image projections $x^{(i)}$. If the intrinsic pa-

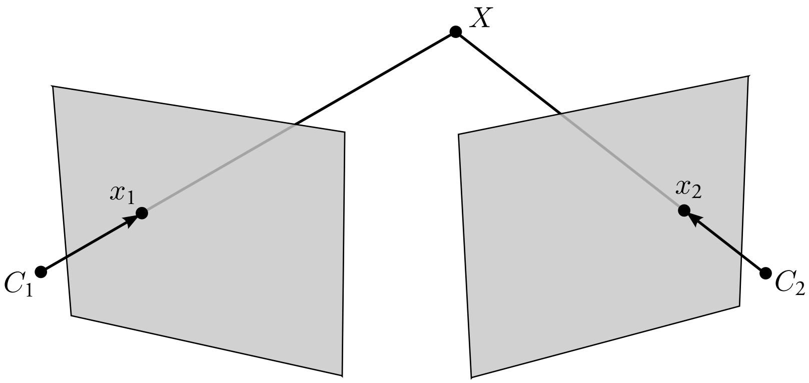


Figure 2: The principle of triangulation.

rameters are known, this is often referred to as pose estimation since only the camera translation and orientation need to be recovered. In the most general case, we seek the 3-by-4 camera matrix M best explaining the observations. Using (6), each projection $x^{(i)}$ gives two linear equations in the components of M,

$$\begin{pmatrix} X^{(i)\top} & 0 & 0 & 0 & x_1^{(i)} X^{(i)\top} \\ 0 & 0 & 0 & X^{(i)\top} & x_2^{(i)} X^{(i)\top} \end{pmatrix} \begin{pmatrix} M_{11} \\ M_{12} \\ \vdots \\ M_{34} \end{pmatrix} = \mathbf{0}\,. \tag{8}$$

With 11 or more equations, i.e. at least six point projections, the camera matrix may be uniquely determined (up to scale). It may then be factored into $M = KP$ using RQ or Cholesky factorization, separating the intrinsic and pose parameters. If the intrinsic parameters are known, methods exist that allow pose estimation from three point projections, with up to four possible solutions; the correct one can be chosen by considering additional points. The linear method is again simple, but minimizes an algebraic rather than a physically meaningful error. It also does not enforce the physical constraints that the observed scene points should all be in front of the camera, a problem which arises frequently in practice using this method.

As shown in Section 2, the assumption of Gaussian measurement noise implies that the maximum likelihood solution is obtained by minimizing the sum of squared reprojection errors. If we instead content ourselves with minimizing the maximum error, camera resectioning may be solved as a sequence of linear feasibility problems [4]. Choose an $\epsilon > 0$, and require the vertical and horizontal components of all reprojection errors to be smaller,

$$\left| x_1^{(i)} - \frac{M_1 X^{(i)}}{M_3 X^{(i)}} \right| < \epsilon \,, \tag{9}$$

$$\left| x_2^{(i)} - \frac{M_2 X^{(i)}}{M_3 X^{(i)}} \right| < \epsilon \,. \tag{10}$$

These constraints may be formulated as

$$\begin{pmatrix} -X^{(i)\top} & 0 & 0 & 0 & 0 & (x_1^{(i)} - \epsilon) X^{(i)\top} \\ X^{(i)\top} & 0 & 0 & 0 & 0 & -(x_1^{(i)} + \epsilon) X^{(i)\top} \\ 0 & 0 & 0 & 0 & -X^{(i)\top} & (x_2^{(i)} - \epsilon) X^{(i)\top} \\ 0 & 0 & 0 & 0 & X^{(i)\top} & -(x_2^{(i)} + \epsilon) X^{(i)\top} \end{pmatrix} \begin{pmatrix} M_{11} \\ M_{12} \\ \vdots \\ M_{34} \end{pmatrix} < \mathbf{0} \,. \tag{11}$$

To ensure that all points end up in front of the camera, the additional constraints $M_3 X^{(i)} > 0$ may be added. The task is then to find the smallest ϵ admitting a solution, which can be accomplished through bisection. Note that this formulation minimizes the maximum component-wise reprojection error. To use the Euclidean error instead, the linear inequalities can be replaced by second order cone constraints, at increased computational cost. Since a number of convex programs must be solved, the above approach is more costly than the direct linear algorithm, but usually gives better results since it minimizes a geometrically meaningful error. The exception is when there are outliers in the input data, i.e. grossly incorrect image measurements, which the max norm is highly sensitive to. The output of the above algorithms is usually used as the starting point for nonlinear optimization to find a maximum likelihood estimate. In Paper C we study the pose estimation problem with uncertainty also in the observed 3D structure, in analogy with the triangulation case.

5 Bundle Adjustment

To get the best reconstruction possible for the given input data, the statistically optimal maximum likelihood solution is preferred. As above, this comes down to minimizing the reprojection errors, but now over all images and all camera and point parameters simultaneously. As already mentioned, this is a difficult problem, and finding a globally optimal solution is generally intractable. Given a good enough initial estimate, however, local optimization methods often find solutions at or near the global optimum. Optimization over all parameters is known as *bundle adjustment*, and the most popular method by far is Levenberg-Marquardt (LM) iteration, a slight variation of the Gauss-Newton method. Define the reprojection error function $r\colon \mathbb{R}^p \to \mathbb{R}^m$ mapping the p camera and point parameters θ to m reprojection errors, stacked in a vector. The Gauss-Newton algorithm attempts to minimize the sum of squared errors $s(\theta) = \sum_{i=1}^{m} r_i(\theta)^2$, which again is suitable under the assumption of Gaussian measurement noise. Taylor expansion gives

$$s(\theta + \delta\theta) \approx s(\theta) + (\nabla s)^\top \delta\theta + \frac{1}{2}\delta\theta^\top H \delta\theta\,, \tag{12}$$

a quadratic approximation of the function around the current estimate θ. The gradient has the special form

$$\nabla s = 2\left(\sum_{k=1}^{m} \frac{\partial r_k}{\partial \theta_1} r_k, \dots, \sum_{k=1}^{m} \frac{\partial r_k}{\partial \theta_p} r_k\right) = 2r^\top J\,, \tag{13}$$

where J is the Jacobian of $r(\theta)$. Similarly, the Hessian matrix H has elements

$$H_{ij} = 2\sum_{k=1}^{m} \frac{\partial r_k}{\partial \theta_i} \cdot \frac{\partial r_k}{\partial \theta_j} + 2\sum_{k=1}^{m} \frac{\partial^2 r_k}{\partial \theta_i \partial \theta_j} \cdot r_k\,. \tag{14}$$

Assuming that the second order derivatives are bounded and the errors r_k small near the minimum, the second term is dropped and $H \approx 2J^\top J$. To minimize s, the approximation (12) is differentiated with respect to $\delta\theta$ and equated to zero, giving $J^\top J \delta\theta = -J^\top r$. In each iteration, this linear system may be solved for the update step $\delta\theta$. In the LM algorithm, one instead solves

$$(J^\top J + \lambda I)\delta\theta = -J^\top r\,. \tag{15}$$

As $\lambda \to \infty$, the solution approaches pure gradient descent. Far from the minimum, the quadratic local approximation of Gauss-Newton may be a poor fit to the actual cost surface, and moving to its minimum could actually increase the reprojection error. If so, λ is increased, giving a solution closer to the gradient descent step. As one approaches the minimum, λ may be decreased to take advantage of the quadratic convergence of Gauss-Newton. Augmenting the diagonal of the matrix also ensures it is positive definite so that a unique solution exists.

The main computational costs of LM are computing the partial derivatives in J and solving the system (15). Due to the special structure of bundle adjustment problems, the Hessian has regular and easily predictable sparsity patterns which can be exploited allowing the solution of very large problems with millions of parameters.

If the variables are partitioned into camera and point parameters so that $\theta = (\theta_c, \theta_p)^\top$, $J = (J_c \;\; J_p)$ and $r = (r_c, r_p)^\top$ are similarly partitioned and the LM system to solve can be written

$$\begin{pmatrix} U & W \\ W^\top & V \end{pmatrix} \begin{pmatrix} \delta\theta_c \\ \delta\theta_p \end{pmatrix} = \begin{pmatrix} -J_c^\top r_c \\ -J_p^\top r_p \end{pmatrix} \tag{16}$$

where $U = J_c^\top J_c$ and $V = J_p^\top J_p$ are block diagonal with one block per camera and point respectively. Block W_{ij} is only non-zero if camera i observes point j. Eliminating θ_p from the top row using the Schur complement of V allows us to solve for the camera parameters separately,

$$\begin{pmatrix} U - WV^{-1}W^\top & 0 \\ W^\top & V \end{pmatrix} \begin{pmatrix} \delta\theta_c \\ \delta\theta_p \end{pmatrix} = \begin{pmatrix} -J_c^\top r_c + WV^{-1}J_p^\top r_p \\ -J_p^\top r_p \end{pmatrix}. \tag{17}$$

After $\delta\theta_c$ is found, the point parameters can be solved for using back substitution. The dominating cost is forming the Schur complement $S = U - WV^{-1}W^\top$ and solving for the camera parameters. Since V is block diagonal it can be inverted in time linear in the number of points, and the back substitution operation is orders of magnitude cheaper. S has a particular block sparsity pattern, where block S_{ij} is non-zero only if camera i and j observe a common point. Thus, depending on the scene geometry, the matrix exhibits varying degrees of sparsity. Real datasets are typically sparse enough that it is much more efficient solving the system using sparse Cholesky factorization or conjugate gradient algorithms than by dense factorization. Paper D presents an alternative method of exploiting the sparsity pattern of S to accelerate and potentially parallelize the computations.

5.1 Robust cost functions

Since the LM algorithm minimizes the sum of squared reprojection errors, it maximizes the reconstruction likelihood assuming normally distributed errors. Gaussian noise is a good approximation for the small measurement errors introduced by the limited resolution of the input images and imprecise feature detection algorithms. It is, however, not a good model for the large errors which sometimes occur due to incorrect matching of features between images. Such outlier data will severely skew the results because of the high cost they incur under the quadratic penalty function. To deal with outliers, robust cost functions giving less importance to large errors are required. A popular choice is the Huber function, defined as

$$C(x) = \begin{cases} x^2 & \text{if } |x| < \beta \\ 2\beta|x| - \beta^2 & \text{otherwise}\,. \end{cases} \tag{18}$$

It assigns quadratic cost to small errors and linear to large, and is convex which ensures it does not give rise to new local minima. The influence of outliers is greatly reduced, so that the result of the bundle adjustment comes close to the ML estimate for the outlier-free data. The outliers may then be detected and removed. To incorporate a robust cost function C into the LM algorithm, iterative reweighting is usually employed. By scaling each component of the vector r by the weight $w_i = \sqrt{C(r_i)}/|r_i|$, the sum of squared errors $s = \sum_{i=1}^{m}(w_i r_i)^2 = \sum_{i=1}^{m} C(r_i)$ assumes the desired value. The step equation (15) is modified to

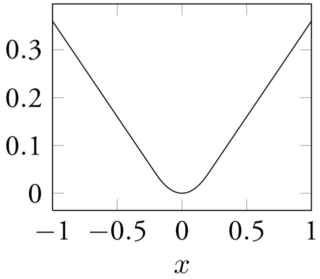


Figure 3: Huber cost function, $\beta = 0.2$.

$$(J^\top W J + \lambda I)\delta\theta = -J^\top W r\,, \tag{19}$$

where $W = \operatorname{diag}(w_1^2, \ldots, w_m^2)$. The weights are recomputed each iteration as the errors r_i change. In Paper C, iterative reweighting is used both for robust bundle adjustment and in triangulation and pose estimation.

6 Covariance Estimation

In Papers B and C we use the fact that $(J^\top \Sigma^{-1} J)^{-1}$, with J as above and Σ the measurement noise covariance matrix, provides an estimate of the covariance of the estimated structure and motion. Under assumptions of Gaussian noise, a simple and intuitive proof of this is given in [3], on which the following is based. The result is exact for affine projection functions; in the non-linear case, the function can be approximated by its tangent plane and an approximate estimate is obtained.

Lemma 1 Let v be a random vector in $\mathbb{R}^m$ with mean $\bar{v}$ and covariance matrix Σ, and $f\colon \mathbb{R}^m \to \mathbb{R}^p$ an affine mapping defined by $f(v) = f(\bar{v}) + A(v - \bar{v})$. Then $f(v)$ is a random variable with mean $f(\bar{v})$ and covariance matrix $A \Sigma A^\top$.

This follows easily from the definitions of mean and covariance, and is needed in the proof of the following theorem:

(**Result 5.9** [3]) Let $f\colon \mathbb{R}^p \to \mathbb{R}^m$ be an affine mapping of the form $f(\theta) = f(\bar{\theta}) + J(\theta - \bar{\theta})$, where J has rank p. Let $x \in \mathbb{R}^m$ be a Gaussian random variable with mean $\bar{x} = f(\bar{\theta})$ and covariance matrix Σ. Let $g\colon \mathbb{R}^m \to \mathbb{R}^p$ be the mapping taking a measurement x to the maximum likelihood parameter estimate $\hat{\theta}$. Then $\hat{\theta} = g(x)$ is a random variable with mean $\bar{\theta}$ and covariance matrix $(J^\top \Sigma^{-1} J)^{-1}$.

Proof. The measurement model $f\colon \mathbb{R}^p \to S \subset \mathbb{R}^m$ takes parameters to expected measurements. The surface S is the space of all possible noise-free measurements. Since J has full rank, f is one-to-one and invertible. Define $\eta\colon \mathbb{R}^m \to S$ as the function mapping any measurement x to the unique closest point on S, in the Mahalanobis norm $\|x\|_\Sigma = \sqrt{x^\top \Sigma^{-1} x}$. The composition $g = f^{-1} \circ \eta$ thus represents the maximum likelihood estimator, and may also be expressed as $g(x) = \operatorname{argmin}_\theta \|x - f(\theta)\|_\Sigma = \operatorname{argmin}_\theta \|x - f(\bar{\theta}) - J(\theta - \bar{\theta})\|_\Sigma$. Evaluating g is a weighted linear least-squares problem, and the normal equations give the solution $g(x) = \hat{\theta} = (J^\top \Sigma^{-1} J)^{-1} J^\top \Sigma^{-1} \big(x - f(\bar{\theta})\big) + \bar{\theta}$. This shows that g is an affine function of x. Since $f(\bar{\theta}) = \bar{x}$ and $\bar{\theta} = f^{-1}(\bar{x}) = g(\bar{x})$, Lemma 1 applies and gives the covariance matrix of $\hat{\theta}$ as $(J^\top \Sigma^{-1} J)^{-1} J^\top \Sigma^{-1} \Sigma \Sigma^{-1} J (J^\top \Sigma^{-1} J)^{-1} = (J^\top \Sigma^{-1} J)^{-1}$. □

This result may also be obtained using standard estimation theory. It can be shown that the maximum likelihood estimator under Gaussian noise, to first order, is unbiased and achieves the Cramér-Rao lower bound, i.e. that the covariance is given by the inverse of the Fisher information matrix, which is indeed $J^\top \Sigma^{-1} J$ (see e.g. [6]).

The condition that f is invertible means that no two points in the parameter space may give rise to the same measurements. This is the case for triangulation and pose estimation, where the cameras or points, respectively, are fixed in the coordinate system. However, when computing the covariance of a whole structure-and-motion system, the problem of gauge freedom appears. Any collection of scene points and cameras may be jointly translated, scaled and rotated without affecting the resulting image projections, thus an entire family of parameters are ML estimates of the same input data. Since for each degree of gauge freedom the rank of the information matrix drops by one, this ambiguity must be eliminated by choosing a minimal parametrization before the above result can be applied. To accomplish this, typically one camera is considered fixed, eliminating six degrees of freedom, and the distance to a second camera constrained, fixing the scale of the reconstruction.

Alternatively, one may take the Moore-Penrose pseudo-inverse of the rank-deficient information matrix instead, as the following results show.

(**Result 5.11** [3]) Let $f\colon \mathbb{R}^p \to \mathbb{R}^m$ be a differentiable mapping taking parameters θ to measurements x. Let S be a smooth manifold of dimension d embedded in $\mathbb{R}^p$ passing through point θ, and such that f is one-to-one on S in a neighborhood of θ, mapping S to a manifold $f(S) \subset \mathbb{R}^m$. Denote by f^{-1} the local inverse of f restricted to the surface $f(S)$ in a neighborhood of x. Let a Gaussian distribution be defined on $\mathbb{R}^m$ with mean x and covariance matrix Σ and let $\eta\colon \mathbb{R}^m \to f(S)$ be the mapping taking a point in $\mathbb{R}^m$ to the closest point on $f(S)$ in the Mahalanobis norm $\|\cdot\|_\Sigma$. Via $f^{-1} \circ \eta$ the probability distribution on $\mathbb{R}^m$ induces a distribution on $\mathbb{R}^p$ with covariance matrix equal to first order to $(J^\top \Sigma^{-1} J)^{+A} = A(A^\top J^\top \Sigma^{-1} J A)^{-1} A^\top$, where A is any matrix whose columns span the tangent space to S at θ.

Proof. Let $g\colon \mathbb{R}^d \to \mathbb{R}^p$ map an open neighborhood $U \subset \mathbb{R}^d$ to an open subset of S containing θ. Then $f \circ g\colon \mathbb{R}^d \to \mathbb{R}^m$ is one-to-one on U. Let J and A be the Jacobian matrices of f and g, respectively. The Jacobian matrix of $f \circ g$ is then JA. Result 5.9 can now be applied to the first-order expansion of $f \circ g$ about θ, transporting the covariance backwards to covariance matrix $(A^\top J^\top \Sigma^{-1} J A)^{-1}$. This matrix has rank and dimension d and so is invertible. Propagating this covariance forward through g into the higher-dimensional space using lemma 1 gives the desired result, $\Sigma_\theta = A(A^\top J^\top \Sigma^{-1} JA)^{-1} A^\top$. Substituting AB for A leaves this expression unchanged if B is invertible, so Σ_θ only depends on the column span of A. ☐

Note that we are free to choose the manifold S (and thus g and A). The following lemma simplifies the result for a particular choice:

Lemma 2 Let M be symmetric, and let M^+ be its Moore-Penrose pseudo-inverse. Then $M^+ = A(A^\top MA)^{-1}A^\top = M^{+A}$ if $A^\top MA$ is invertible and $\text{span}(A) = \text{span}(M)$.

Proof. Let $M = UDU^\top$ be the singular value decomposition of M. If M has rank r, U may be partitioned into $U = \begin{pmatrix} U' & U'' \end{pmatrix}$ where U' are the first r columns and $\text{span}(U') = \text{span}(M)$. We may write $M = U'D'U'^\top$ where $D' = \text{diag}(\sigma_1, \dots, \sigma_r)$. The pseudo-inverse can then be computed as $M^+ = U'(D')^{-1}U'^\top$. As remarked above, $M^{+A} = M^{+AB}$ for any invertible B. Since by assumption M and A span the same space, there is an invertible B s.t. $AB = U'$ (given by $B = A^+U'$). Now $M^{+A} = M^{+U'} = U'(U'^\top MU')^{-1}U'^\top = U'(U'^\top U'D'U'^\top U')^{-1}U'^\top = U'(D')^{-1}U'^\top = M^+$. ☐

By taking the pseudo-inverse of the Fisher information matrix, we are thus choosing $\text{span}(A) = \text{span}(J^\top\Sigma^{-1}J) = \text{span}(J^\top)$. This corresponds to the restricted parameter manifold S being locally orthogonal to the null-space of the Jacobian J. This is natural, since moving in this space does not change the measurements, but only explores the various gauge freedoms of the parametrization. In a sense, the pseudo-inverse allows us to compute covariances of structure-and-motion reconstructions independently of the coordinate system; if the gauge is locked by fixing parameters, the computed covariances will be expressed relative to these. On the other hand, the cost of computing the pseudo-inverse may be prohibitive in practice, while the ordinary inverse is not. As discussed in Paper C, different regularizing constraints may then be added to the reprojection function making J full rank.

