Generalized Subgraph Preconditioners for Large-Scale Bundle Adjustment

Yong-Dian Jian, Doru C. Balcan and Frank Dellaert
College of Computing, Georgia Institute of Technology
{ydjian,dbalcan,dellaert}@cc.gatech.edu

Abstract

We present a generalized subgraph preconditioning (GSP) technique to solve large-scale bundle adjustment problems efficiently. In contrast with previous work which uses either direct or iterative methods as the linear solver, GSP combines their advantages and is significantly faster on large datasets. Similar to [11], the main idea is to identify a sub-problem (subgraph) that can be solved efficiently by sparse factorization methods and use it to build a preconditioner for the conjugate gradient method. The difference is that GSP is more general and leads to much more effective preconditioners. We design a greedy algorithm to build subgraphs which have bounded maximum clique size in the factorization phase, and also result in smaller condition numbers than standard preconditioning techniques. When applying the proposed method to the "bal" datasets [1], GSP displays promising performance.

1. Introduction

Large-scale visual modeling with Structure from Motion (SfM) algorithms is an important problem. Recently, systems capable of handling millions of images have been built to realize this task [2, 13, 19], enabling automated 3D model generation from unstructured internet photo collections.

Bundle adjustment is used to find the optimal estimates of camera poses and 3-D points [22]. Mathematically speaking, it refers to the problem of minimizing the total reprojection error of the 3-D points in the images. The classical strategy to solve this problem is to apply a damped Newton's method (e.g., Levenberg-Marquardt) and solve the reduced camera system by Cholesky factorization. However, this strategy does not scale well because the memory requirement of factorizing these large matrices grows quadratically with the number of variables in the worst case.

Several recent works suggest using iterative methods such as the conjugate gradient (CG) method to solve the linear systems arising in bundle adjustment, as its memory requirement grows only linearly with the number of variables. The convergence speed of the CG method depends on how *well conditioned* the original problem is. Hence having a good preconditioner is crucial to make CG converge faster, yet most of the previous approaches apply only standard preconditioning techniques, neglecting to exploit SfM-specific constraints [1, 2, 7, 8, 14].

In robotics, Dellaert et al. [11] recently proposed Subgraph-Preconditioned Conjugate Gradients (SPCG), which aims to combine the advantages of direct and iterative methods to solve 2-D Simultaneous Localization and Mapping (SLAM) problems. The main idea is to pick a subset of measurements that can be solved efficiently by direct methods, and use it to build a preconditioner for the CG method. They show that SPCG is superior to using either direct or iterative methods alone. However, for the bundle adjustment problem, whose graph structure is bipartite and highly unbalanced, SPCG may over-estimate the uncertainty of the variables and hence lead to unsatisfactory preconditioners.

In this paper, we adapt the idea of SPCG to solve large-scale bundle adjustment problems efficiently, and propose the Generalized Subgraph Preconditioning (GSP) technique to avoid the over-estimation of uncertainty. In contrast with SPCG, which works with the Jacobian of measurements and treats each of them as a factor, GSP operates on the *Hessian* of measurements, and decompose each of them into three factors. The resulting graph representation is more general and provides a better tool to design subgraph preconditioners. From this perspective, the problem of designing a good subgraph preconditioner is reduced to picking a subset of the factors that (1) can be solved efficiently by direct methods, and also (2) make the linear systems well-conditioned.

Compared to conventional matrix preconditioning machinery, GSP not only provides an expressive language to design subgraph preconditioners, but also explains the standard Jacobi preconditioner naturally. We will also show how GSP leads to reducing the condition numbers.

An important open question in [11] is how to pick a

This work is supported in part by Intel and National Science Foundation, Robust Intelligence program award 0713134.

good subgraph. To this end, we introduce the ideas developed in the field of combinatorial preconditioners to help build good subgraph preconditioners [6]. The insight is that a good subgraph should not only be sparse but also have small structural distortion (*stretch*) with respect to the original graph. Yet finding the optimal subgraph that satisfies the above criteria is intractable for large graphs. Instead we propose a greedy algorithm to construct a family of subgraphs by incrementally adding edges to reduce stretch without inducing large cliques in the factorization phase.

This paper has three major contributions: we (1) adapt the ideas of SPCG to the bundle adjustment problem, (2) propose GSP which generalizes SPCG and leads to more effective subgraph preconditioners, and (3) develop a greedy algorithm based on the ideas in combinatorial preconditioners to construct a family of subgraph preconditioners. We use the proposed method to solve large-scale datasets and have promising results.