References

[1] Martin Byröd, Klas Josephson, and Kalle Åström. Fast optimal three view triangulation. In Yasushi Yagi, In So Kweon, Sing Bing Kang, and Hongbin Zha, editors, *Asian Conference on Computer Vision*, 2007.

[2] Richard Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[3] Richard Hartley and Andrew Zisserman. *Multiple View Geometry*. Cambridge University Press, 2003.

[4] Fredrik Kahl and Richard Hartley. Multiple View Geometry Under the L_∞ Norm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1603–1617, 2008.

[5] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-Sturm vs. optimal correction. In *Proceedings of the British Machine Vision Conference*, pages 18.1–18.10. BMVA Press, 2008. doi:10.5244/C.22.18.

[6] Daniel D Morris. *Gauge Freedoms and Uncertainty Modeling for 3D Computer Vision*. PhD thesis, Carnegie Mellon University, 2001.

Paper A

A Step Towards Self-calibration in SLAM: Weakly Calibrated On-line Structure and Motion Estimation

SEBASTIAN HANER AND ANDERS HEYDEN
Centre for Mathematical Sciences, Lund University

Abstract: We propose a structure and motion estimation scheme based on a dynamic systems approach, where states and parameters in a perspective system are estimated. An on-line method for structure and motion estimation in densely sampled image sequences is presented. The proposed method is based on an extended Kalman filter and a novel parametrization. We derive a dynamic system describing the motion of the camera and the image formation. By a change of coordinates, we represent this system by normalized image coordinates and the inverse depths. Then we apply an extended Kalman filter for estimation of both structure and motion. Furthermore, we assume only weakly calibrated cameras, i.e. cameras with unknown and possibly varying focal length, unknown and constant principal point and known aspect ratio and skew. The performance of the proposed method is demonstrated in both simulated and real experiments. We also compare our method to the one proposed by Civera et al. and show that we get superior results.

1 Introduction

Estimation of 3D structure and motion from 2D images is a central problem in computer vision. There exist essentially two different approaches to solve this problem: batch approaches and iterative (recursive) approaches. Batch methods aim at providing an accurate result by using information from all images at the same time. These approaches are typically based on multi-view tensors, bundle adjustment or convex optimization, see [9] for the former and [13] for the latter. These methods are not suitable for real-time applications, both due to their complexity and off-line nature, requiring all images to be gathered before any computations can be made. Iterative (or recursive) algorithms aim for real-time performance, by updating a current estimate as soon as a new image becomes available. These algorithms are either based on variations of methods used for batch approaches, e.g. iteratively estimating the camera pose and the structure, [3], or by fast estimation of relative motion [18].

Yet another approach is to formulate the camera motion and the imaging process as a dynamic system and apply non-linear observers to estimate the structure and the translational and rotational velocities of the motion. The standard approach is to apply an extended Kalman filter to a dynamic system, with a perspective transformation in the output equations. One of the pioneering approaches is [2], where an extended Kalman filter is applied directly to the dynamic system, without any re-parametrization. Another approach, based on tracking the essential matrix can be found in [19].

For structure estimation only, i.e. known motion, a number of non-linear observers based on methods for automatic control theory have been developed, e.g. [17, 12, 4, 8, 1, 15, 14, 6]. Similar approaches, based on adaptive non-linear observers, for full structure and motion estimation can be found in [20, 21, 10].

Lately, [7, 5] developed a new parametrization for use with the extended Kalman filter, using the inverse depth, adjusting the uncertainties to the imaging situation and fixing the imaging rays from the first camera in order to gain stability. The parametrization is highly redundant but performs well in most situations, both in terms of accuracy and robustness. Another approach based on inverse scaling can be found in [16].

This paper describes how a re-parametrization of the underlying perspective dynamic system can be used to formulate the structure and motion estimation problem as an observer problem of a non-linear dynamic system, with a linear output function. We will show that this novel parametrization will result in a more accurate extended Kalman filter. Moreover, we will allow a weakly calibrated camera, i.e. a camera with unknown and possibly varying focal length, unknown and constant principal point and known aspect ratio and skew.

2 Problem Formulation

Consider a calibrated perspective camera that is observing a moving rigid object. Observe that it is just a philosophical difference between assuming a fixed camera and a moving object or a moving camera and a fixed object, since it is only the relative motion that can be estimated, but for modelling purposes one or the other might be preferable. Assume a camera system where the camera is situated at the origin and the optical axis is aligned with the z-axis. Let y_i denote the image coordinates and x_i denote the (time-varying) object coordinates. Introducing

$$\xi = \left(\frac{x_1}{x_3}, \frac{x_2}{x_3}\right)^\top , \tag{A.1}$$

motion and image formation can be described by the dynamic system

$$\begin{aligned} \dot{x} &= Ax + b \\ y &= C_f \xi + \delta \, , \end{aligned} \tag{A.2}$$

where

$$A = S(\omega) = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \tag{A.3}$$

is the skew symmetric matrix obtained from the (possibly time varying) angular velocity vector

$$\omega = \left(\omega_1, \omega_2, \omega_3\right)^\top , \tag{A.4}$$

$$b = \left(b_1, b_2, b_3\right)^\top \tag{A.5}$$

denotes the (possibly time varying) translational velocity, and C_f and δ are intrinsic camera parameters. In our case we have

$$C_f = \begin{pmatrix} af_c & sf_c \\ 0 & f_c \end{pmatrix} , \tag{A.6}$$

where s denotes the (known) skew, a the (known) aspect ratio and f_c the (unknown and possibly varying) focal length and

$$\delta = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} , \tag{A.7}$$

denotes the (unknown) principal point. After a suitable change of coordinates, we may assume that

$$C_f = \begin{pmatrix} f_c & 0 \\ 0 & f_c \end{pmatrix} , \tag{A.8}$$

since s and γ are assumed to be known.

We can now state the problem as follows:

On-line structure and motion estimation: Given the image coordinates y from (A.2), estimate recursively the object coordinates x, the (time varying) motion parameters ω and b, the (time varying) focal length f and the principal point δ.

3 The Parametrization

Consider (A.2) and define the scalar γ and the vector z by

$$\gamma = \frac{1}{\sqrt{x^\top x}}, \quad z = \gamma x \,, \tag{A.9}$$

which can be interpreted as the inverse distance to the object. Observe that ξ, according to (A.1) and by the definition of z, also can be expressed as

$$\xi = \left(\frac{z_1}{z_3}, \frac{z_2}{z_3}\right)^\top \,. \tag{A.10}$$

This means, using (A.9) and the definition of ξ in (A.1), that the vector z may be assumed known, since it can be expressed as

$$z = \frac{1}{\sqrt{\xi_1^2 + \xi_2^2 + 1}} \left(\xi_1, \xi_2, 1\right)^\top \,. \tag{A.11}$$

This vector can be interpreted as the image coordinates on a spherical image plane.

In the case of calibrated cameras, z is a measurable signal, and can therefore be considered an output of the system (A.2). The parametrization exploits this fact, and aims at rewriting the system (A.2) so that z appears explicitly in the equations. In the self-calibration case, i.e. where the focal length f_c and the principal point (x_0, y_0) are unknown, z is measurable up to a transformation involving the intrinsic parameters of the camera.

Using (A.2) and the fact that $x^\top A x = 0$ since A is skew-symmetric, gives, introducing

$$g_0(z) = I - zz^\top \,, \tag{A.12}$$

a rewritten dynamic system, corresponding to (A.2), on the form

$$\begin{aligned} \dot{z} &= Az + g_0(z) b \gamma \\ \dot{\gamma} &= -\gamma^2 z^\top b \\ y &= C_f \xi + \delta \,. \end{aligned} \tag{A.13}$$

For the motion of more than one point, a dynamic system corresponding to (A.13) is obtained as

$$\begin{aligned} \dot{z}^i &= Az^i + g_0(z^i)b\gamma^i \\ \dot{\gamma}^i &= -(\gamma^i)^2(z^i)^\top b \quad , \quad i \in \{1,2,\dots,N\}, \\ y^i &= C_f\xi^i + \delta \end{aligned} \tag{A.14}$$

where N denotes the number of feature points. Equation (A.9) together with (A.13) and its multipoint version (A.14), constitute the desired dynamic vision parametrization, from which we shall proceed. Observe that the dynamic system contains four state variables per point; three for z and one for γ and that z has to fulfill the constraint $|z| = 1$.

4 The Extended Kalman Filter

The extended Kalman filter estimates the system state s_k given a previous estimate $\hat{s}_{k-1}$, a new measurement μ and state transition and observation models $s_k = f(s_{k-1})$ and $\mu_k = h(s_k)$. At every time-step the new state and the state covariance P are predicted,

$$\begin{aligned} \hat{s}_{k|k-1} &= f(\hat{s}_{k-1|k-1}) \\ P_{k|k-1} &= F_{k-1}P_{k-1|k-1}F_{k-1}^\top + Q_{k-1} \end{aligned} \tag{A.15}$$

and, given a new measurement μ_k, corrected to

$$\begin{aligned} \hat{s}_{k|k} &= \hat{s}_{k|k-1} + K_k\big(\mu_k - h(\hat{s}_{k|k-1})\big) \\ P_{k|k} &= P_{k|k-1} - K_kH_kP_{k|k-1} \end{aligned} \tag{A.16}$$

where

$$\begin{aligned} F_{k-1} = \left.\frac{\partial f}{\partial s}\right|_{\hat{s}_{k-1|k-1}} , \quad H_k = \left.\frac{\partial h}{\partial s}\right|_{\hat{s}_{k|k-1}} \\ K_k = P_{k|k-1}H_k^\top(H_kP_{k|k-1}H_k^\top + R_k)^{-1} \end{aligned} \tag{A.17}$$

and Q and R the assumed process and measurement noise covariances, respectively.

Adapting the dynamic system (A.14) to the EKF setting, the state vector is taken to be

$$s = \left(b^\top, \omega^\top, f_c, x_0, y_0, (z^1)^\top, \gamma^1, \ldots, (z^N)^\top, \gamma^N\right)^\top, \tag{A.18}$$

while the measurement vector is given by

$$\mu = h(s) = \left(y_1^1, y_2^1, \ldots, y_1^N, y_2^N\right)^\top \tag{A.19}$$

with components

$$y^i = C_f \xi^i + \delta = \begin{pmatrix} f_c & 0 \\ 0 & f_c \end{pmatrix} \begin{pmatrix} z_1^i / z_3^i \\ z_2^i / z_3^i \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}. \tag{A.20}$$

The update equation is a discretized version of (A.14):

$$\left.\begin{aligned} \tilde{z}^i &= e^{S(\omega_k)} z_k^i + g_0(z_k^i) b_k \gamma_k^i \\ \tilde{\gamma}^i &= \gamma_k^i - (\gamma_k^i)^2 (z_k^i)^\top b_k \\ z_{k+1}^i &= \tilde{z}^i |\tilde{z}^i|^{-1} \\ \gamma_{k+1}^i &= \tilde{\gamma}^i |\tilde{z}^i| \end{aligned}\right. , \quad i \in \{1, 2, \ldots, N\}, \tag{A.21}$$

where $|z^i| = 1$ is enforced.

Note that we assume a camera-centric coordinate system and estimate only linear and angular velocities, which must be integrated over time to recover the absolute motion.

Adding features

A main advantage of the camera-centric coordinate system is the ease with which new features can be inserted into the filter; the uncertainty of new features is independent of any extrinsic camera parameters, unlike in the unified inverse depth parametrization. Removing features simply means deleting the corresponding entries, rows and columns in the state vector and covariance matrices, but if a feature is only temporarily occluded or otherwise not detected, it can still be kept in the filter if it is assigned an infinite measurement uncertainty.

Complexity

The computational complexity of the filter can be made lower than that of e.g. the unified inverse depth parametrization. First note that since the output function h is linear if the measured image coordinates are first transformed using equation (A.11), it is fully represented by the Jacobian H, and very sparse. In fact, all non-zero elements equal one, and multiplying a matrix by H amounts to removing rows or columns of the matrix. Thus four matrix multiplications can be avoided in the filter update step. The update equation, however, is not linear, due to the camera-centric representation, and the Jacobian F will no longer be (nearly) diagonal, as in the world-centric case. But it will still be rather sparse, with the first 6 columns full and the rest block diagonal with 4-by-4 blocks. The a-priori covariance update can thus still be performed quite efficiently. The fact that only three or four parameters are required per feature is a considerable advantage compared to the unified inverse depth scheme, where features must eventually be converted to a Cartesian parametrization to maintain frame rate.

5 Experiments

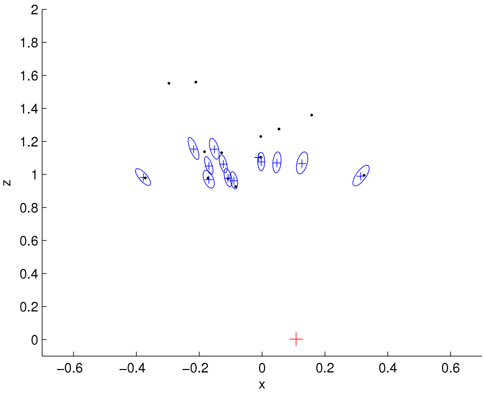
In the following experiments, no priors on the structure or motion are given. Features are initialized at an arbitrary depth and with large uncertainty in the γ coordinate. The linear and angular velocities are assumed constant, and acceleration is modelled as zero-mean Gaussian process noise. When assuming varying focal length, the variation is also modelled as process noise, while an unknown principal point is assumed fixed but initialized with some uncertainty at the center of the image.

As has been reported in [5], the EKF can converge under these circumstances; however, it is found that fixing the depth of one point, thus determining the overall scale, greatly aids convergence. Further, the normalization step of the update equation (A.21) has been found not to be strictly necessary (when using the full parametrization) and in fact does not significantly impact the results.

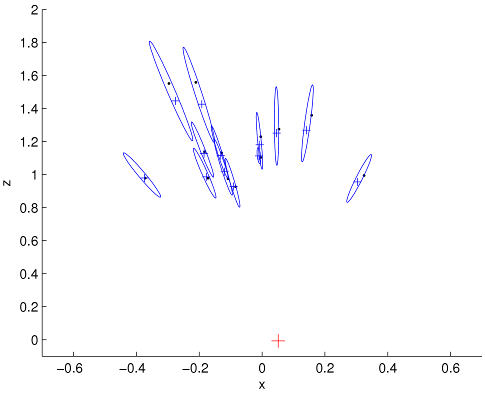
Constant and known intrinsic parameters

We repeat an experiment in [16] and show that the proposed parametrization does not suffer from the underestimation of uncertainty associated with the inverse depth parametrization of [5] and typically converges faster as a result (Figure A.1

and A.2). This issue of inconsistency is common to many SLAM algorithms and is analyzed in e.g. [11].



(a) Unified inverse depth



(b) Proposed

Figure A.1: Position and covariance estimates after observing 30 frames of simulated data (black: ground truth, blue: estimate $\pm\sigma$). The inverse depth parametrization underestimates the errors, here leading to slower convergence, while the proposed parametrization more accurately captures the depth uncertainty.

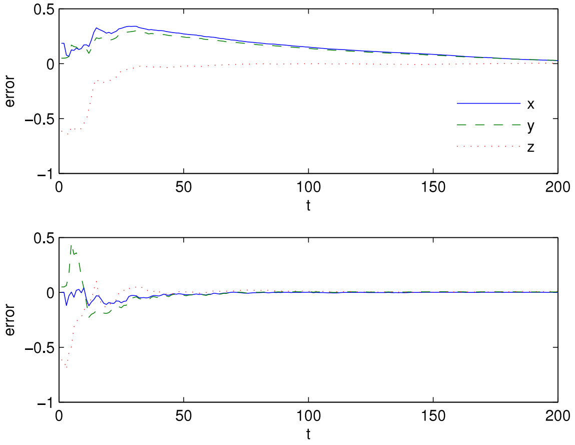


Figure A.2: Convergence plot of the Cartesian coordinates of a point in a simulated reconstruction problem. Top: inverse depth, bottom: proposed parametrization.

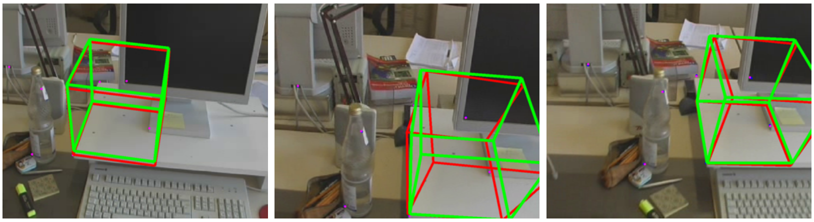


Figure A.3: Visual result of integrating geometry into a tracked video sequence (from left to right, frames 1, 50 and 70 are shown). The green box shows the solution using the proposed method, while the red was computed using the inverse depth parametrization. Although the re-projection errors are similar, the proposed method produces a more accurate motion estimate.

Varying focal length and unknown principal point

In experiments on noisy simulated data the filter is able to track varying focal length and determine an offset in the principal point (Figure A.4 and A.5). Observe that even if the convergence rate for the principal point is relatively low, the motion estimation and structure estimation is converging much faster.

Real data

We also apply the proposed and unified inverse depth methods to a real video sequence. The camera motion and 3D coordinates of 7 feature points tracked (using the KLT algorithm) over 70 frames of a desktop sequence are reconstructed. Some geometry is overlaid to verify the results (Figure A.3). A (subjective) assessment indicates that the proposed method gives a more consistent reconstruction than unified inverse depth.

In a more general setting we use the following structure and motion framework:

- From the first frame, extract SURF features and add them to the state vector. The observed features are initialized to lie in a plane at unit depth. To set the overall scale, the depth of one feature is fixed by assigning zero uncertainty in the γ coordinate.
- For subsequent frames:
    1. Extract and match features to those active in the filter. Remove outliers by fitting an affine transformation between the observed feature locations and their predicted locations in a RANSAC scheme.
    2. Assign active features not detected in the current frame infinite measurement uncertainty. Features not detected for a set number of frames, e.g. 100, are removed from the filter by deleting the appropriate entries.
    3. Update the filter state.
    4. If the number of active features is too low, select new ones from the unmatched features in the current frame and initialize their depth as the mean depth of the currently visible active features.

While SURF features are computationally expensive (compared e.g. to the approach in [7]), they facilitate the redetection of previously observed landmarks. In Figure A.6 the algorithm is applied to a typical augmented reality scenario.

6 Conclusions

We have used a novel parametrization together with the extended Kalman filter for full structure and motion estimation, and successfully dealt with unknown

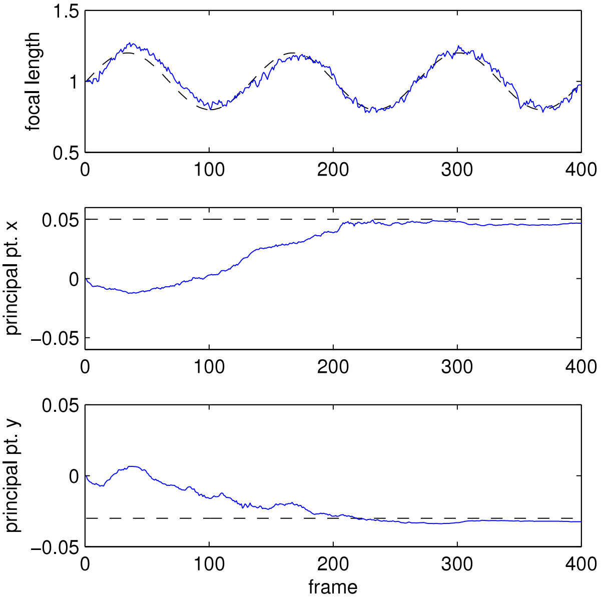


Figure A.4: Result of experiment on simulated data. 15 intermittently visible features were observed by a circling camera at a noise level of about 1 pixel. The focal length varied while the principal point remained fixed. Dashed lines: ground truth.

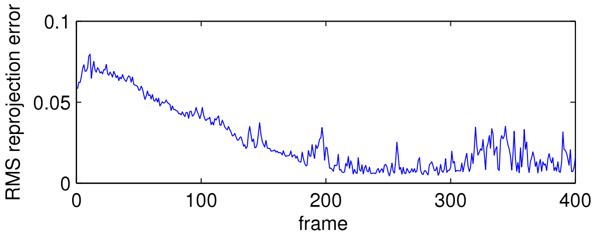


Figure A.5: Re-projection error in the above experiment. The error increases towards the end of the sequence as the camera moves further from the point cloud.

and varying focal length and unknown principal point. The filter has low computational complexity, is shown to perform well on both simulated and real data and has been favorably compared to state-of-the-art approaches.

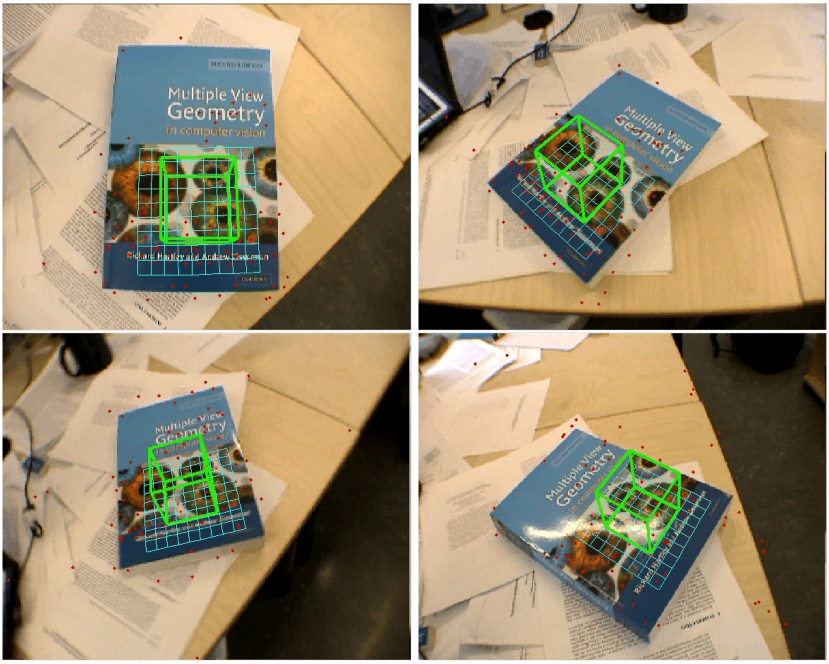


Figure A.6: Visual result of integrating geometry into a video sequence using the proposed parametrization and a fully calibrated camera. SURF feature matching and RANSAC provide robust tracking. The cube is automatically aligned after detecting the dominant plane.

References

[1] Rixat Abdursul, Hiroshi Inaba, and Bijoy K. Ghosh. Nonlinear observers for perspective time-invariant linear systems. *Automatica*, 40:481–490, 2004.

[2] Ali Azarbayejani and Alex P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1995.

[3] P.A. Beardsley, A. Zisserman, and D.W. Murray. Sequential updating of projective and affine structure from motion. *International Journal of Computer Vision*, 23(3):235–259, 1997.

[4] Xinkai Chen and Hiroyuki Kano. A new state observer for perspective systems. *IEEE Transactions on Automatic Control*, 47(4):658–663, April 2002.

[5] J. Civera, A.J. Davison, and J. Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

[6] Ola Dahl, Fredrik Nyberg, and Anders Heyden. Nonlinear and adaptive observers for perspective dynamic systems. In *American Control Conference*, July 2007.

[7] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[8] W. E. Dixon, Y. Fang, D. M. Dawson, and T. J. Flynn. Range identification for perspective vision systems. *IEEE Transactions on Automatic Control*, 48(12):2232–2238, December 2003.

[9] Richard Hartley and Andrew Zisserman. *Multiple View Geometry*. Cambridge University Press, 2003.

[10] A. Heyden and O. Dahl. Provably convergent structure and motion estimation for perspective systems. In *Control Decision Conference*, 2009.

[11] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Analysis and improvement of the consistency of extended Kalman filter-based SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 473–479, Pasadena, CA, 2008.

[12] Mrdjan Jankovic and Bijoy K. Ghosh. Visually guided ranging from observations of points, lines and curves via an identifier based nonlinear observer. *Systems & Control Letters*, 25:63–73, 1995.

[13] F. Kahl. Multiple view geometry and the L_∞-norm. In *International Conference on Computer Vision*, pages 1002–1009. IEEE Computer Society Press, 2005.

[14] Dimitrios Karagiannis and Alessandro Astolfi. A new solution to the problem of range identification in perspective vision systems. *IEEE Transactions on Automatic Control*, 50(12):2074–2077, December 2005.

[15] Lili Ma, YangQuan Chen, and Kevin L. Moore. Range identification for perspective dynamic systems with 3d imaging surfaces. In *American Control Conference*, June 2005.

[16] D. Marzorati, M. Matteucci, D.A. Migliore, and D.G. Sorrenti. Monocular slam with inverse scaling parametrization. In *BMVC08*, 2008.

[17] Larry Matthies, Takeo Kanade, and Richard Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3:209–236, 1989.

[18] D. Nister. An efficient solution to the five-point relative pose problem. *Int. Conf. Computer Vision and Pattern Recognition*, 2:195–202, 2003.

[19] Stefano Soatto. 3-d structure from visual motion: Modeling, representation and observability. *Automatica*, 33(7):1287–1312, 1997.

[20] Stefano Soatto and Pietro Perona. Reducing structure from motion: A general framework for dynamic vision, part 1: Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9), 1998.

[21] Stefano Soatto and Pietro Perona. Reducing structure from motion: A general framework for dynamic vision, part 2: Implementation and experimental assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9), 1998.

Paper B

Optimal View Path Planning for Visual SLAM

SEBASTIAN HANER AND ANDERS HEYDEN
Centre for Mathematical Sciences, Lund University

Abstract: In experimental design and 3D reconstruction it is desirable to minimize the number of observations required to reach a prescribed estimation accuracy. Many approaches in the literature attempt to find the next best view from which to measure, and iterate this procedure. This paper discusses a continuous optimization method for finding a whole set of future imaging locations which minimize the reconstruction error of observed geometry along with the distance traveled by the camera between these locations. A computationally efficient iterative algorithm targeted toward application within real-time SLAM systems is presented and tested on simulated data.

1 Introduction

Visual simultaneous localization and mapping (SLAM) is the task of determining the position and orientation of a camera while concurrently building a map of the environment, using the camera images and possibly other sensors as input. It is a chicken-and-egg type problem; given the map, localization is relatively easy and given the camera positions, map triangulation is straightforward. Accomplishing both at once is at the heart of the SLAM problem, which has received a lot of attention in both the robotics and vision research communities. Much effort is spent improving the robustness and accuracy of algorithms, particularly with respect to error accumulation, drift and loop closing (see e.g. [1, 11]). A less studied problem is how to make efficient use of the information collected in active SLAM systems, i.e. systems where the motion of the sensor can be controlled. This article considers the problem of maximizing the useful information gained from a fixed number of images by active planning of the vision sensor movement. Specifically, we consider the task of finding a camera trajectory between two predetermined locations such that the reconstruction accuracy of observed geometry is maximized while the path length is minimized. The envisioned application is robot path planning, where the accuracy usually is a secondary objective, so the focus is on providing the best reconstruction given time or distance constraints.

In this work we only consider the geometric aspects of the problem and do not account for availability of texture or object occlusion, which are of course issues in a real system relying on feature tracking. We further assume the following:

- An initial maximum likelihood estimate of the structure is available, based on observations up to that point.
- All cameras along the trajectory are oriented towards a particular point of interest, e.g. the centroid of the features to be estimated.
- The camera can be positioned with such relative accuracy that its pose and location is fully known at each observation.

These assumptions may be relaxed, as discussed in Section 6.2. Finally, the robot path is represented by a sequence of camera locations, and the number of cameras on the path must be chosen in advance.

As an experimental design problem, so-called 'camera network design' has been studied extensively in the photogrammetry literature. The emphasis is on obtaining the most accurate reconstruction given a limited number of cameras, and time can be spent finding an optimal configuration. For example, in [5] a genetic optimization algorithm is used to search the high-dimensional parameter space of camera placements. Similar stochastic algorithms are usually employed since the problem is intrinsically multi-modal i.e. the objective function has many local minima, cf. [6]. In the context of 3D reconstruction in controlled environments, the task at hand is usually referred to as 'next best view planning', suggesting that given an approximate reconstruction we seek a single next view that will reduce the error the most. This is the case in [14] where the authors reconstruct objects using a camera mounted on a robotic arm. The object geometry is estimated using a Kalman filter, and the next imaging location is determined by searching a discrete parameter space and evaluating the expected information gain in the filter at each position. A different approach is taken in [13] where the next imaging location is decided based only on the single currently least well-determined feature, allowing a simple closed form solution. In the above problem formulations there are usually few or no constraints imposed on possible sensor configurations, computational complexity is less of an issue and the 'next best view' approaches do not consider more than one future observation. This work will show that given constraints on the camera positions, good solutions for many future observations can be found relatively quickly. For a recent general survey of the sensor planning field see the book by Chen et al. [2].

The work most similar in spirit to ours is [4] where the path of a robot moving in the plane is planned based on the expected reconstruction accuracy of an observed object. An approximation of the geometry is given and the expected information gain from observing the object from a particular vantage point is determined on a discrete grid of camera locations. Each grid cell is assigned a cost proportional to the inverse of the information gain, and a minimum cost path is found between the starting point and the global minimum grid cell. The algorithm does not take into account the new information gained after an actual observation is made, however, and becomes computationally expensive if we allow the camera to move in three dimensions. The minimum cost path formulation also restricts the choice of cost function. This work proposes an efficient continuous optimization approach to the problem of finding a short path with large information gain.

2 Problem Formulation

The planner takes as input an initial estimate of the structure, the current location of the sensor and the desired destination. The output is a path, represented by a discrete set of sensor locations, connecting these points. The number of locations on the path can be set explicitly or deduced from e.g. the robot's speed and sample rate and the distance to be travelled. For the experiments in this paper the sensor is assumed to be a single fully calibrated camera, although extension to stereo and multi-camera systems is straightforward. The standard pinhole camera model is used, so that the relation $x = f(P, X)$ between a world point X and its projection x in homogeneous coordinates is given by

$$\lambda f(P, X) = KM \begin{pmatrix} X \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & \mid & -Rt \end{pmatrix} \begin{pmatrix} X \\ 1 \end{pmatrix} \tag{B.1}$$

where R and t are the camera rotation and translation and K represents the known intrinsic calibration parameters. However, any differentiable projection function $f(P, X)$ may be substituted, e.g. to include radial distortion terms.

In the interest of reducing the parameter space dimension, each camera is parametrized only by its position and is automatically oriented toward a point of interest, typically chosen as the centroid of the structure under consideration. Features are deemed visible if they fall within the camera's field of view; possible occlusion by other objects is not considered. The measurement uncertainty of features is also considered fixed.

We define the optimization problem as follows: *minimize the reconstruction uncertainty of observed geometry and the distance traveled by the sensor between imaging locations*. These are conflicting objectives, which are combined in a cost function defined below.

3 Cost Function

Lacking ground truth data or other *a priori* information, the quality of a reconstruction can only be judged by the statistical uncertainty of the estimate. Condensing a probability distribution into a scalar quality measure is not entirely straight-forward, however, and choices must be made depending on the intended application. Also, in most situations only estimates of the probability distribution are available, e.g. the mean and covariance. In the experimental design literature, many summary statistics have been proposed and are usually functions of the eigenvalues of the covariance matrix, e.g. the trace and determinant, cf. [9]. In the structure-from-motion problem, the eigenvalues have a direct geometric interpretation which we consider below.

If we assume the position and orientation of the camera is fully known when an observation is made, the structure estimates corresponding to individual features are independent of each other, and the covariance matrix is block diagonal with 3-by-3 blocks (assuming point features). The eigenvalues of each block correspond to the semi-axes of the ellipsoid representing the variance of the feature location. We would like these ellipsoids to be as small as possible, but in what sense? If we minimize the volume, i.e. the determinant, we admit solutions where a point may be very well-determined in two directions but with a large uncertainty in the third (typically the depth). Minimizing the determinant of the entire covariance matrix (the so-called D-optimality criterion) could favor solutions where one point is very well determined while others are much less certain. For navigation and mapping purposes, we would like all, or at least the majority of features to be reconstructed to reasonable accuracy. Minimizing the largest eigenvalue (E-optimality) would achieve this, but results in a non-smooth objective function. We choose to minimize the sum of the eigenvalues (A-optimality), i.e. the trace of the covariance matrix, which provides a good trade-off with the added computational benefit of not having to calculate individual eigenvalues.

Before introducing the cost function, we discuss how to compute the trace given a set of measurements.

3.1 Calculating Covariance

In many recent SLAM systems (e.g. [8, 12, 10]) maximum likelihood estimates obtained via bundle adjustment are available. We assume the structure estimate is optimal in the ML sense with respect to the observations; then the information matrix is given to first order by $I = J^\top R^{-1} J$ where J is the Jacobian of the reprojection error evaluated at the minimum, and R the measurement noise covariance [7]. Also, the (pseudo-)inverse of I gives an approximation of the covariance matrix. Since information is additive, including new observations in the estimate amounts to summing the individual information matrices. In other words, to calculate the effect of new observations on the structure estimate, we compute the Jacobian of each observation and add the corresponding information matrices to the initial one. New observations may of course shift the ML estimate, invalidating the approximation, but this is avoided in a natural way as discussed in Section 4.

Given a world point X and a camera P, let $\hat{x}$ be the measured image coordinate, and $f(P, X)$ the projection function mapping X to the expected image coordinate x. Define the re-projection error as $E_X(P, X, \hat{x}) = f(P, X) - \hat{x}$ with Jacobian

$$J_X = \frac{dE_X}{dX} = \begin{pmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \frac{\partial f_1}{\partial X_3} \\ \frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2} & \frac{\partial f_2}{\partial X_3} \end{pmatrix} . \tag{B.2}$$

If several points $X^{1,\dots,N}$ are observed simultaneously, let

$$E(P, X^{1:N}, \hat{x}^{1:N}) = \begin{pmatrix} E_{X^1} \\ \vdots \\ E_{X^N} \end{pmatrix} \tag{B.3}$$

with block diagonal Jacobian

$$J = \begin{pmatrix} J_{X^1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & J_{X^N} \end{pmatrix} . \tag{B.4}$$

The information matrix for a single image is then given by

$$I(P, X^{1:N}) = \begin{pmatrix} J_{X^1}^\top R_1^{-1} J_{X^1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & J_{X^N}^\top R_N^{-1} J_{X^N} \end{pmatrix} \tag{B.5}$$

where usually the $R_i = \left(\begin{smallmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{smallmatrix}\right)$.

The final information matrix given the initial information I_0 and images from camera positions $P^{1,\dots,M}$ is now

$$I_M = I_0 + \sum_{j=1}^{M} I(P^j, X^{1:N}) \ . \tag{B.6}$$

Note that the computation is linear in the number of observed features and the number of images, and that the covariance of the estimate is the inverse, $\Sigma_{P^{1:M}, X^{1:N}} = I_M^{-1}$. For notational convenience, from hereon let P denote the set $P^{1:M}$ of camera poses along a path, and $X = X^{1:N}$ the estimated structure.

3.2 Cost Function

We propose the following cost function:

$$\begin{aligned} C(P, X) &= \frac{1}{N}\mathrm{tr}(\Sigma_{P,X}) + \frac{\alpha}{(M-1)^{1-q}} \sum_{j=1}^{M-1} \|P_{\mathrm{pos}}^{j+1} - P_{\mathrm{pos}}^{j}\|^q \\ &= U(P, X) + \alpha D(P) , \end{aligned} \tag{B.7}$$

i.e. the uncertainty measure plus a function of the camera path, weighted by a constant factor $\alpha > 0$, where $q \geq 1$. The normalization constants N^{-1} and $(M-1)^{q-1}$ are designed to make the cost approximately invariant with respect to the number of observed features and camera positions on the path. Note that by choosing $q > 1$, $D(P)$ will favor solutions with equidistant spacing between the camera positions, and introducing an offset d, $D(P) = \sum_{j=1}^{M-1}(\|P_{\mathrm{pos}}^{j+1} - P_{\mathrm{pos}}^{j}\| - d)^q$, we can impose the soft constraint that the path length be $d(M-1)$, if desired.

3.3 Cost Function Properties

The multi-modality of the objective functions normally used in next best view planning makes optimization difficult. The proposed cost function is no exception, but due to the somewhat local nature of the sought solution there are obvious bounds on the cost and geometry of the path.

Proposition B.1. *$U(P^{1:M}, X)$ is a non-negative decreasing function of the number of observations M.*

Proof. The information matrix I is positive semidefinite. Including a new observation amounts to adding another positive semidefinite matrix ΔI to I, and the result is again positive semidefinite. By the Courant-Fischer theorem, we know that the (sorted) eigenvalues satisfy $\lambda_i(I + \Delta I) \geq \lambda_i(I)$ for all $i = 1, \ldots, n$ and equivalently $\lambda_i(\Sigma_{\text{updated}}) = \lambda_i\big((I + \Delta I)^{+}\big) \leq \lambda_i(I^{+}) = \lambda_i(\Sigma_{\text{initial}})$. Evidently $\operatorname{tr}(\Sigma_{\text{updated}}) \leq \operatorname{tr}(\Sigma_{\text{initial}})$. □

Theorem B.2. *The length of the path at the minimum P^* is bounded.*

Proof. Given any initial estimate $\hat{P}$ of the path, we have

$$\begin{aligned} \alpha D(P^*) &\leq U(\hat{P}, X) + \alpha D(\hat{P}) - U(P^*, X) \\ &\leq U(\hat{P}, X) + \alpha D(\hat{P}) \\ &\leq U_{\text{initial}} + \alpha D(\hat{P}) \end{aligned}$$

where $U_{\text{initial}} = \frac{1}{N}\operatorname{tr}(\Sigma_0)$ and Σ_0 the covariance of the current structure estimate. Since $\|P_{\text{pos}}^{j+1} - P_{\text{pos}}^{j}\| < \|P_{\text{pos}}^{j+1} - P_{\text{pos}}^{j}\|^q + 1$, the length of P^* is bounded from above by $(M-1)^{1-q}\big(\alpha^{-1}U_{\text{initial}} + D(\hat{P})\big) + M - 1$. □

We see that the path must be contained inside an ellipsoid with foci at the (fixed) first and last camera positions, and that the bound can be computed easily in advance. As expected, the optimal path approaches the line segment between the foci as α grows.