2. Bundle Adjustment

Below we review the bundle adjustment problem, whose goal is to jointly estimate the optimal camera parameters and 3-D structure by minimizing the total reprojection error. We define x_i $(i \in 1 \cdots M)$ as the camera parameters, l_j $(j \in 1 \cdots N)$ as the 3-D points, and z_k $(k \in 1 \cdots K)$ as the measurement of the 3-D point l_{kj} in camera x_{ki}. We define $h_k(x_{ki}, l_{kj})$ as the measurement function of a 3-D point in a camera. The cost function can be written as

$$\sum_{k=1}^{K} \|h_k(x_{ki}, l_{kj}) - z_k\|^2. \tag{1}$$

The standard approach is to linearize Eq. (1) around the current estimate to derive a linear system

$$\mathbf{A}\delta\mathbf{x} = \mathbf{b}, \tag{2}$$

where $\mathbf{A}$ is a rectangular matrix containing the Jacobian of all measurements with respect to the variables, and $\mathbf{b}$ is a vector of measurement error. An alternative is to form and solve the normal equation

$$(\mathbf{A}^T\mathbf{A})\delta\mathbf{x} = \mathbf{A}^T\mathbf{b}, \tag{3}$$

where $\mathbf{A}^T\mathbf{A}$ is an approximation to the Hessian of Eq. (1). In Levenberg-Marquardt, we add a damping term:

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{D})\delta\mathbf{x} = \mathbf{A}^T\mathbf{b}, \tag{4}$$

where λ is a non-negative scalar, and $\mathbf{D}$ can be an identity matrix or the diagonal of $\mathbf{A}^T\mathbf{A}$. The linear system corresponding to the normal equation (4) is

$$\begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda\mathbf{D}} \end{bmatrix} \delta\mathbf{x} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \tag{5}$$

2.1. Direct Methods

Using direct methods to solve Eq. (4) has been well-studied in literature [15, 22]. A common practice is to eliminate all 3-D points first, and use Cholesky factorization to solve the reduced camera system. Yet as shown in [2, 8, 14], this strategy works well on small problems, but does not scale satisfactorily because (1) the cost of forming and storing the reduced camera systems is prohibitive for large problems, and (2) building the reduced camera system could destroy the sparse problem structure and hence make it more expensive to solve. Therefore direct methods cannot be applied to solve large-scale bundle adjustment without using hierarchical or incremental techniques [17, 20]. A better alternative is to use iterative methods.

2.2. Iterative Methods

Using iterative methods such as the conjugate gradient (CG) method to solve large-scale bundle adjustment is beneficial because it involves only matrix-vector multiplication operations and hence requires less memory than direct methods [18]. Yet the number of required CG iterations depends on how well conditioned is the original problem.

Several preconditioners have been applied to make bundle adjustment well-conditioned. Agarwal et al. [1] examined the performance of several standard preconditioners and implementation strategies on large-scale datasets. Byröd and Åström proposed to use a multi-scale and the block-Jacobi preconditioners [7, 8]. Jeong et al. [14] suggested using the band-diagonal of the reduced camera system as a preconditioner. Yet these methods are very generic; we show that by exploiting the problem structure of bundle adjustment we can obtain better preconditioners.

3. Subgraph-Preconditioned Conjugate Gradients

To proceed, we take a graphical perspective on the bundle adjustment problem. In particular, the sparse Jacobian matrix $\mathbf{A}$ in Eq. (2) can be regarded as representing a *factor graph*, where the cameras and the 3-D points are the vertices, and each measurement (block-row) corresponds to a *factor* connecting the corresponding camera and point. Fig. 1 gives an example to illustrate the graph and matrix representations of the bundle adjustment problem.

Recently Dellaert et al. [11] proposed the Subgraph-Preconditioned Conjugate Gradient (SPCG) method, which aims to combine the advantages of direct and iterative methods to solve 2-D Simultaneous Localization and Mapping (SLAM) problems. The main idea is to identify a subproblem (subgraph) that can be solved efficiently by direct methods (e.g., a subgraph with small tree-width) and use it to build a preconditioner for the conjugate gradient method. Their results show that this technique is a better alternative to using either direct or iterative methods alone.