This result suggests that we may attempt to find and compare several local minima by optimizing with varying initial paths sampled from within the feasible ellipsoid.

4 Proposed Algorithm

As noted in the introduction, the next best view problem is known to suffer from multiple local minima, cf. [6]; this is true for all reasonable choices of U. Finding the global minimum is a difficult problem, and the prevailing approach in the literature seems to be more or less exhaustive search over a discretized parameter space, [14, 4], or stochastic optimization methods, [3, 5]. In the interest of speed, however, we adopt a gradient based optimization scheme, using the well-known Levenberg-Marquardt (LM) method. LM minimizes the 2-norm of a residual vector r, which we construct as

$$r = \left(\frac{\mathrm{tr}(\Sigma_{P,X^1})}{N}, \ldots, \frac{\mathrm{tr}(\Sigma_{P,X^N})}{N}, \frac{\alpha \| P^2_{\mathrm{pos}} - P^1_{\mathrm{pos}} \|^q}{(M-1)^{1-q}}, \ldots, \frac{\alpha \| P^M_{\mathrm{pos}} - P^{M-1}_{\mathrm{pos}} \|^q}{(M-1)^{1-q}} \right)^{\frac{1}{2}}$$

(the exponent indicates element-wise square root) so that $\|r\|^2 = C(P, X)$. The parameter space is the $M - 2$ intermediate camera positions; the camera orientation is determined by its position and the interest point.

The final hurdle is how to evaluate the cost function *before* any observations are made. The best we can do is predict what the camera will see at a particular location given the current best estimate of the structure. Assuming that measurements are corrupted with zero-mean noise, the expected observation is simply the projection $x = f(P_i, X)$. Such an observation has zero reprojection error, and so does not affect the ML estimate.

The optimization is applied within the following framework:

1. Given an initial estimate of the structure, calculate its centroid and let this be the camera's point of interest. Select a target location for the camera, i.e. select the end point of the path.

2. Generate an initial path by linear interpolation between the first and last camera locations. The number of discrete camera locations along the path could be selected to match the image sampling rate and speed of the robot, but this would normally result in far too many locations and a very high-dimensional search space. However, it stands to reason that more images taken from approximately the same vantage point do not contribute qualitatively to the reconstruction, so a relatively sparse distribution of camera locations is sufficient.

3. Find a minimum of the cost function wrt. P using the LM algorithm. For improved convergence, an optimal step length may be selected through line search.

4. Move the camera to the next location along the path and make an actual observation. Update the structure estimate with this new information, and update the camera interest point location and path end point, if needed.

Repeat steps 3 and 4, each time with one less camera location along the path and using the previous path estimate as an initial guess.

5 Experiments

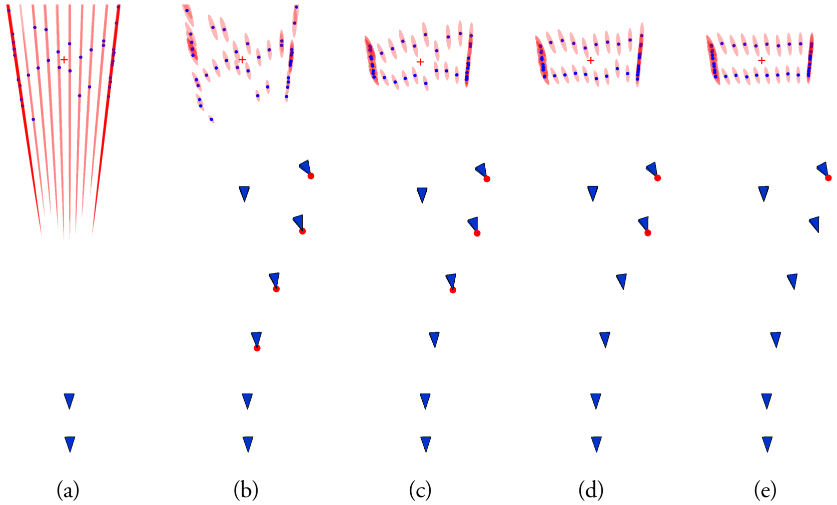
We first apply the above algorithm to the scenario of a robot trying to pass through a doorway. The doorway is represented by a rectangular array of point features which are optimally triangulated from the first two views, see Figure B.1(a). In all experiments we assume an image measurement noise σ equivalent to about one pixel. The target location is placed in front of the doorway, and the path is discretized with four waypoints in between. The optimization is run until convergence and the robot is moved to the next prescribed location along the path, where a new image is acquired and the structure estimate is updated using bundle adjustment.

The influence of the parameter α is illustrated in Figure B.2 and Table B.1. The robot passes by a point cloud, and to get a closer look it must make a detour. A large α penalizes long paths at the expense of reconstruction accuracy.

6 Discussion

6.1 Computational Complexity

As noted in Section 3.1, the cost function can be evaluated in $\mathcal{O}(MN)$ time. The LM algorithm requires the computation of the Jacobian of the residual vector r each iteration. The analytic expression may be very complicated and expensive to evaluate, so a finite difference approximation is preferred. The cost function must be differentiated with respect to $3(M-2)$ parameters, requiring $3(M-2)+1$ function evaluations to compute the Jacobian. But the covariance matrix is a function of a sum of individual information matrices, where only one term changes



(a) (b) (c) (d) (e)

Figure B.1: Doorway scenario. The robot wishes to approach the passage while determining its geometry as accurately as possible. The first two cameras on the path represent the last two images the robot has acquired and provide the initial optimal triangulation of the geometry. Red dots indicate which cameras are free to move, the red cross is the point of interest. In this case subsequent observations do not visibly change the initially planned path. The uncertainty ellipsoids represent 5σ in (a) and 50σ in (b)-(e). Note that in the latter cases the *expected* uncertainties, given all observations along the path, are displayed. The values $q = 3$ and $\alpha = 4.5 \cdot 10^{-7}$ were used.

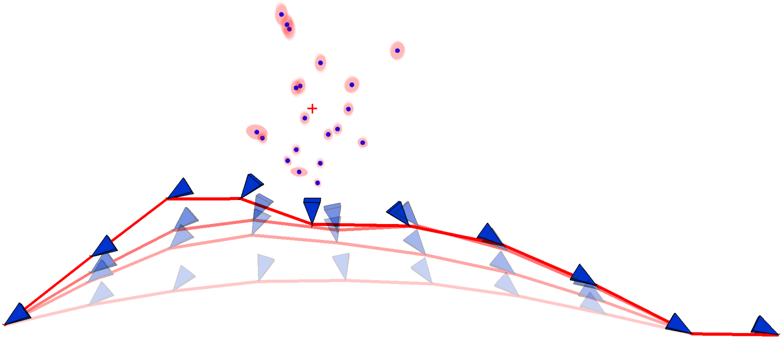


Figure B.2: Here the robot passes (from right to left) by a point cloud and makes a detour to get as close to the features as possible; this is natural, since the closer the feature, the higher its angular resolution. Four cases are plotted, fading out with increasing values of α.

	Optimized path		Straight path	
α	Rel. err.	Rec. err.	Rel. err	Rec. err
$1.0 \cdot 10^{-7}$	$1.64 \cdot 10^{-3}$	$8.32 \cdot 10^{-4}$	$2.02 \cdot 10^{-3}$	$1.03 \cdot 10^{-3}$
$0.5 \cdot 10^{-7}$	$1.25 \cdot 10^{-3}$	$7.15 \cdot 10^{-4}$	"	"
$0.2 \cdot 10^{-7}$	$5.36 \cdot 10^{-4}$	$4.53 \cdot 10^{-4}$	"	"

Table B.1: Relative error $U(P, X)/U(P^{1:2}, X)$ and absolute reconstruction error $\frac{1}{N}\sum_{i=1}^{N} \|X^i - X^i_{\text{true}}\|$, where $X^{1:N}_{\text{true}}$ is the ground truth structure being observed, computed for different values of α in the scenario of Figure B.2. The relative error represents the expected decrease in uncertainty from the initial estimate given by the first two images, the reconstruction error the actual error after all observations have been made. As α is decreased, the optimized path deviates more from the straight line between the first and last camera position, and the reconstruction error is decreased.

as the camera parameters are perturbed one at a time. By careful bookkeeping of the information matrices only 4 instances need to be computed for each camera instead of all $3(M-2)+1$ of a naïve implementation. This lowers the complexity of computing the Jacobian from $\mathcal{O}(M^2N)$ to $\mathcal{O}(MN)$. Nevertheless, in real-time applications computing the path should take a few seconds at most, and recent SLAM systems track hundreds or thousands of features. It may therefore be necessary to restrict attention to a subset of reconstructed features, e.g. those with the largest uncertainty, when evaluating the cost.

Furthermore, due to the iterative nature of the optimization, the path computation may be aborted before convergence but still yield a good approximation, depending on available time and computational resources.

6.2 Extensions

The assumptions in Section 1 can of course be relaxed. If an initial ML structure estimate is not available, we can either choose to ignore any prior information and initialize the algorithm using optimal triangulation from the most recent images, or simply substitute a non-ML estimate (e.g. from an EKF). If the estimate is good enough, the inverse of the covariance matrix will still be a good approximation to the Fisher information. Even if it's a poor approximation we would expect the optimized paths to yield better reconstruction accuracy than a straight or random one.

The requirement that the camera be oriented toward a particular point is only intended to reduce the dimension of the parameter space. Optimization over the orientations, or other rules for selecting orientation based on camera position and estimated structure could easily be incorporated.

It is also assumed that the camera position and orientation are known to high accuracy when acquiring images. Obviously, this is rarely true in a practical SLAM system, where there may be considerable uncertainty in the robot location. However, the location is usually well-determined relative to nearby, recently observed features, so for short-term local path planning this is a fair approximation. Nevertheless, incorporating the camera uncertainty in the covariance estimation would be straightforward, but would also introduce correlations between features. The information and covariance matrices would no longer be block diagonal, raising the computational load considerably, and the cost function would possibly have to be modified to include the camera location uncertainty. The practical gain of incorporating such information is less clear.

The nature of the optimization scheme makes it easy to incorporate different constraints. For example, in the basic formulation (B.7) points behind the camera contribute the same information as if they were in the corresponding position in front of the camera, resulting in a physically incorrect model of image acquisition. This can be rectified by weighting the information gain from each observation by a function of the point's depth. A suitable function can be seen in Figure B.3, which smoothly diminishes the influence of points as they come too close to the camera. Similar weighting must also be employed to encourage the camera to keep the scene structure in its limited field of view, where points near or outside the image borders are downweighted. The effects of this weighting can be seen in Figure B.2, where the path taken for small values of α would otherwise have passed right through the point cloud.

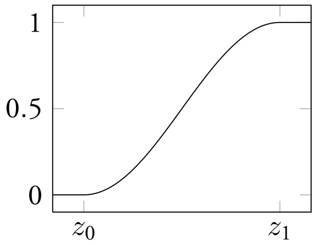


Figure B.3: Differentiable weight function

It is also possible to include penalty constraints on the path curvature, and obstacles in the robot's path can be modeled as a potential field added to the cost function, see Figure B.4.

7 Conclusion

This paper has presented a continuous optimization approach to certain instances of the next best view planning problem, aimed toward application in SLAM systems. Unlike previous algorithms the next best view is chosen with consideration of several expected future observations. While the solutions are only locally optimal, experiments show that reconstruction accuracy is still much improved, at a computational cost linear in the number of cameras and features.

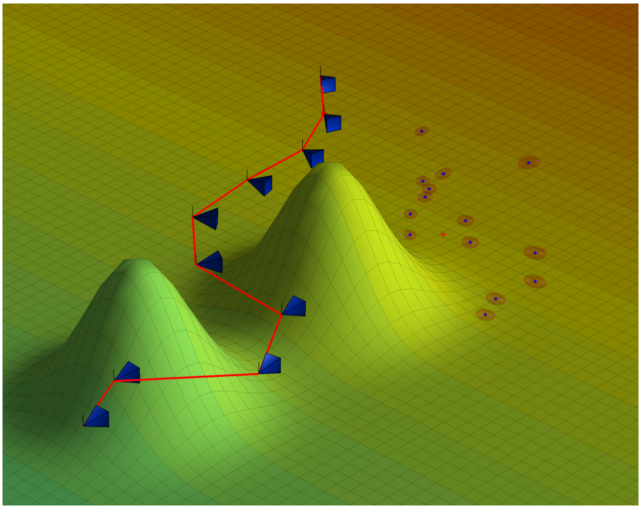


Figure B.4: Obstacle avoidance may be accomplished by adding a smooth potential to the cost function.

References

[1] Tom Botterill, Steven Mills, and Richard Green. Bag-of-words-driven, single-camera simultaneous localization and mapping. *Journal of Field Robotics*, 2010.

[2] Shengyong Chen, Y. F. Li, Jianwei Zhang, and Wanliang Wang. *Active Sensor Planning for Multiview Vision Tasks*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[3] S.Y. Chen and Y.F. Li. Automatic sensor placement for model-based robot vision. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(1):393 –408, feb. 2004.

[4] E. Dunn, J. van den Berg, and J.-M. Frahm. Developing visual sensing strategies through next best view planning. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4001 –4008, oct. 2009.

[5] Enrique Dunn, Gustavo Olague, and Evelyne Lutton. Parisian camera placement for vision metrology. *Pattern Recognition Letters*, 27(11):1209 – 1219, 2006. Evolutionary Computer Vision and Image Understanding.

[6] C.S. Fraser. Network design considerations for non-topographic photogrammetry. *Photo Eng. and Remote Sensing*, 50(8):1115–1126, 1984.

[7] Richard Hartley and Andrew Zisserman. *Multiple View Geometry*. Cambridge University Press, 2003.

[8] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[9] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons Canada, Ltd., 5th edition, 2000.

[10] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8):1178 – 1193, 2009.

[11] Pedro Piniés, Lina María Paz, Dorian Gálvez-López, and Juan D. Tardós. Ci-graph simultaneous localization and mapping for three-dimensional reconstruction of large and complex environments using a multicamera system. *Journal of Field Robotics*, 27(5):561–586, 2010.

[12] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Scale drift-aware large scale monocular slam. In *Proc. Robotics; Science and Systems*, 2010.

[13] Michael Trummer, Christoph Munkelt, and Joachim Denzler. Online next-best-view planning for accuracy optimization using an extended e-criterion. In *Proc. International Conference on Pattern Recognition (ICPR'10)*, volume 0, pages 1642–1645. IEEE Computer Society, 2010.

[14] Stefan Wenhardt, Benjamin Deutsch, Joachim Hornegger, Heinrich Niemann, and Joachim Denzler. An information theoretic approach for next best view planning in 3-d reconstruction. In *Proc. International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 103–106. IEEE Computer Society, 2006.

Paper C

Covariance Propagation and Next Best View Planning for 3D Reconstruction

SEBASTIAN HANER AND ANDERS HEYDEN
Centre for Mathematical Sciences, Lund University

Abstract: This paper examines the potential benefits of applying next best view planning to sequential 3D reconstruction from unordered image sequences. A standard sequential structure-and-motion pipeline is extended with active selection of the order in which cameras are resectioned. To this end, approximate covariance propagation is implemented throughout the system, providing running estimates of the uncertainties of the reconstruction, while also enhancing robustness and accuracy. Experiments show that the use of expensive global bundle adjustment can be reduced throughout the process, while the additional cost of propagation is essentially linear in the problem size.

1 Introduction

Three-dimensional reconstruction from unordered image sequences is a well-studied problem in the computer vision literature, see e.g. [13, 2, 7, 12, 4]. Part of the challenge is that little is known about the input data at the outset in terms of scene coverage or camera calibration. Active sensor planning, on the other hand, is the problem of finding the optimal input data to a reconstruction algorithm, given full control over image acquisition (see [3] for an overview). In the photogrammetry literature this is known as the 'camera network design' problem. For example, in [5] a genetic algorithm is used to search a high-dimensional parameter space of camera placements to find the optimal measurement setup, given a limited number of cameras. In a serial acquisition process, the 'next best view' (NBV) problem asks from which viewpoint to capture the next image, given a partial reconstruction, to minimize some objective such as the reconstruction error. NBV planning is most effective when the user has full control over image acquisition, and has been applied to vision metrology using cameras mounted on robotic arms [16, 15], and autonomous robot exploration [6, 8].

This paper applies view planning to the unordered image reconstruction problem; although we are not free to choose any viewpoint, there is usually a choice between a subset of the images at every step of a sequential algorithm. The aim is to choose the image giving the smallest error, which we approximate as the trace of the camera covariance matrix times the reprojection error. To be able to determine the covariance, it is necessary to know the uncertainty of the observed geometry. In the following sections, it is shown how this is achieved by propagating covariances when resectioning cameras and triangulating points, and how as a side effect the algorithms gain robustness and better approximate the maximum likelihood estimate.

2 Estimation from Uncertain Geometry

The cornerstones of sequential structure-and-motion are triangulation and camera pose estimation. Usually, one attempts to find the maximum likelihood solution given noisy image measurements, but assuming that all other parameters are known exactly. This is of course rarely the case, since points and cameras are triangulated and resectioned using noisy data. Below, we derive algorithms that also take the uncertainty of the 3D structure or camera parameters into account.