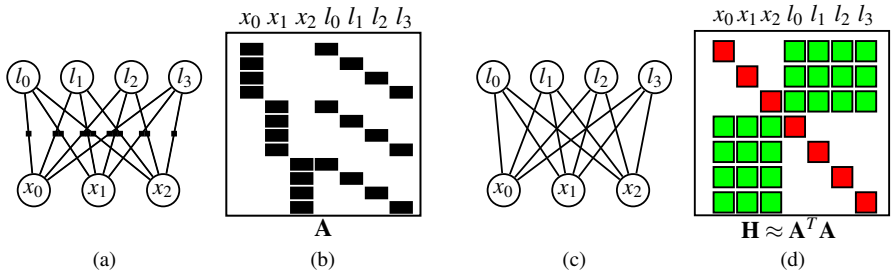

Figure 1: A toy bundle adjustment problem with three camera poses and four landmarks. All of the landmarks are observed by all of the cameras. Circles are variables and squares are factors. (a) The factor graph representation of the Jacobian $\mathbf{A}$. Each factor indicates a function between two variables. (b) The symbolic matrix representation of $\mathbf{A}$. Each row denotes one factor, and each column indicates one variable. (c) The Gaussian Markov Random Field representation of the approximated Hessian $\mathbf{H} \approx \mathbf{A}^T\mathbf{A}$. (d) The symbolic matrix representation of $\mathbf{H}$. Both rows and columns indicate variables. A diagonal block indicates the certainty of a variable given the other variables are known. An off-diagonal block indicates whether two variables are correlated given that the other variables are known, and each block has a corresponding factor in (a) or edge in (c).

We summarize SPCG based on the above graphical representation. Suppose we want to solve a linear system (graph) as in Eq. (5). We pick a subset of the rows (factors), and denote it as $(\mathbf{A}_1, \mathbf{b}_1)$, and denote the remaining rows as $(\mathbf{A}_2, \mathbf{b}_2)$. We can re-arrange Eq. (5) as

$$\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} \delta\mathbf{x} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}. \tag{6}$$

When applying QR factorization to $\mathbf{A}_1$, we have $\mathbf{A}_1 = \mathbf{Q}_1\mathbf{R}_1$. By left-multiplying the upper part with $\mathbf{Q}_1^T$, we get

$$\begin{bmatrix} \mathbf{R}_1 \\ \mathbf{A}_2 \end{bmatrix} \delta\mathbf{x} = \begin{bmatrix} \mathbf{Q}_1^T\mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}. \tag{7}$$

Suppose $\mathbf{c}_1 = \mathbf{Q}_1^T\mathbf{b}_1$ and $\bar{\mathbf{x}} = \mathbf{R}_1^{-1}\mathbf{c}_1$ is the optimal solution by considering only the upper part of (6). Then by re-parameterizing $\mathbf{y} = \mathbf{R}_1(\mathbf{x} - \bar{\mathbf{x}})$, we have

$$\begin{bmatrix} \mathbf{I} \\ \mathbf{A}_2\mathbf{R}_1^{-1} \end{bmatrix} \delta\mathbf{y} = \begin{bmatrix} \mathbf{0} \\ \mathbf{c}_2 \end{bmatrix}, \tag{8}$$

where $\mathbf{c}_2 = \mathbf{b}_2 - \mathbf{A}_2\mathbf{R}_1^{-1}\mathbf{c}_1$. Eq. (8) couples the solution of the subgraph part ($\mathbf{R}_1^{-1}$) to precondition the remaining part. The intuition behind the re-parameterization is that we penalize the deviation of $\mathbf{y}$ from the subgraph solution $\bar{\mathbf{x}}$. Finally Eq. (8) is solved by using the least-squares variant of the conjugate gradient method [4].

Fig. 2 illustrates the SPCG technique. Suppose we pick a spanning tree of the original graph as in Fig. 2a, we can solve it efficiently with sparse direct methods, and use its solution to precondition the re-parameterized system.

Yet for bundle adjustment, whose graph structure is bipartite and unbalanced, SPCG over-estimates uncertainty of the variables and produces unsatisfactory preconditioners.

4. Generalized Subgraph Preconditioners

We present a Generalized Subgraph Preconditioning (GSP) technique, which generalize the idea of SPCG and is more suitable for large-scale bundle adjustment. In contrast with SPCG, which works with the Jacobian of measurements and treats each of them as a factor, GSP operates on the Hessian, and decomposes each of the measurements into three factors in the Gaussian Markov Random Field (GMRF). The resulting graph representation is more general and provides a better tool to design subgraph preconditioners. Compared to conventional matrix preconditioning machinery, GSP not only provides an expressive language to design subgraph preconditioners, but also explains the standard Jacobi preconditioner naturally.