2.1 Pose Estimation

Consider the problem of camera pose estimation given N 3D point coordinates X and their measured projections in one image, $\tilde{x}$. Assuming there are errors in the image measurements, the problem is to find the maximum likelihood solution, i.e. the camera parameters θ^* satisfying

$$\theta^* = \arg\max_{\theta} \ \mathcal{L}(\theta)\,, \tag{C.1}$$

where

$$\mathcal{L}(\theta) = \mathcal{L}(\theta \,|\, \tilde{x}, X) = p(\tilde{x} \,|\, \theta, X) \tag{C.2}$$

is the likelihood function. In this formulation it is assumed that the structure parameters X are precisely known. More generally, given a probability distribution of X, the problem is to maximize

$$\mathcal{L}(\theta) = \int_{\mathbb{R}^{3N}} p(\tilde{x} \,|\, \theta, X) p(X)\, dX\,. \tag{C.3}$$

We restrict our attention to the case of Gaussian distributions. Then we have

$$\mathcal{L}(\theta) \propto \int_{\mathbb{R}^{3N}} e^{-\|\tilde{x}-f(X,\theta)\|_R^2} \cdot e^{-\|X-\bar{X}\|_Q^2}\, dX\,, \tag{C.4}$$

where $f(X, \theta)$ is the projection of the points X using camera parameters θ, R the measurement error covariance, Q and $\bar{X}$ are the covariance matrix and mean of the distribution of X and $\|y\|^2_\Sigma = y^\top \Sigma^{-1} y$ the squared Mahalanobis distance. Next, we project the distribution of X onto the image plane, by integrating along the light rays. Formally, for a given θ we parametrize each 3D point by its image projection $x = f(X, \theta)$ and depth ρ, so that

$$\mathcal{L}(\theta) \propto \int_{\mathbb{R}^{2N}} e^{-\|\tilde{x}-x\|^2_R} \left(\int_{\mathbb{R}^N} e^{-\|(x,\rho)-\bar{X}\|^2_Q} \, d\rho \right) dx \, . \tag{C.5}$$

The right-hand factor is a distribution on the $2N$-dimensional generalized image plane, and may be seen as the projection of a random variable, i.e. $f\big(\mathcal{N}(\bar{X}, Q), \theta\big)$. By Taylor expansion about $\bar{X}$, f can be approximated by $\tilde{f}(X, \theta) = f(\bar{X}, \theta) + J(X - \bar{X})$, and for affine functions $\tilde{f}\big(\mathcal{N}(\mu, \Sigma), \theta\big) = \mathcal{N}\big(\tilde{f}(\mu, \theta), J_X \Sigma J_X^\top\big)$ with $J_X = \frac{\partial f}{\partial X}\big|_\theta$. We now have

$$\mathcal{L}(\theta) \underset{\sim}{\propto} \int_{\mathbb{R}^{2N}} e^{-\|\tilde{x}-x\|^2_R} \cdot e^{-\|f(\bar{X},\theta)-x\|^2_{JQJ^\top}} \, dx \, , \tag{C.6}$$

which may be seen as the convolution $(u * v)(\tau) = \int u(x) v(\tau - x) \, dx$ of the Gaussians $u(x) = e^{-\|x-\tilde{x}\|^2_R}$ and $v(x) = e^{-\|x-0\|^2_{JQJ^\top}}$, with $\tau = f(\bar{X}, \theta)$. The convolution of Gaussians is particularly simple, $\mathcal{N}(\tilde{x}, R) * \mathcal{N}(0, JQJ^\top) = \mathcal{N}(\tilde{x}, R + JQJ^\top)$, giving

$$\mathcal{L}(\theta) \underset{\sim}{\propto} e^{-\|\tilde{x}-f(\bar{X},\theta)\|^2_{R+JQJ^\top}} \, . \tag{C.7}$$

Maximizing the likelihood is then equivalent to minimizing

$$-\log \mathcal{L}(\theta) \underset{\sim}{\propto} \|\tilde{x} - f(\bar{X}, \theta)\|^2_{R+JQJ^\top} \, , \tag{C.8}$$

which can be solved using an iteratively reweighted nonlinear least-squares algorithm. In fact, only a minor modification to a standard algorithm for minimizing the reprojection error is required. For example, a Levenberg-Marquardt optimization loop would be modified to

```
while not converged do
    ...
    $W \leftarrow (R + J_X Q J_X^\top)^{-1}$
    $\delta\theta \leftarrow (J_\theta^\top W J_\theta + \lambda I)^{-1} J_\theta^\top W b$
```

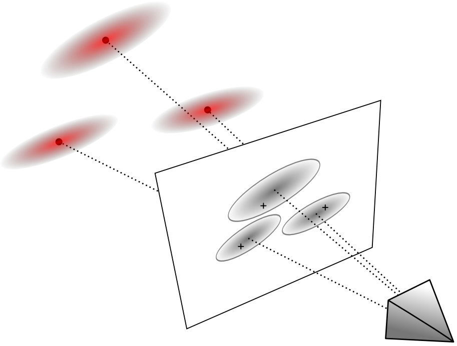


Figure C.1: Resectioning: the uncertainties of the 3D points are projected onto the image plane and convolved with the image measurement uncertainty giving the reprojection error metric. Note that the projections are not necessarily independent; however, in this work inter-point covariances are discarded for computational reasons.

```
    ...
end while
```

where $J_\theta = \frac{\partial f}{\partial \theta}\big|_\theta$ and J_X as above. After convergence, the covariance matrix of the recovered camera parameters θ^* can be estimated by the inverse of the Hessian matrix evaluated at the minimum, $\Sigma_\theta \approx (J_{\theta^*} W^* J_{\theta^*}^\top)^{-1}$ [9, 11].

Of course, a good initial guess is required to start the iterative algorithm, and can be obtained using standard minimal or linear solvers. The general effect of taking the distribution of X into account is to give more weight to well-determined 3D points than uncertain ones when finding the camera pose.

2.2 Triangulation

Handling uncertainty in camera parameters when triangulating 3D structure is completely analogous to the pose estimation case. The linearized problem formulation is to find

$$\theta^* = \arg\min_\theta \|\tilde{x} - f(\theta, \bar{P})\|^2_{R+JSJ^\top}\,, \tag{C.9}$$

where θ now represents the 3D structure, $\bar{P}$ is the mean of the distribution of the cameras with covariance S and $J = \frac{\partial f}{\partial P}\big|_\theta$.

2.3 Complexity

The introduction of the weight matrix W in the algorithms above inevitably incurs extra computational costs. In particular, if the input variables are correlated, W will be a full matrix and the natural sparsity of the problems is lost. To mitigate this, we will assume no correlation between pairs of cameras or points, so that W is block diagonal. Such simplification is also necessary since the full covariance matrix of even a moderately sized reconstruction problem would occupy hundreds of gigabytes of memory. Furthermore, it may not be necessary to recompute W every iteration, since the projection is not expected to change significantly given a good initialization.

3 Covariance Propagation

The proposed algorithms open the possibility of covariance propagation throughout the reconstruction process. Uncertainties in 3D points are transferred to uncertainty in resectioned cameras, which in turn transfer uncertainty to triangulated points, and so on. In this manner, a rough estimate of the covariances is available at any time and can be used, for example, to improve reconstruction accuracy and for next best view planning, which we exploit to reduce error accumulation.

Below we detail a system for 3D reconstruction from unordered image sequences and show the benefits that can be gained.

3.1 Selecting the Seed

In choosing the set of images on which to initialize the reconstruction, we strive for the following: the initial reconstruction should be stable, contain many structure points and it should be near the center of the camera graph (the graph with each camera a vertex and edges between cameras observing common features). The latter is motivated by the fact that error accumulation is a function of the distance from the seed; if the 'degrees of separation' from the seed is kept low, error accumulation can be minimized. We therefore wish to minimize the distance of every camera to the seed. For our purposes we define the center as any vertex of the camera connectivity graph with minimal *farness*, the sum of shortest distances from the node to all others. We define the edge weights of the graph as $1/\max(0, n_c - 4)$, where n_c is the number of observed points common to both cameras. This heuristic, while ignoring the actual two-view geometry, is based on the assumption that cameras sharing many observed points are well-determined relative to each other. The maximum imposes a 5 point overlap threshold, needed to determine relative motion between views. Now, all shortest paths in the graph can be computed and summed for each camera, the k lowest scoring yielding a set of candidate images. For each candidate, an adjacent view with a balance between many common image points and good parallax is selected as in [13], i.e. each pairing is scored according to the proportion of outliers to a homography fit. The top-scoring pair is selected, and standard two-view reconstruction is performed, followed by bundle adjustment.

3.2 Fixing the Gauge

Reconstruction from image measurements only is subject to global translation, rotation and scale ambiguity. Unlike [14], which measured pairwise covariances in local coordinate systems, we need globally referenced covariances and so must compute these for the seed reconstruction. For the covariances to be defined we must fix the gauge, especially the scale, since the dimension of the nullspace of the Hessian matches the number of degrees of freedom of the system. From a theoretical standpoint, taking the pseudoinverse of the unconstrained Hessian is the most satisfying solution [9, 11], however it can be computationally very expensive if the seed views share many points (i.e. > 1000). An alternative approach is to constrain the parameters of the system by adding penalties to the cost function, making the Hessian full rank so it can be inverted without finding an SVD. Different constraints lead to somewhat different estimates of the covariance; one way is to lock the first camera and impose a distance constraint on the mean of the structure points, as was done in [14], or one can simply fix the distance between the first and second camera. The first prior gives results closer to the pseudoinverse, but also destroys the sparsity of the Hessian matrix making inversion more expensive. In cases where the pseudoinverse is too expensive we choose the second option which preserves sparsity.

After fixing the scale, there is still a difficulty in quantifying just how large an uncertainty is, since it must be put in relation to the overall size of the reconstruction. The scale is unknown in the beginning, since there is no guarantee that the distance between the seed cameras is representative of the whole scene. This has implications for the various outlier rejection thresholds used in the reconstruction pipeline.

4 Next Best View Planning

View planning in a sequential reconstruction process aims to actively choose the most favorable sensor configuration (camera position and orientation) to achieve a certain goal, in this case geometric accuracy. In each iteration, we can choose which camera to resection among those observing a sufficient number of triangulated points. Usually, the camera observing the largest number of triangulated points is chosen first. However, if the geometry is such that the pose is poorly determined, triangulations using the image will have larger errors, propagating to subsequently resectioned cameras, etc. It therefore makes sense to minimize the error accumulation in every step. To this end, we propose to select the camera with lowest estimated reconstruction error, by exhaustive search among candidate images. The covariance is computed by first resectioning the camera using a linear or minimal solver and taking the inverse of the Hessian, $\Sigma_{\text{cam}} \approx (J_\theta W J_\theta^\top)^{-1}$ as defined in Section 3. As a scalar measure of reconstruction error we use $\text{trace}(\Sigma_{\text{cam}}) \cdot \epsilon_{\text{rp}}$, where ϵ_{rp} is the mean reprojection error. This turns out to give better results than the covariance alone; a small estimated covariance does not necessarily imply a low reprojection error, and a well-determined camera should ideally have both. Note that the score can be cached for each camera between iterations and need only be recomputed if more points in the camera's view have been triangulated. While the number of views that need to be resectioned in each iteration is dependent on the particular data set and could theoretically grow with the number of triangulated points, in practice this number is found to be approximately constant throughout the reconstruction process and typically between 10 and 50.

5 Reconstruction Pipeline

We apply NBV planning and covariance propagation to the problem of reconstruction from unordered image collections. We will assume that matching and tracking of image features has been performed and is outlier free. If not (as in the last experiment below), the proposed method is easily integrated with outlier detection schemes such as RANSAC. The algorithm is mainly standard fare:

1. Find initial seed views (Section 3.1).
2. Reconstruct and bundle adjust the seed geometry.

3. Compute the covariance of the seed (Section 3.2).

4. Choose a camera to resect following Section 4. Resect using a linear method; if it fails (i.e. large reprojection error or points behind the camera) try an L_∞ formulation [10] instead. If that also fails, choose another camera and try again. Else, refine the camera pose by minimizing (C.8) and store its covariance.

5. Triangulate all points seen by at least two resectioned cameras using a linear method. Compute an approximate uncertainty by evaluating the Hessian of the standard reprojection error and taking the trace of the inverse. Well-determined points, i.e. with low covariance and reprojection error, as specified by thresholds, are kept and further refined by minimizing (C.9). Store the covariance derived from this reprojection error.

6. (Optional) Bundle adjust over all or part of the reconstruction, and update covariances accordingly.

7. If possible, goto 4) and find the next view, else terminate.

In the first experiments below, bundle adjustment is only performed on the seed to demonstrate the efficacy of the approach in reducing error accumulation. In real use, step 6 should performed at regular intervals. The cost of updating the covariances afterwards is a computational bottleneck, which needs to be adressed.

6 Experiments

Figure C.2 shows a simple synthetic example of the dependence on the seed of the propagated covariances. Although the *relative* uncertainties between all cameras remain the same in our linearized Gaussian propagation model, in reality the reconstruction errors depend heavily on the path taken.

Next, the algorithm is applied to a dataset extracted from photos of the 'Spilled blood' church in St. Petersburg. The reconstruction and a comparison with a standard method is shown in Figures C.3 and C.4. The comparison shows the mean standard reprojection error and the ground truth deviation, defined as the mean distance of each triangulated point from its ground truth position, after the two point clouds have been aligned using a Procrustes transformation. The 'ground truth' in this case has been obtained from the system described in

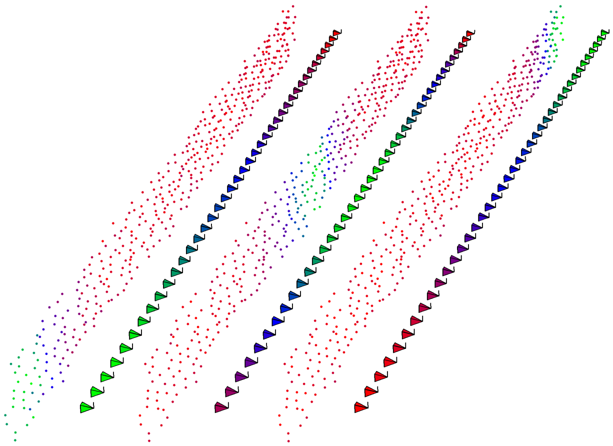


Figure C.2: Toy example of camera array observing a wall illustrating the covariance estimation results depending on which seed is chosen (from left to right, cameras 1 and 2, 19 and 20, 39 and 40). The points and cameras are color coded by the trace of their covariance, with green through blue to red for increasing uncertainty. Choosing the seed in the middle reduces the maximum camera uncertainty with respect to the seed.

[12]. The plain method, without covariance propagation or NBV planning, runs into trouble around iteration 300 and does not manage to resection all cameras, whereas the proposed algorithm does and is generally more robust and accurate. A similar comparison is made for the 'Trafalgar' dataset of [1] in Figure C.5.

Finally, we compare three variants of the proposed algorithm on the Lund Cathedral dataset. The covariance propagation and next best view-planning can be used independently, i.e. the next image can be chosen by the maximum overlap principle while propagating covariances, or the next view can be chosen based on camera uncertainty calculated using zero point covariances, with no propagation. As Figure C.7 shows, using NBV planning alone doesn't work well at all and the process breaks down, like the plain method. Propagation without planning works almost as well as both combined, and is probably the greatest contributing factor.

Next, we apply the proposed algorithm to a more realistic scenario. The 'Örebro castle' dataset is corrupted by replacing 5% of matches with artificially generated outliers. Camera resectioning is preceded by RANSAC outlier detection, and we allow local bundle adjustment to be performed every 20 iterations,

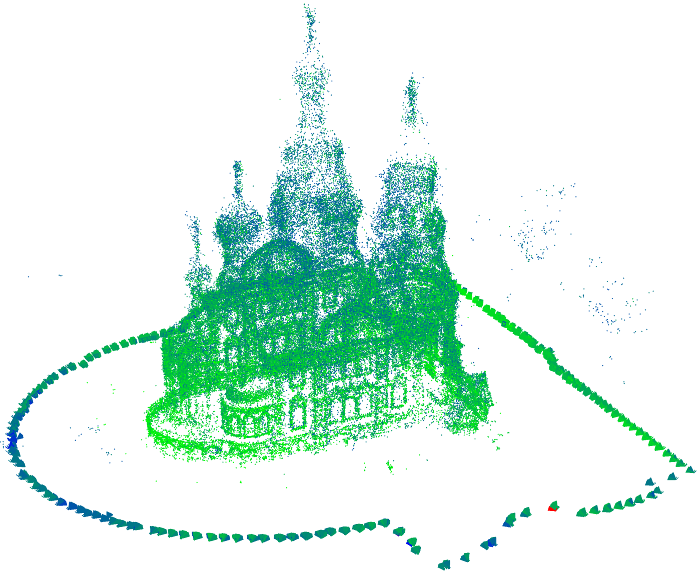


Figure C.3: Resulting point cloud reconstruction of the ‘Spilled blood’ dataset using the proposed algorithm, color coded by estimated covariance. No bundle adjustment has been performed. The dataset has 781 images, 162,521 points and 2,541,736 feature measurements.

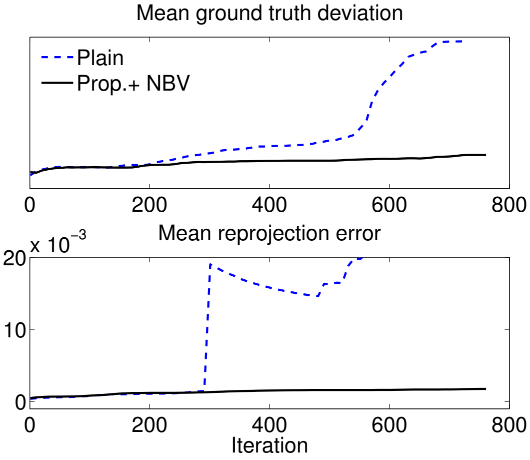


Figure C.4: Comparison between the proposed algorithm and a 'plain' method on the 'Spilled blood' dataset. In the plain method the standard reprojection error is minimized instead, and the next camera is chosen by the maximum overlap principle. Running times were 22 and 13 min respectively. There is no absolute scale on the top graph since it depends on the overall scale of the reconstruction, which is arbitrary.

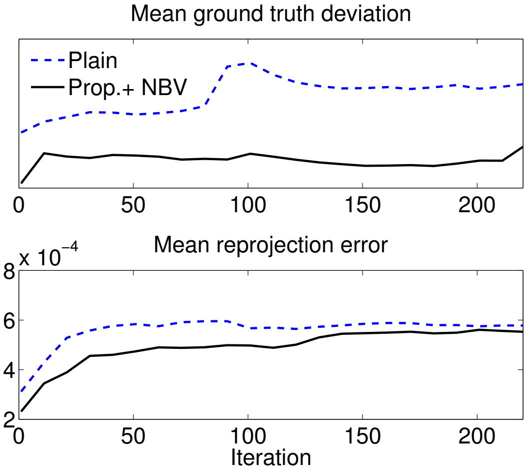


Figure C.5: Results for the Trafalgar dataset (256 images). Running times were 83 and 138 s respectively.

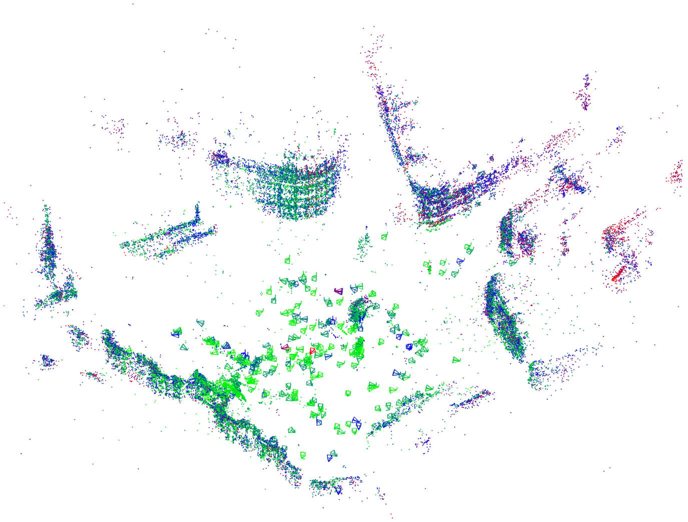
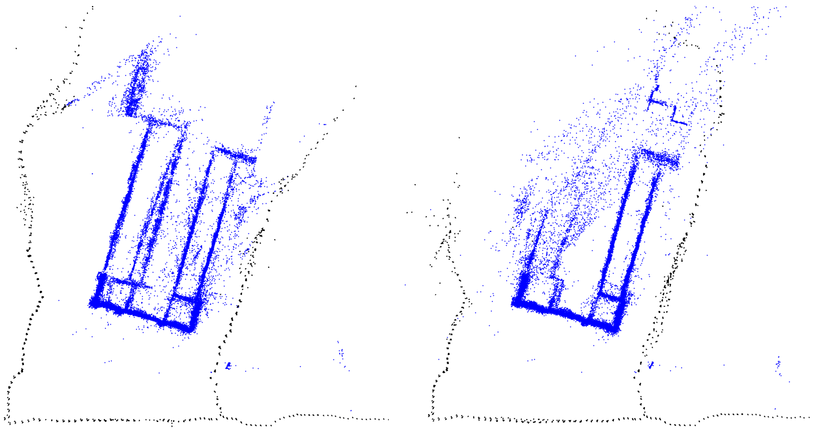


Figure C.6: Resulting point cloud reconstruction of the Trafalgar dataset.

optimizing over the 20 last resectioned cameras and the triangulated points visible in these views. Measurements from other cameras where the points are visible are also included. A robust Huber cost function is used, and measurements with high reprojection error after convergence are deemed outliers and removed. The resulting point cloud reconstructions are shown in Figure C.9. The proposed method produces higher quality output and, surprisingly, is faster in this case. Note that in this experiment the covariances are not updated after bundle adjustment, and still there is marked improvement.

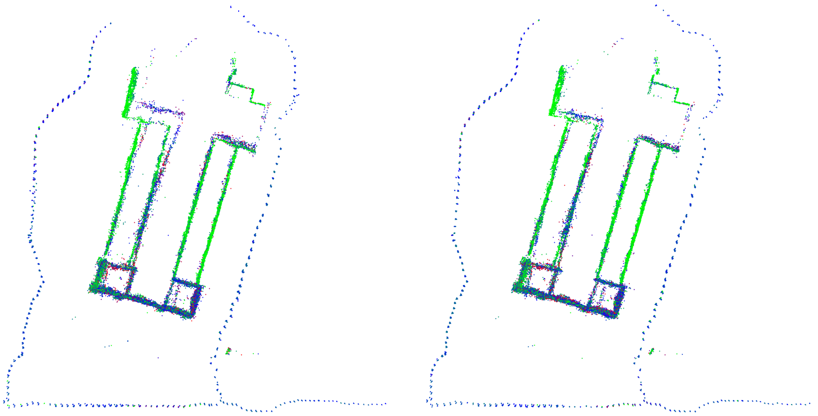
After bundle adjustment, the estimated covariances of the affected parameters are no longer valid and need to be updated. As mentioned in Section 3.2, inverting the whole Hessian matrix of the LM system is infeasible for all but the smallest problems. However, since we only need the diagonal blocks of the covariance, a lot of work can be saved. If the covariance matrix corresponding to the camera parameters only is known, the individual point covariance blocks can be computed very efficiently. The Hessian of a typical BA problem has the particular structure

$$H = \begin{pmatrix} A^\top \Sigma^{-1} A & A^\top \Sigma^{-1} B \\ B^\top \Sigma^{-1} A & B^\top \Sigma^{-1} B \end{pmatrix} = \begin{pmatrix} U & W \\ W^\top & V \end{pmatrix} \tag{C.10}$$



(a) Plain

(b) NBV, no propagation



(c) Propagation only

(d) Propagation and NBV

Figure C.7: Lund Cathedral dataset (1060 images, 45770 points, 408625 projections) reconstructed using the baseline algorithm, next best view-planning only without propagating covariances, propagating covariances but using the maximum overlap principle, and the proposed algorithm, using both NBV planning and propagation.

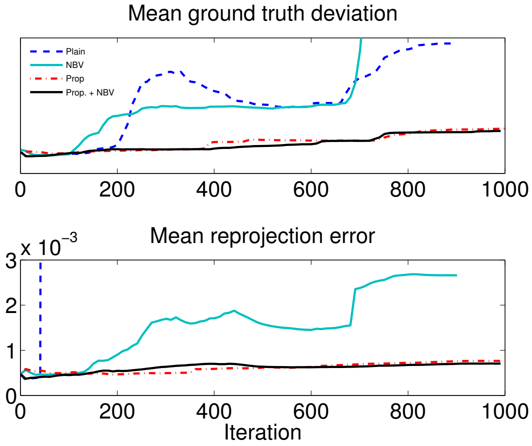
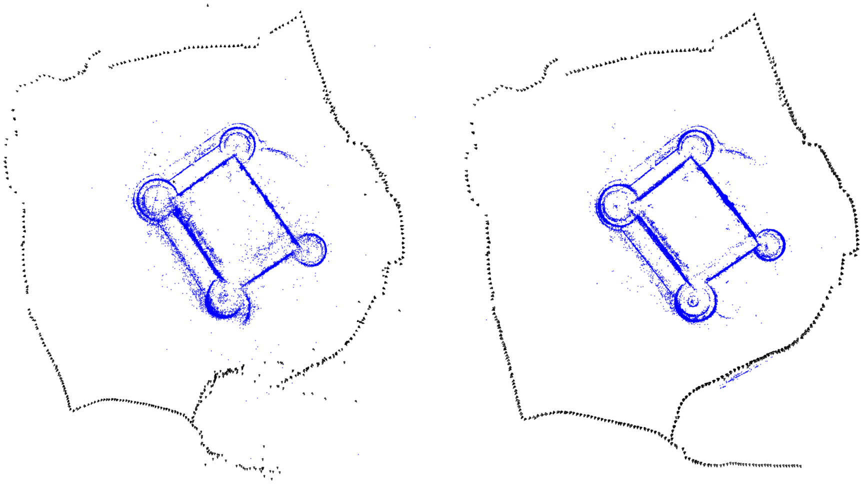


Figure C.8: Error plots for the Lund Cathedral dataset.

where Σ is the measurement noise covariance matrix and U and V are block diagonal with blocks corresponding to the cameras and points respectively. In the fixed-gauge case, the covariance of the camera parameters is then given by $\Sigma_{\mathrm{c}} = (U - WV^{-1}W^{\top})^{-1}$ and the covariance for point i is given by $\Sigma_{\mathrm{p}_i} = V_i^{-1}W_i^{\top}\Sigma_{\mathrm{c}}W_iV_i^{-1} + V_i^{-1}$ where V_i is the corresponding block of V, and W_i the corresponding rows of W (see [9] for details). Exploiting the sparsity of W, the product $W_i^{\top}\Sigma_{\mathrm{c}}W_i$ can be evaluated in time proportional to the number of cameras observing point i. The dominating cost is in practice computing the camera covariance, i.e. forming and inverting the Schur complement, typically of cubic cost in the number of cameras. When this number is low, the covariance update is fast (on the order of a few seconds), but as the reconstruction grows the cost becomes prohibitive and the time would be better spent on bundle adjustment. However, as shown above, updating the covariances is not critical to the performance of the algorithm, and an approximation may be sufficient. For example, experiments have shown that only inverting the diagonal blocks corresponding to individual cameras of the Schur complement gives a reasonable approximation to the full inverse. Nevertheless, efficient updating of the covariances remains an open problem and is the subject of future work.



(a) Standard method, 908 s.

(b) Propagation and NBV planning, 782 s.

Figure C.9: Reconstructions of the Örebro castle dataset with 5% outliers and local bundle adjustment every 20 iterations. The difference in processing time is due to the plain method often failing to resection cameras using the fast linear method and falling back on the slower but more robust L_∞ solver, and more often failing and reattempting triangulation as new views are resectioned.

7 Conclusion

The proposed method increases robustness to errors such as poorly resectioned cameras and poorly triangulated points, reduces error accumulation and also provides estimates of reconstruction accuracy which could be further processed for outlier detection etc. This comes at a cost of up to a twofold increase in running time. However, this cost is practically linear in the problem size, whereas iterated bundle adjustment costs between $\mathcal{O}(n^3)$ and $\mathcal{O}(n^4)$, depending on problem structure. Thus, trading less frequent bundling for covariance propagation and next best view planning should pay off for large problems.

References

[1] Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. Bundle adjustment in the large. In *Proceedings of the 11th European Conference on Computer Vision: Part II*, ECCV'10, pages 29–42, Berlin, Heidelberg, 2010. Springer-Verlag.

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *ICCV*, pages 70–79, 2009.

[3] Shengyong Chen, Y F Li, Jianwei Zhang, and Wanliang Wang. *Active Sensor Planning for Multiview Vision Tasks*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[4] David Crandall, Andrew Owens, Noah Snavely, and Daniel P Huttenlocher. Discrete-Continuous Optimization for Large-Scale Structure from Motion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3001–3008. IEEE, June 2011.

[5] Enrique Dunn, Gustavo Olague, and Evelyne Lutton. Parisian camera placement for vision metrology. *Pattern Recognition Letters*, 27(11):1209–1219, 2006.

[6] Enrique Dunn, Jur van den Berg, and Jan-Michael Frahm. Developing visual sensing strategies through next best view planning. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4001–4008, 2009.

[7] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building Rome on a Cloudless Day. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Proceedings of the 11th European Conference on Computer Vision, ECCV'10*, volume 6314 of *Lecture Notes in Computer Science*, chapter 27, pages 368–381. Springer, 2010.

[8] Sebastian Haner and Anders Heyden. Optimal view path planning for visual SLAM. In Anders Heyden and Fredrik Kahl, editors, *Image Analysis*, volume 6688 of *Lecture Notes in Computer Science*, pages 370–380. Springer Berlin / Heidelberg, 2011.

[9] Richard Hartley and Andrew Zisserman. *Multiple View Geometry*. Cambridge University Press, 2003.

[10] Fredrik Kahl and Richard Hartley. Multiple View Geometry Under the L_∞ Norm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1603–1617, 2008.

[11] Daniel D Morris. *Gauge Freedoms and Uncertainty Modeling for 3D Computer Vision*. PhD thesis, Carnegie Mellon University, 2001.

[12] Carl Olsson and Olof Enqvist. Stable structure from motion for unordered image collections. In *Proceedings of the 17th Scandinavian conference on Image analysis*, SCIA'11, pages 524–535, Berlin, Heidelberg, 2011. Springer-Verlag.

[13] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the World from Internet Photo Collections. *International Journal of Computer Vision*, 80(2):189–210, nov 2007.

[14] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, jun 2008.

[15] Michael Trummer, Christoph Munkelt, and Joachim Denzler. Online Next-Best-View Planning for Accuracy Optimization Using an Extended E-Criterion. In *Proc. International Conference on Pattern Recognition (ICPR'10)*, volume 0, pages 1642–1645. IEEE Computer Society, 2010.

[16] Stefan Wenhardt, Benjamin Deutsch, Joachim Hornegger, Heinrich Niemann, and Joachim Denzler. An Information Theoretic Approach for Next Best View Planning in 3-D Reconstruction. In *Proc. International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 103–106. IEEE Computer Society, 2006.

Paper D

Decomposable Bundle Adjustment using a Junction Tree

PEDRO PINIÉS[1], LINA M. PAZ[1], SEBASTIAN HANER[2], ANDERS HEYDEN[2]
[1]*Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza*
[2]*Centre for Mathematical Sciences, Lund University*

Abstract: Bundle adjustment is an integral component of most multiple-view reconstruction algorithms in computer vision. The *sparse bundle adjustment* algorithm exploits sparsity patterns in the problem structure and input data for efficient solution. The dominating cost depends on the fill-in of the reduced camera matrix whose pattern is known as the secondary structure of the problem. In centered object applications, where a large number of images are taken in a small area, the camera matrix obtained when points are eliminated is dense. On the other hand, visual mapping systems where long trajectories are traversed yield sparse matrices. In this paper, we propose a *decomposable bundle adjustment* method which naturally adapts to the fill-in pattern of the camera matrix improving the performance of visual mapping systems. The algorithm decomposes the normal equations into small subsystems which are ordered in a junction tree structure. To solve the original system, local factorizations of the small dense matrices are passed between clusters in the tree. The proposed algorithm has been tested on simulated and real data for different environment configurations, showing good performance.

1 Introduction

Bundle adjustment (BA) is a non-linear least squares technique used for the refinement of cameras and 3D structure parameters from a set of images. The standard reference is the work presented in [14]. Efficient solutions can be achieved by exploiting the sparseness of the problem, known as *primary structure* [5], where constraints just exist between points and cameras. The key idea is to eliminate point elements from the system and solve for the reduced camera matrix instead. For dense camera matrices the computational cost of BA is $\mathcal{O}(n_c^3)$ where n_c is the number of cameras. A well optimized sparse bundle adjustment (SBA) implementation is presented in [10]. In [9] the secondary structure of the system (connections between cameras) is exploited for visual mapping applications where long trajectories are traversed. In this case, the camera matrix becomes a sparse system [6] which is solved using a sparse Cholesky solver (CHOLMOD [3]) along with an engineering approach to index sparse matrices efficiently. The final solution obtained is a maximum a posteriori (MAP) estimate of the camera poses and the environment structure.

In this paper, we propose a *decomposable bundle adjustment* (DBA) method to efficiently solve the reduced camera matrix system, obtaining at the same time the probability marginals of the variables involved. This algorithm decomposes the original system into smaller dense submatrices which are ordered in a tree. To decompose the normal equations in an ordered and correct way we use a *junction tree* structure [2]. The result is an exact bundle adjustment algorithm with the following main advantages:

- We do not have to choose between dense (e.g. LAPACK Cholesky) or sparse (e.g. CHOLMOD) linear solvers depending on the secondary structure of the problem. Since the camera matrices in the tree are dense we can always use dense solvers which are very well optimized.
- When the optimization ends, the small Hessian in each node of the tree represents the marginal information matrix of the camera and point elements in that cluster.
- Matrices present in different branches of the tree can be treated independently which means that parallel or distributed solutions can be implemented to efficiently solve bundle adjustment problems.

Using a junction tree structure to decompose and solve a linearized system of equations is closely related to the work presented in [4] and the recent and quite interesting Bayes Tree structure proposed in [8]. The main differences with these techniques are twofold: First, these methods are based on factorizing the measurement Jacobian using QR matrix decompositions and, as a consequence, they do not obtain clique marginals of the clusters but a MAP estimate. In our case the local information matrices of the cameras are decomposed using Schur complement operations, in order to calculate the belief propagation messages, which allows us to easily recover the covariance marginals for each cluster. These marginals are important to check the decisions made during data association. Second, instead of using a general ordering algorithm to solve the sparse system we take advantage of the special structure of the complete information matrix in pure visual scenarios. In visual applications the number of elements in the structure drastically outnumber the camera poses. The junction tree is built according to the reduced camera matrix, which is the costly system to solve in BA, whereas the points and observations are added afterwards. This way each tree cluster replicates in a smaller scale the same typical primary structure of BA. As a consequence, solving for the points at each tree node requires linear time.

The paper is organized as follows. In Section 2 we explain the concept of decomposable systems. Then, in Section 3 we show that the normal equations solved in each iteration of BA are decomposable. In Section 4 we present the proposed DBA algorithm with an explanation of its main protocol to perform computations along the junction tree. The results on simulated experiments and real data are presented in Section 5. The potential of the DBA algorithm to be parallelized is treated in Section 6. Finally we draw the conclusions and discuss future work in Section 7.

2 Decomposable Systems

This subsection is mainly based on [12]. A linear system $\mathbf{U}\mathbf{x} = \mathbf{u}$ is called decomposable if there is no direct relation between some of its variables. For example, the following system

$$\begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_{12} & \mathbf{0} \\ \mathbf{U}_{21} & \mathbf{U}_2 & \mathbf{U}_{23} \\ \mathbf{0} & \mathbf{U}_{32} & \mathbf{U}_3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \end{bmatrix} \tag{D.1}$$

is decomposable since there is no direct relation between variables $\mathbf{x}_1$ and $\mathbf{x}_3$ ($\mathbf{U}_{13} = \mathbf{0}$ and $\mathbf{U}_{31} = \mathbf{0}$). The advantage of a decomposable system is that it can be split into smaller subsystems. Equation (D.1) can be decomposed as $\mathbf{U}'\mathbf{x}' = \mathbf{u}'$ and $\mathbf{U}''\mathbf{x}'' = \mathbf{u}''$ given by:

$$\begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_{12} \\ \mathbf{U}_{21} & \mathbf{U}_2' \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2' \end{bmatrix} \tag{D.2}$$

$$\begin{bmatrix} \mathbf{U}_2'' & \mathbf{U}_{23} \\ \mathbf{U}_{32} & \mathbf{U}_3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_2'' \\ \mathbf{u}_3 \end{bmatrix} \tag{D.3}$$

such that $\mathbf{U}_2 = \mathbf{U}_2' + \mathbf{U}_2''$ and $\mathbf{u}_2 = \mathbf{u}_2' + \mathbf{u}_2''$. Let us express this decomposition as $\mathbf{U} = \mathbf{U}' \oplus \mathbf{U}''$ and $\mathbf{u} = \mathbf{u}' \oplus \mathbf{u}''$ where $\oplus$ stands for the generalized sum of matrix or vector elements according to their index.

In order to obtain a solution for the original system, (D.2) and (D.3) cannot be solved independently since both systems are linked by $\mathbf{x}_2$. Instead the following protocol can be applied:

1. Perform Gaussian elimination in (D.2) to eliminate the influence of $\mathbf{x}_1$ on the common element $\mathbf{x}_2$, obtaining

$$\begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_{12} \\ \mathbf{0} & \mathbf{U}_2'^m \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2'^m \end{bmatrix} \tag{D.4}$$

   where

$$\mathbf{u}_2'^m = \mathbf{u}_2' - \mathbf{U}_{21}\mathbf{U}_1^{-1}\mathbf{u}_1 \tag{D.5}$$

$$\mathbf{U}_2'^m = \mathbf{U}_2' - \mathbf{U}_{21}\mathbf{U}_1^{-1}\mathbf{U}_{12}\,. \tag{D.6}$$

   Matrix $\mathbf{U}_2'^m$ is known as the Schur complement of $\mathbf{U}_1$.

2. Add this marginal to (D.3), i.e. $\mathbf{U}'' \oplus \mathbf{U}_2'^m$ and $\mathbf{u}'' \oplus \mathbf{u}_2'^m$:

$$\begin{bmatrix} \mathbf{U}_2'' + \mathbf{U}_2'^m & \mathbf{U}_{23} \\ \mathbf{U}_{32} & \mathbf{U}_3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_2'' + \mathbf{u}_2'^m \\ \mathbf{u}_3 \end{bmatrix} \tag{D.7}$$

3. Solve for $\mathbf{x}_2$ and $\mathbf{x}_3$. To solve for $\mathbf{x}_1$ just apply back substitution in (D.4):

$$\mathbf{U}_1\mathbf{x}_1 = \mathbf{u}_1 - \mathbf{U}_{12}\mathbf{x}_2 \tag{D.8}$$

It can be shown that this procedure is equivalent to eliminating $\mathbf{x}_1$ in (D.1), solving for $\mathbf{x}_2$ and $\mathbf{x}_3$ and then applying back substitution to obtain $\mathbf{x}_1$.

3 Decomposable Structure of Bundle Adjustment

Given a set of measured image feature locations $\mathbf{z}_{ij}$, the goal of bundle adjustment is to find 3D point positions P_j and camera parameters C_i that minimize the L_2 norm of the reprojection error $\mathbf{r}(\mathbf{x})$ [7]. Using a first order approximation for the residuals yields a linearized least squares problem,

$$\min_{\mathbf{x}} \|\mathbf{r}(\mathbf{x} + \delta\mathbf{x})\|^2 \approx \|\mathbf{r}(\mathbf{x}) + \mathbf{J}\delta\mathbf{x}\|^2 \tag{D.9}$$

where $\mathbf{x}$ comprises the set of cameras and points and $\mathbf{J}$ is the Jacobian of the residual. At each iteration, the step $\delta\mathbf{x}$ is calculated by solving the normal equations

$$(\mathbf{J}^\top\mathbf{J} + \mathbf{I}\lambda)\delta\mathbf{x} = -\mathbf{J}^\top\mathbf{r}(\mathbf{x})\,. \tag{D.10}$$

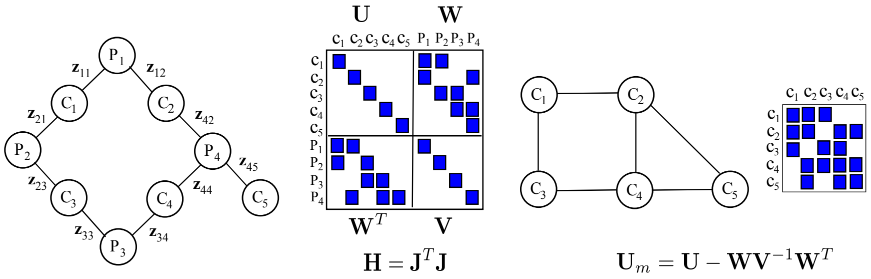


Figure D.1: Bundle adjustment example. (Left) The relation between cameras and points is given by the undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ where $\mathcal{V}$ is the set of vertices representing cameras and points and $\mathcal{E}$ is the edge set of $\mathcal{G}$. Observe that edge $(C_i, P_j) \in \mathcal{E}$ if there is an observation $\mathbf{z}_{ij}$ of point j by camera i. (Right) Camera graph $\mathcal{G}_C(\mathcal{V}_C,\mathcal{E}_C)$ obtained when points are eliminated. Observe that cameras get connected if they observe a common point. Matrices $\mathbf{H}$ and $\mathbf{U}^m$ depicted in the example can be interpreted as the corresponding adjacency matrices of the graphs.