4.1. A Factor Graph Representation of the Hessian

To gain insight into the performance properties of both Jacobi and SPCG preconditioners, we investigate the structure of the approximated *Hessian* matrix $\mathbf{H} \approx \mathbf{A}^T\mathbf{A}$ appearing in the normal equation (3). This sparse symmetric matrix can also be represented as a graph, specifically a Gaussian Markov Random Field (GMRF) [12, 16]. Figs. 1c and 2c illustrate the GMRF representation of the Hessian.

Yet the common undirected graph representation of the Hessian is not expressive enough for designing good subgraph preconditioners in SfM, prompting us to resort to a more fine-grained *factor graph* representation of the Hessian. The main difference is that we create one binary and two unary factors out of each measurement, and keep all of them in the factor graph. The number of unary factors attached to a variable is equal to the number of associated measurements, with one binary factor per measurement.

As an example, consider the measurement between x_0 and l_0 in Fig. 1a and assume A_{x_0} and A_{l_0} are the corresponding block entries in the first row of the Jacobian. Then three factors corresponding to $A_{x_0}^T A_{x_0}$, $A_{l_0}^T A_{l_0}$ and $A_{x_0}^T A_{l_0}$ are added to the graph. Notice that the first two are unary factors of x_0 and l_0, and the third is a binary factor between them. They encode the information contributed by this mea-

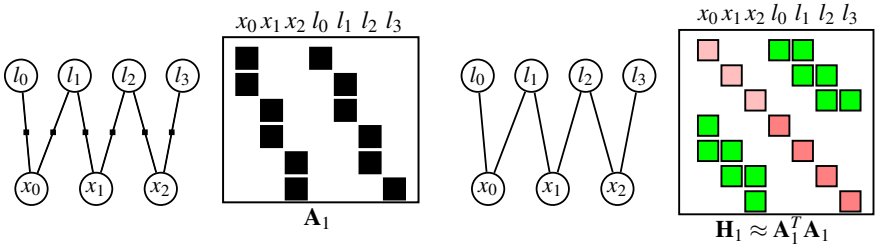

Figure 2: An illustration of the SPCG technique. (a) (b) The factor graph and symbolic Jacobian matrix that correspond to a subset of the measurements (sub-problem). (c) (d) The Gaussian Markov Random Field and the approximated Hessian matrix of the sub-problem. The corresponding non-zero off-diagonal blocks are identical to those in Fig. 1d, but the diagonal entries are smaller than those in Fig. 1d. It leads to over-estimating the uncertainty of the variables, especially for the camera variables. This is problematic for large-scale bundle adjustment where the graph is bipartite and highly unbalanced.

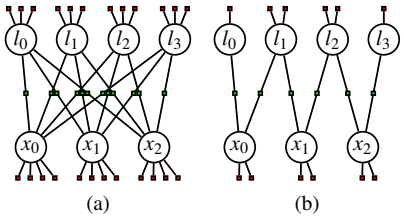

Figure 3: The factor graph representations of the Hessian matrices in Figs. 1 and 2, respectively.

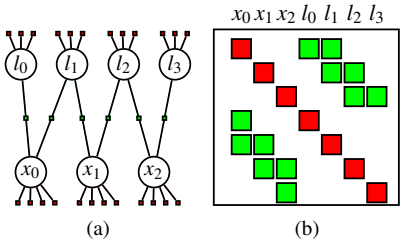

Figure 4: Subgraph that GSP produces but SPCG cannot.

surement to the conditional Gaussian densities. Repeating this process for all measurements, we build the factor graph representation of the Hessian illustrated in Fig. 3.

From this perspective, the problem of designing a good subgraph preconditioner is reduced to picking a subset of factors from the graph that (1) can be solved efficiently by direct methods, and also (2) make the linear systems well-conditioned. Once a subgraph is selected, we can use sparse Cholesky factorization to build the preconditioner (i.e., $\mathbf{H_1} = \mathbf{R_1^T R_1}$). The detail of how to to pick the subgraph will be discussed in Sec. 5.

The difference between GSP and SPCG is critical for large-scale bundle adjustment, whose graph structure is bipartite and highly unbalanced. By using the factor graph representation of the Hessian, GSP is more expressive than SPCG because we can always build a GMRF from a subset of measurements, but not vice versa. For instance, suppose we want to pick a subset of factors in GMRF as in Fig. 4a out of the GMRF in Fig. 3a. One can immediately see that no subset of measurements in Fig. 1a corresponds to this GMRF. Hence the GSP is indeed a generalization of SPCG.