The damping parameter λ is commonly added to ensure a decreasing step when Newton directions are rejected [11]. The Jacobian $\mathbf{J}$ can be partitioned into camera and point parts $[\mathbf{J}_C, \mathbf{J}_P]$. For an observation $\mathbf{z}_{ij}$ the corresponding element of $\mathbf{J}_C$ is given by $J_{C_{ij}} = \partial\mathbf{r}(C_i, P_j)/\partial C_i$ whereas for $\mathbf{J}_P$ we have $J_{P_{ij}} = \partial\mathbf{r}(C_i, P_j)/\partial P_j$.

Figure D.1 left shows a simple example that will help us study the special structure of BA. Since each row in $\mathbf{J}$ only relates a camera and point pair, the information matrix and vector in (D.10) inherit a special fill-in structure called *primary structure* represented by the sparse block diagonal camera and point matrices $\mathbf{U}$ and $\mathbf{V}$. Matrix $\mathbf{W}$ represents the pairwise relation of cameras and points.

The sparse bundle adjustment algorithm in [5] takes advantage of the primary structure to efficiently handle the normal equations. The key idea is to eliminate the points obtaining a smaller camera system $\mathbf{U}^m$, solve for the cameras and then apply back substitution to solve for the points. When $\mathbf{U}^m$ is sparse, the camera system is commonly solved using CHOLMOD routines [3] in addition to reordering strategies like COLAMD while in the dense case, Cholesky factorization is implemented using LAPACK routines.

After Gaussian elimination the reduced camera matrix $\mathbf{U}^m$ is given by:

$$\begin{bmatrix} C_{1|1}+C_{1|2} & C_{12|1} & C_{13|2} & \mathbf{0} & \mathbf{0} \\ C_{12|1}^\top & C_{2|1}+C_{2|4} & \mathbf{0} & C_{24|4} & C_{25|5} \\ C_{13|2}^\top & \mathbf{0} & C_{3|2}+C_{3|3} & C_{34|3} & \mathbf{0} \\ \mathbf{0} & C_{24|4}^\top & C_{34|3}^\top & C_{4|3}+C_{4|4} & C_{45|4} \\ \mathbf{0} & C_{25|5}^\top & \mathbf{0} & C_{45|4}^\top & C_{5|4} \end{bmatrix} \tag{D.11}$$

Figure D.1 right represents the new camera graph $\mathcal{G}_C(\mathcal{V}_C, \mathcal{E}_C)$ obtained. The notation used for the elements in (D.11) is chosen as a mnemotechnic rule to easily calculate the matrix $\mathbf{U}^m$ from graph $\mathcal{G}$ in Figure D.1 left. A diagonal element $C_{i|j}$ stands for the information gained about camera i due to its relation with point j. Off-diagonal elements $C_{ik|j}$ represent the indirect relation between cameras C_i and C_k that appears when a common observed point P_j is eliminated. These elements are calculated as follows:

$$C_{i|j} = U_{ij} - W_{ij} V_j^{-1} W_{ij}^\top \tag{D.12}$$

$$C_{ik|j} = -W_{ij} V_j^{-1} W_{jk}^\top \tag{D.13}$$

where $U_{ij} = J_{C_{ij}}^\top J_{C_{ij}}$, $W_{ij} = J_{C_{ij}}^\top J_{P_{ij}}$ and $V_j = \sum_{i \setminus (i,j) \in \mathcal{E}_\mathcal{G}} J_{P_{ij}}^\top J_{P_{ij}}$.

Expression (D.11) reflects that for visual mapping scenarios, BA produces camera matrices $\mathbf{U}^m$ that can be decomposable. In fact, for long camera trajectories the camera matrix becomes increasingly sparse.

4 Decomposable Bundle Adjustment using a Junction Tree

We propose a decomposable algorithm for BA based on the operations explained in Section 2. To deal with complex systems we use a junction tree structure that allows us to automatically decompose and solve the original system in the correct order.

Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, a junction tree for $\mathcal{G}$ is an undirected graph $\mathcal{T}(\mathcal{N}, \mathcal{E}_\mathcal{T})$ with the following properties:

- Each vertex $\mathcal{N}_i \in \mathcal{N}$ is a subset of $\mathcal{V}$. These vertices are called clusters.
- For each edge $(\mathcal{V}_i, \mathcal{V}_j) \in \mathcal{E}$ there is some cluster $\mathcal{N}_k$ containing both $\mathcal{V}_i$ and $\mathcal{V}_j$.

Algorithm 1 $[\mathbf{x}, \mathcal{T}] = \textbf{DBA}(\mathbf{x}_0, \mathbf{z})$

```
Initialize $\lambda$;
$\mathbf{x} = \mathbf{x}_0$; $k = 0$; stop= 0;
$\mathcal{T} = \textbf{buildJT}(\mathbf{x}, \mathbf{z})$;
$\mathbf{r} = \textbf{calculateJacobiansAndResiduals}(\mathcal{T})$;
while (not stop) and ($k <$maxIter) do
    $k = k + 1$;
    $\textbf{collectEvidence}(\mathcal{T} \rightarrow \mathcal{N}_{root}, \lambda)$;
    $[\delta_{\mathbf{x}}, \mathbf{r}_{new}] = \textbf{distributeEvidence}(\mathcal{T} \rightarrow \mathcal{N}_{root}, \lambda)$;
    if $\|\delta_{\mathbf{x}}\| \leq \epsilon_1(\|\mathbf{x}\| + \epsilon_1)$ then
        stop= 1;
    else
        if $\|\mathbf{r}_{new}\|^2 < \|\mathbf{r}\|^2$ then
            stop = $(\|\mathbf{r}\| - \|\mathbf{r}_{new}\| < \epsilon_2\|\mathbf{r}\|)$;
            $\mathbf{x} = \mathbf{x} + \delta_{\mathbf{x}}$;
            $\textbf{updateJT}(\mathcal{T}, \mathbf{x})$;
            $\mathbf{r} = \textbf{calculateJacobiansAndResiduals}(\mathcal{T})$;
            Decrease $\lambda$;
        else
            Increase $\lambda$;
        end if
    end if
end while
```

- For any two clusters $\mathcal{N}_i$ and $\mathcal{N}_j$, all clusters on the unique path joining them contain the intersection $\mathcal{N}_i \cap \mathcal{N}_j$. For each edge $(\mathcal{N}_i, \mathcal{N}_j) \in \mathcal{E}_{\mathcal{T}}$ we associate a separator $\mathcal{S}_{ij} = \mathcal{N}_i \cap \mathcal{N}_j$.

Our proposed method is shown in Algorithm 1. Before the iteration loop starts the function **buildJT** builds a junction tree that decomposes the original system by distributing cameras, points and measurements to each cluster. Then, for each iteration, a two-way message passing protocol [1] is implemented to solve the normal equations using the basic operations explained in Section 2. In the first pass, the **collectEvidence** function propagates, from the leaves up to the root, information of common elements between a cluster and its parent using Gaussian elimination. In the second pass, the **distributeEvidence** function starts solving for variables at the root and then goes down to the leaves performing back substitution to solve for the remaining cluster variables. In the following subsections we analyze in more detail these functions.

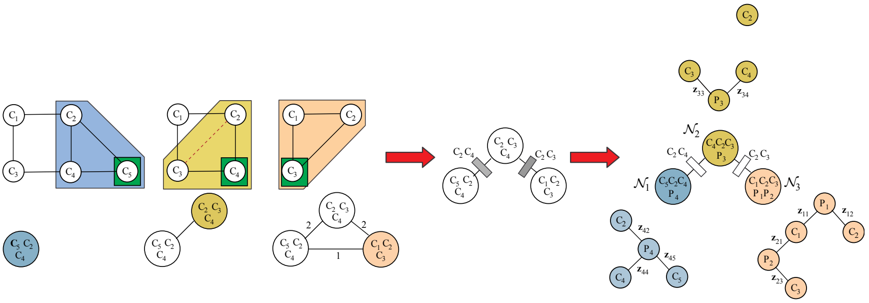


Figure D.2: Building a junction tree from the camera graph $\mathcal{G}_C$. (Left) The camera graph at each step of the vertex elimination process is shown. The elimination order chosen is C_5, C_4, C_3. The square surrounding a vertex shows the camera to be eliminated. The shadowed area represents the cluster $\mathcal{N}_i$ created by the eliminated node and its neighbors. We can also see the cluster graph created during the elimination process. Note that after camera C_3 is eliminated the clusters that would be obtained eliminating C_1 or C_2 would not be maximal and therefore are not shown. (Middle) Junction tree obtained from $\mathcal{G}_C$. (Right) Final junction tree created after assigning points and measurements to the corresponding camera clusters. For each cluster the associated undirected graph of camera and points relations is also represented. Notice that points P_j and observations $\mathbf{z}_{ij}$ are uniquely distributed among the tree clusters.

4.1 Building a Junction Tree

Using $\mathbf{z}$ and the mnemotechnic rule explained in Section 3 we build the adjacency matrix of cameras $\mathcal{G}_C$. Then a junction tree for $\mathcal{G}_C$ is built applying the following steps:

1. Choose an ordering for $\mathcal{V}_C$ and eliminate vertices. We assign each eliminated vertex and its neighbors to a cluster $\mathcal{N}_i$. After a node is eliminated its neighbor nodes in $\mathcal{G}_C$ get connected.

2. Build a graph with the maximal clusters, i.e. those that are not contained in other clusters.

3. Weight each edge of the cluster graph $(\mathcal{N}_i, \mathcal{N}_j)$ with the number of common elements between $\mathcal{N}_i$ and $\mathcal{N}_j$. The junction tree is given by the maximum weight spanning tree of the cluster graph.

Different junction trees can be obtained for the same graph depending on the elimination order and the maximum spanning tree chosen. In this paper we use

COLAMD [3] for the elimination ordering. Figure D.2 left shows an example of the construction of a junction tree. Note that a camera can belong to multiple clusters.

Once the cameras have been distributed we uniquely assign each point P_j and its corresponding measurements $\mathbf{z}_{*j}$ to a cluster of the tree. The cluster must contain all the cameras from which the point is observed. If more than one cluster fulfills the condition the point is assigned to the node with fewer elements. Figure D.2 right shows the final junction tree obtained for our example.

4.2 Collect Evidence

An implementation of the function **collectEvidence** is shown in Algorithm 2. For each cluster in the tree this algorithm recursively collects information from its children. To facilitate the explanation we will make use of Figure D.3 left. In the example $\mathcal{N}_2$ is the root of the tree with children $\mathcal{N}_1$ and $\mathcal{N}_3$. Line 4 of the algorithm eliminates points from the current cluster system obtaining the reduced camera matrix $\mathbf{U}^m_{\mathcal{N}_i}$. The camera marginals of the example can be calculated from Figure D.2 right using the mnemotechnic rule:

$\mathbf{U}^m_{\mathcal{N}_1}$	$C_{2\|4}$	$C_{24\|4}$	$C_{25\|4}$
	$C^\top_{24\|4}$	$C_{4\|4}$	$C_{45\|4}$
	$C^\top_{25\|4}$	$C^\top_{45\|4}$	$C_{5\|4}$
$\mathbf{U}^m_{\mathcal{N}_2}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
	$\mathbf{0}$	$C_{3\|3}$	$C_{34\|3}$
	$\mathbf{0}$	$C^\top_{34\|3}$	$C_{4\|3}$
$\mathbf{U}^m_{\mathcal{N}_3}$	$C_{1\|1} + C_{1\|2}$	$C_{12\|1}$	$C_{13\|2}$
	$C^\top_{12\|1}$	$C_{2\|1}$	$\mathbf{0}$
	$C^\top_{13\|2}$	$\mathbf{0}$	$C_{3\|2}$

Observe that the original camera matrix in (D.11) has been decomposed as $\mathbf{U}^m = \mathbf{U}^m_{\mathcal{N}_1} \oplus \mathbf{U}^m_{\mathcal{N}_2} \oplus \mathbf{U}^m_{\mathcal{N}_3}$.

Suppose that we are in cluster $\mathcal{N}_1$. In line 15 we obtain the cameras $\mathbf{s} = (C_2, C_4)$ in common with its parent $\mathcal{N}_2$. In line 18 we apply Gaussian elimination to obtain the Schur complement $\mathbf{U}^m_{\mathcal{N}_1}(C_2, C_4)$ of C_5. The same procedure is carried out for cluster $\mathcal{N}_3$ obtaining the marginal $\mathbf{U}^m_{\mathcal{N}_3}(C_2, C_3)$ of the elements in common with $\mathcal{N}_2$. Finally line 9 adds these matrices to the parent camera sys-

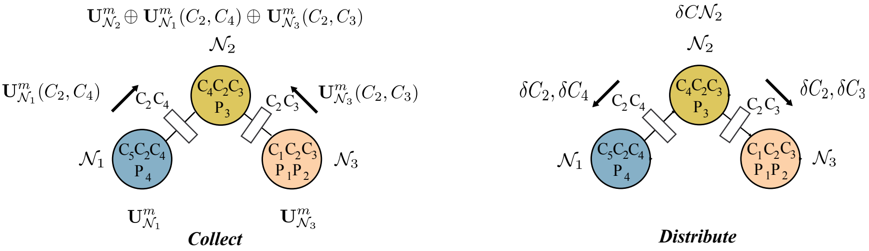

$\mathbf{U}^m_{\mathcal{N}_2} \oplus \mathbf{U}^m_{\mathcal{N}_1}(C_2, C_4) \oplus \mathbf{U}^m_{\mathcal{N}_3}(C_2, C_3)$
$\mathcal{N}_2$
$\mathbf{U}^m_{\mathcal{N}_1}(C_2, C_4)$
C₂C₄
C₄C₂C₃ P₃
C₂C₃
$\mathbf{U}^m_{\mathcal{N}_3}(C_2, C_3)$
$\mathcal{N}_1$
C₅C₂C₄ P₄
C₁C₂C₃ P₁P₂
$\mathcal{N}_3$
$\mathbf{U}^m_{\mathcal{N}_1}$
$\mathbf{U}^m_{\mathcal{N}_3}$
Collect
$\delta C \mathcal{N}_2$
$\mathcal{N}_2$
$\delta C_2, \delta C_4$
C₂C₄
C₄C₂C₃ P₃
C₂C₃
$\delta C_2, \delta C_3$
$\mathcal{N}_1$
C₅C₂C₄ P₄
C₁C₂C₃ P₁P₂
$\mathcal{N}_3$
Distribute

Figure D.3: (Left) Collect function. Information about cameras C_2 and C_4 in cluster $\mathcal{N}_1$ are sent to its parent $\mathcal{N}_2$ by means of their marginal $\mathbf{U}^m_{\mathcal{N}_1}(C_2, C_4)$. The same procedure is performed by cluster $\mathcal{N}_3$. (Right) Distribute function. Root cluster $\mathcal{N}_2$ sends updated information back to its children.

tem, $\mathbf{U}^m_{\mathcal{N}_2} = \mathbf{U}^m_{\mathcal{N}_2} \oplus \mathbf{U}^m_{\mathcal{N}_1}(C_2, C_4) \oplus \mathbf{U}^m_{\mathcal{N}_3}(C_2, C_3)$. Notice that we are basically following steps 1 and 2 explained in Section 2 to solve a decomposed system.

4.3 Distribute Evidence

The **distributeEvidence** function is shown in Algorithm 3. Since the root contains all the information required, line 3 directly solves for the correction $\delta C_{\mathcal{N}_2}$ of cameras in $\mathcal{N}_2$. For the rest of the clusters line 9 sends back the solution of the common cameras to its children whereas line 10 performs a back substitution operation to solve for the remaining camera elements in the cluster. For example, $\mathcal{N}_1$ receives the correction for cameras (C_2, C_4) from its parent $\mathcal{N}_2$ and then calculates the correction for C_5 using back substitution. For all clusters in the tree, line 13 performs back substitution to calculate point corrections. Finally, in line 17 the function is recursively called to traverse the whole tree down to the leaves.

5 Results

The experimental setup is based on a MATLAB comparison between the DBA and a standard SBA implementation. For the SBA we use the MATLAB built-in CHOLMOD library to solve sparse camera matrices. In order to speed up the execution and provide the same resource conditions common operations of both algorithms (i.e. computing residuals, Jacobians, and the inverse of block diagonal matrices) have been implemented in C code using MEX functions. Both SBA

Algorithm 2 **collectEvidence**($\mathcal{T} \rightarrow \mathcal{N}_i, \lambda$)

```
$\mathbf{u}_i = \mathbf{J}_{ci}^{\top}\mathbf{r}_i;\ \mathbf{v}_i = \mathbf{J}_{pi}^{\top}\mathbf{r}_i;$
$\mathbf{U}_i = \mathbf{J}_{ci}^{\top}\mathbf{J}_{ci};\ \mathbf{V}_i = \mathbf{J}_{pi}^{\top}\mathbf{J}_{pi};\ \mathbf{W}_i = \mathbf{J}_{ci}^{\top}\mathbf{J}_{pi};$
$\mathbf{V}_i = \mathbf{V}_i + \lambda\mathbf{I};$
$[\mathbf{u}_i^m, \mathbf{U}_i^m] = \mathbf{gaussElim}(\mathbf{u}_i, \mathbf{v}_i, \mathbf{U}_i, \mathbf{V}_i, \mathbf{W}_i)$

for $j = \mathbf{children}(\mathcal{T} \rightarrow \mathcal{N}_i)$ do
    collectEvidence($\mathcal{T} \rightarrow \mathcal{N}_j, \lambda$)
    {$\mathbf{s}$: elements in $\mathcal{S}_{ji}$ separator}
    $\mathbf{U}_i^m(\mathbf{s},\mathbf{s}) = \mathbf{U}_i^m(\mathbf{s},\mathbf{s}) + \mathbf{U}_j^m(\mathbf{s},\mathbf{s})$
    $\mathbf{u}_i^m(\mathbf{s}) = \mathbf{u}_i^m(\mathbf{s}) + \mathbf{u}_j^m(\mathbf{s})$
end for

if $\mathcal{T} \rightarrow \mathcal{N}_i \neq \mathcal{T} \rightarrow \mathcal{N}_{root}$ then
    $k = \mathbf{parent}(\mathcal{T} \rightarrow \mathcal{N}_i)$
    {$\mathbf{s}$: elements in $\mathcal{S}_{ik}$ separator}
    {$\mathbf{q}$: camera elements not in $\mathcal{S}_{ik}$}
    $\mathbf{U}_i^m(\mathbf{q},\mathbf{q}) = \mathbf{U}_i^m(\mathbf{q},\mathbf{q}) + \lambda\mathbf{I};$
    $[\mathbf{u}_i^m(\mathbf{s}), \mathbf{U}_i^m(\mathbf{s},\mathbf{s})] =$
    $\mathbf{gaussElim}(\mathbf{u}_i^m(\mathbf{s}), \mathbf{u}_i^m(\mathbf{q}), \mathbf{U}_i^m(\mathbf{s},\mathbf{s}), \mathbf{U}_i^m(\mathbf{q},\mathbf{q}), \mathbf{U}_i^m(\mathbf{s},\mathbf{q}));$
end if
```

and DBA run on a 2.6 GHz Pentium Core Quad provided with 4 Gb of RAM.

5.1 Running Time Evaluation on Synthetic Experiments

Experiment	n_P	n_z	t_{SBA}	t_{DBA}	%
Zig-Zag	79516	1083424	2.83s	1.77s	37.46
Outward	72134	570900	3.47s	2.41s	30.62
Random	69764	798798	6.83s	4.02s	41.16

Table D.1: Synthetic experiments setup

We have designed three synthetic experiments. Cameras are placed in a regular 3D environment where points are uniformly distributed. The depth at which a point is detected is limited to lie between 10 and 40 m. This is a valid supposition since in real applications objects have different levels of description at

Algorithm 3 distributeEvidence($\mathcal{T} \rightarrow \mathcal{N}_i, \lambda$)

```
if $\mathcal{T} \rightarrow \mathcal{N}_i = \mathcal{T} \rightarrow \mathcal{N}_{root}$ then
    $\mathbf{U}_i^m = \mathbf{U}_i^m + \lambda \mathbf{I}$;
    $\delta_{\mathbf{x}_c}^i = \mathbf{U}_i^m \backslash \mathbf{u}_i^m$;
else
    $k =$ parent($\mathcal{T} \rightarrow \mathcal{N}_i$)
    {$\mathbf{s}$: elements in $\mathcal{S}_{ik}$ separator}
    {$\mathbf{q}$: camera elements not in $\mathcal{S}_{ik}$}

    $\delta_{\mathbf{x}_c}^i(\mathbf{s}) = \delta_{\mathbf{x}_c}^k(\mathbf{s})$;
    $\delta_{\mathbf{x}_c}^i(\mathbf{q}) =$ backSubs($\mathbf{u}_i^m(\mathbf{q}), \delta_{\mathbf{x}_c}^i(\mathbf{s}), \mathbf{U}_i^m(\mathbf{q},\mathbf{q}), \mathbf{U}_i^m(\mathbf{s},\mathbf{q})$);
    $\delta_{\mathbf{x}_c}^i = \begin{bmatrix} \delta_{\mathbf{x}_c}^i(\mathbf{s}) \\ \delta_{\mathbf{x}_c}^i(\mathbf{q}) \end{bmatrix}$
end if
$\delta_{\mathbf{x}_p}^i =$ backSubs($\mathbf{v}_i, \delta_{\mathbf{x}_c}^i, \mathbf{V}_i, \mathbf{W}_i$)
$\delta_{\mathbf{x}}^i = \begin{bmatrix} \delta_{\mathbf{x}_c}^i \\ \delta_{\mathbf{x}_p}^i \end{bmatrix}$

for $j =$ children($\mathcal{T} \rightarrow \mathcal{N}_i$) do
    distributeEvidence($\mathcal{T} \rightarrow \mathcal{N}_j, \lambda$)
end for
```

different distances. Also, to avoid geometry configuration problems, we only use those points that are observed from at least three separate camera positions. The first experiment consists of a set of cameras located on a zig-zag path. The second experiment is designed such that cameras follow an inward-outward path. For the third experiment, cameras are located randomly. For this simulation we fixed the maximum number of points in a bounded area while in the rest of the experiments the point cloud grows as long as we add more cameras as occurs in exploration trajectories.

Figure D.4, shows the running time per iteration for DBA and SBA. In order to analyze the scalability, both algorithms are run several times for different number of cameras from 300 to 1500. Table D.1 summarizes the main cost per iteration at the maximum number of cameras along with the percentage of time reduction. The number of points and processed observations are also listed. For both algorithms the computation time per iteration considers the time required to carry out Schur complement and back substitution operations, as well as the time required to solve the complete camera reduced system. For DBA we ad-

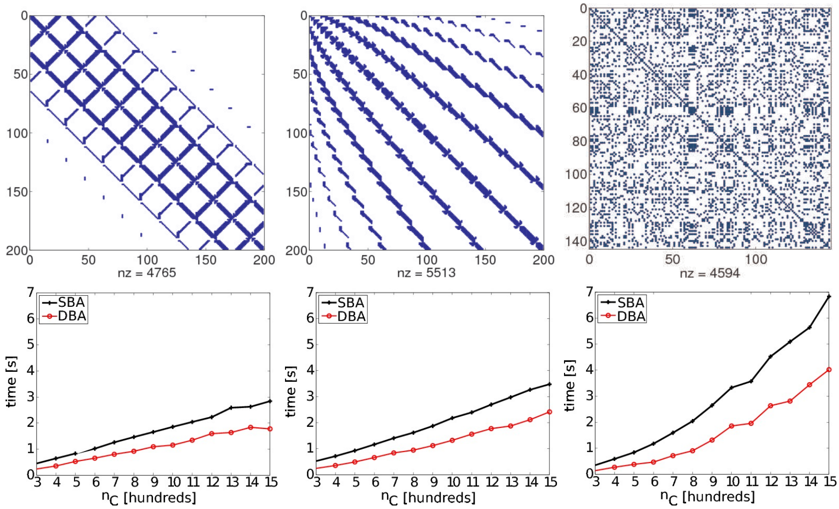


Figure D.4: Synthetic experiments: zig-zag (left column), in-outward (middle column) and random (right column). Example of the *secondary structure pattern* (top). Running time per iteration (bottom). Results obtained by increasing n_C from 3 to 15 hundred positions.

ditionally consider the time of adding all the camera marginals sent by children clusters when collecting information up the junction tree (lines 9-10 of Algorithm 2). The 'zig-zag' (left) and 'in-outward' (middle) experiments show an almost linear growth with n_c since adding new cameras only changes the dimension of the reduced camera matrix but not the pattern of the secondary structure. In these cases, DBA achieves an average reduction rate of 36.8% and 40.65% respectively. The 'random' experiment (right) yields a more connected camera graph such that the secondary structure gets denser leading to a quadratic time growth. This occurs because we increase n_C in a fixed size area. On average, DBA takes 49.8% less time than required by SBA. We have verified in all cases that the calculations of residuals, Jacobians and forming the Hessian submatrices take the same time for both SBA and DBA algorithms.

5.2 Evaluation on Real Data

We have carried out a comparison of DBA and the standard SBA for four real image collections. Three of the datasets are part of the Microsoft Photo Tourism

Experiment	n_C	n_P	n_z	t_{SBA}	t_{DBA}	%	n_V	JT_{depth}	n_{PB}
Ladybug	783	83581	372940	27.57s	11.73s	57.45	192	56	16
Public square	519	32068	286162	4.45s	2.38s	46.52	56	22	5
Trafalgar	256	65127	225688	1.97s	1.49s	24.37	86	23	19
Venice	87	110844	554826	5.59s	5.04s	9.86	4	4	1

Table D.2: Real experiments setup. Columns: number of cameras n_C, number of points n_P, number of observations n_z, number of junction tree clusters n_V, junction tree depth JT_{depth}, number of parallelizable branches n_{PB}.

project [13]. Table D.2 summarizes the setup configuration for each collection. The first dataset corresponds to a tree-lined avenue traversed by a car with a mounted Ladybug sensor, a camera system providing six images at each acquisition instant. The second dataset is an ordered sequence of images of a public square. The third collection is a set of tourist images taken at Trafalgar Square. Finally, the fourth dataset corresponds to tourist pictures of a portion of Saint Mark Square in Venice. For all datasets, a modified version of the Bundler software [13] with partial optimization has been used to obtain the initial seed. The experiments present different fill-in patterns for the reduced camera matrix as shown in Figure D.5, left column.

For the ‘ladybug’, ‘public square’ and ‘Trafalgar’ datasets, we can identify separate regions in the secondary structure matrix mainly due to the distributed location of the cameras in the experiments. In these cases, there is an important computational time reduction for DBA, especially in the first two experiments where DBA requires approximately 50% less time than SBA. The ‘Venice’ experiment is an example of little gain using DBA since all cameras are looking at a common part of the scene. For object centered applications this indicates that DBA and SBA achieve almost the same performance. Observe that the junction tree built for the ‘Venice’ experiment gets automatically adapted to the dense relation between cameras since only 4 consecutive clusters are generated, see Figure D.5 (bottom row).

6 Parallelization

The most costly operations of a BA method are the calculations needed to build the camera and point Jacobians, the construction of the information vectors and matrices and the solution of the linear system of equations. An important feature of the proposed algorithm is its potential capability to parallelize or distribute over several machines most of these operations.

Since all nodes of the tree have an independent set of observations the **calculateJacobiansAndResiduals** function can be run in parallel for all nodes simultaneously. For the collect and distribute functions we can take advantage of the junction tree structure. The key insight is that calculations performed in different branches of the tree are independent and therefore can be parallelized. For example, line 7 in Algorithm 2 can be run simultaneously for all children of the current node. This can be easily understood using Figure D.3 left. Observe that the matrices sent to the root $\mathcal{N}_2$ by clusters $\mathcal{N}_1$ and $\mathcal{N}_3$ can be calculated simultaneously. Similarly, line 17 in Algorithm 3 can also be called in parallel. Table D.2 shows for each of the real experiments the number of potentially parallelizable branches n_{PB}. Observe that for the first three experiments a Core Quad processor could take advantage of all four cores since $n_{PB} > 4$. However, it would be interesting to reduce the depth of the junction tree JT_{depth} in order to make the branches of the tree more balanced and increase n_{PB}.

7 Conclusions

In this paper we have analyzed the decomposable structure of bundle adjustment, which depends on the sparseness of the reduced camera matrix. To take advantage of this property, we have proposed a new decomposable bundle adjustment algorithm that automatically splits the original normal equations into smaller systems using a junction tree. A good capability of DBA is that the tree structure gets adapted to the secondary structure of the problem revealing its natural sparsity. To solve the decomposed system a two-way passing algorithm based on local Gaussian elimination and back substitution operations is implemented. An interesting side effect of the message passing protocol is that each clique tree obtains its corresponding marginal information matrix that can be used to check or carry out data association decisions. The performance of the proposed algorithm has been empirically evaluated using simulated and real experiments. In these experiments the DBA algorithm obtains good results compared to a standard SBA implementation for a wide range of camera configurations.

A very appealing property of DBA is that matrices present in different branches of the tree can be treated independently. This makes DBA suitable for parallel and distributed implementations. For future research we are interested in developing algorithms for multiple core systems using this technique as well as analyzing good ordering strategies to build junction trees that increase the number of parallel branches. In addition, we are interested in developing an incremental version of this algorithm following the nice ideas presented in [8].

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, New York, 1999.

[3] Tim A. Davis. *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms. SIAM, 2006.

[4] F. Dellaert, A. Kip, and P. Krauthausen. A Multifrontal QR Factorization Approach to Distributed Inference applied to Multi-robot Localization and Mapping. In *20th National Conference on Artificial Intelligence (AAAI)*, volume 3, 2005.

[5] Chris Engels, Henrik Stewénius, and David Nistér. Bundle adjustment rules. In *In Photogrammetric Computer Vision*, 2006.

[6] Nicolas Guilbert, Adrien Bartoli, and Anders Heyden. Affine Approximation for Direct Batch Recovery of Euclidean Structure and Motion from Sparse Data. *International Journal of Computer Vision*, 69:317–333, 2006.

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, U. K., 2000.

[8] M. Kaess, V. Ila, R. Roberts, and F. Dellaert. The Bayes Tree: An Algorithmic Foundation for Probabilistic Robot Mapping. In *Intl. Workshop on the Algorithmic Foundations of Robotics*, 2010.

[9] Kurt Konolige. Sparse Sparse Bundle Adjustment. In *British Machine Vision Conference*, Aberystwyth, Wales, 08/2010 2010.

[10] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[11] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[12] M. A. Paskin and G. D. Lawrence. Junction tree algorithms for solving sparse linear systems. Technical Report UCB/CSD-03-1271, Computer Science Division (EECS), University of California, Berkeley, California 94720, September 2003.

[13] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, November 2008.

[14] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

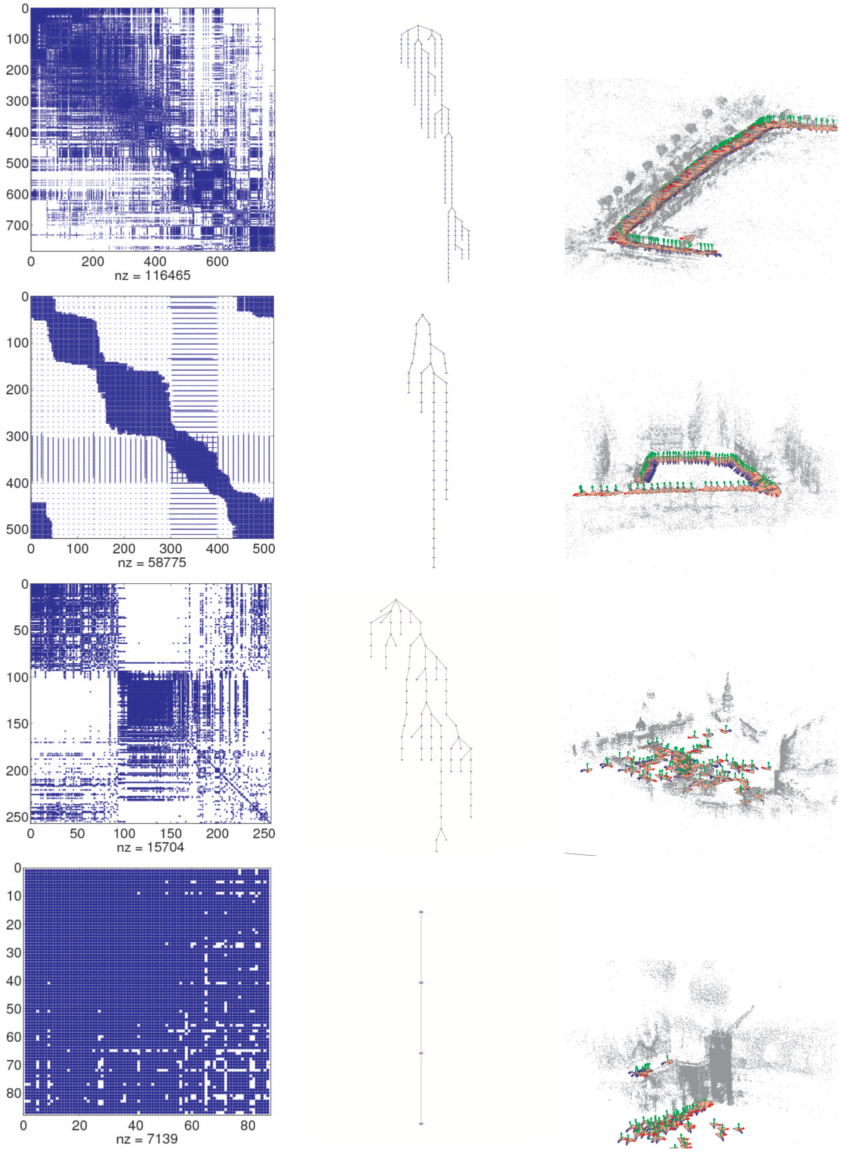
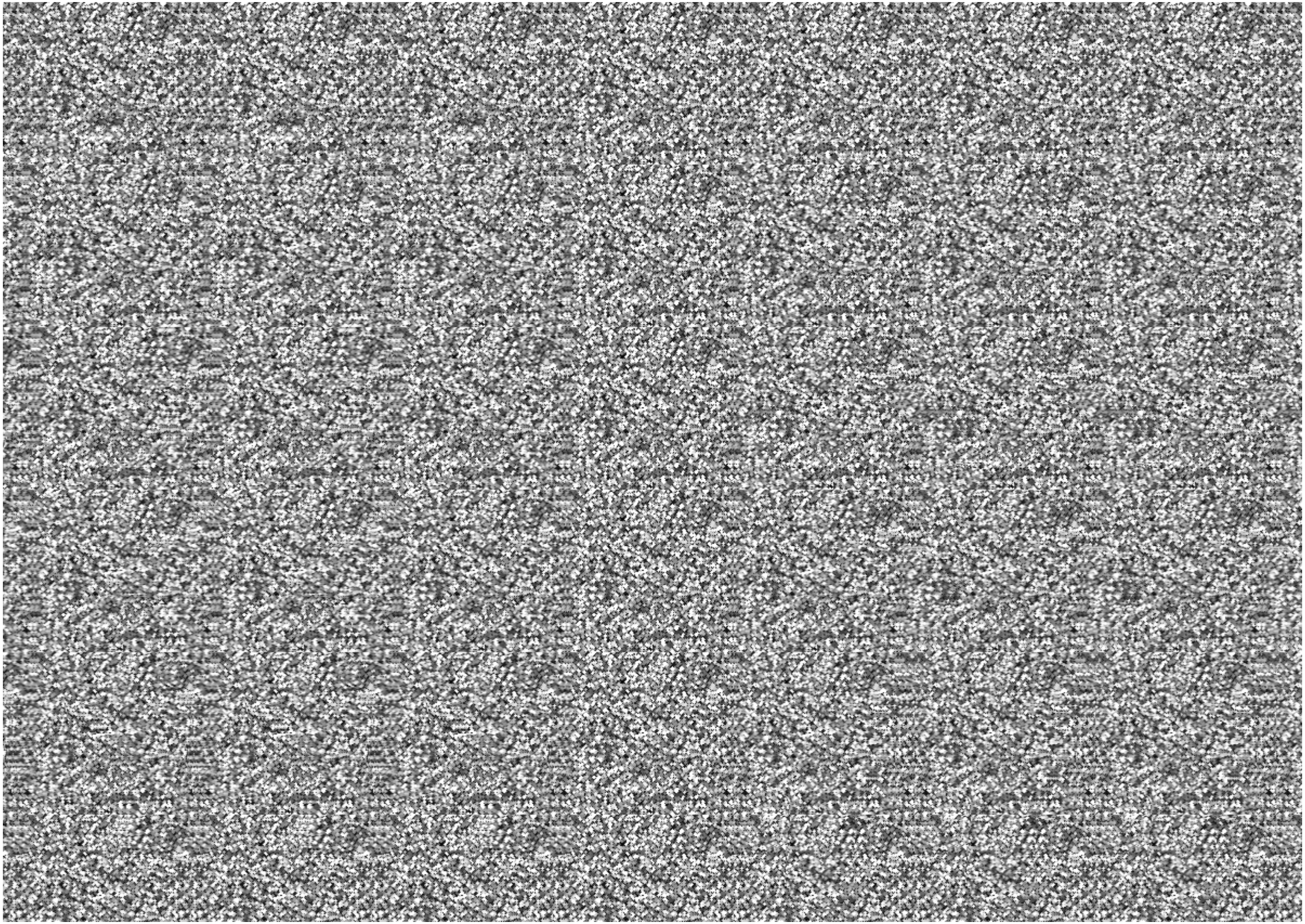


Figure D.5: Real experiments: 'ladybug', 'public square', 'Trafalgar' and 'Venice'. The secondary structure matrix (left column). Junction tree built (middle column). Obtained BA reconstruction (right column).



Random dot stereogram generated from a 3D reconstruction of a statue obtained using structure-and-motion techniques.