The amount of information that SPCG brings in for each variable corresponds to the associated measurements in the subgraph. In bundle adjustment, if SPCG picks a spanning tree as the subgraph, then it can only collect at most two out of potentially thousands of unary factors for the camera vertices. This results in over-estimating the uncertainty of the variables and hence leads to unsatisfactory preconditioners. This idea is illustrated in Fig. 2d.

Adding more measurements to the subgraph might help, but it also makes the resulting subgraph harder to solve by sparse direct methods. In contrast, GSP provides the flexibility to keep part or all of the unary factors (information) for each variable, and hence overcomes this problem.

4.2. The Jacobi Preconditioner

The Jacobi method is a generic preconditioning technique. Here we show it has a simple explanation within the GSP framework. The Jacobi preconditioner works by taking only the diagonal entries of the Hessian matrix, and discarding all off-diagonal entries. A simple generalization is the block-Jacobi preconditioner which treats each camera or 3-D point as an entity, and it corresponds to picking the block diagonal of the Hessian matrix. They can be solved efficiently because all blocks are independent. In GSP machinery, the block-Jacobi preconditioner corresponds to picking all unary factors and discarding all binary factors of the Hessian. The idea is illustrated in Fig. 5. Note that hereafter when we refer to the Jacobi preconditioner, we actually mean the block-Jacobi preconditioner.

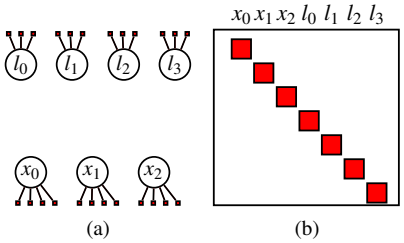

Figure 5: Block-Jacobi preconditioner of the toy problem.

4.3. The Positive Semidefiniteness of GSP

Any GSP preconditioner matrix is positive semidefinite (psd). Discarding off-diagonal block pairs $A_{x_0}^T A_{l_0}$, $A_{l_0}^T A_{x_0}$ in the Hessian while leaving the block-diagonal unchanged corresponds to replacing a binary factor by two unary factors in the Jacobian factor graph. The replaced binary factor corresponds to $\mathbf{A}$'s block-row with nonzero blocks A_{x_0} and A_{l_0}, while each new unary factor contains exactly one of these blocks. The inner product of the new factor matrix with itself is psd, which guarantees the validity of GSP preconditioners. Note that discarding symmetrical off-diagonal entries of an *arbitrary* symmetric psd matrix may not produce a psd matrix. In the scalar case, Boman et al. [5] proved that matrices with this property must admit a factorization $\mathbf{A}^T\mathbf{A}$, with $\mathbf{A}$ having a factor width ≤ 2.

5. Subgraph Construction Algorithm

5.1. Matrix Preconditioners

Conventional matrix preconditioning techniques focus more on the efficiency of solving the preconditioners rather than on directly minimizing the condition number of the preconditioned system [18]. For example, the Jacobi preconditioner offers good computational efficiency by discarding the conditional correlation between variables. The incomplete Cholesky preconditioner controls the computational cost by limiting the amount of fill-in and discarding negligible entries during the factorization process. The Symmetric Successive Over-relaxation preconditioner splits the matrix into lower- and upper-triangular pieces to avoid any factorization. Although these techniques work to some extent in practice, deriving theoretical bounds on their condition numbers is not trivial, and their actual meaning is also hard to interpret graphically or probabilistically.

5.2. Combinatorial Preconditioners

Recently, combinatorial preconditioners have been studied to analyze and construct effective preconditioners for the conjugate gradient method. Promising results have been reported on solving linear systems with symmetric and diagonally dominant matrices [6, 21]. The main idea is to find ultra-sparsifiers such that the original graph and the approximating graph have similar conductance – a measure of how fast information travels between different parts of the graph. Insisting on sparse approximating graphs produces preconditioners that can be solved efficiently by direct methods, while maintaining the graph conductance effectively reduces the condition number of the preconditioned systems, therefore the number of CG iterations.

If the subgraph is restricted to be a spanning tree, Boman and Hendrickson [6] recognized that the condition number of the preconditioned system is upper bounded by the *stretch* of the original graph with respect to the spanning tree. More specifically, suppose $G = (V, E, w)$ is the graph of the original system where V, E and w denote the vertices, edges and the weights of the edges respectively. If T is a spanning tree of G, then for every edge $e = (u, v) \in E$, there is a unique path in T connecting u and v. The stretch of e with respect to T is defined as

$$\mathbf{st}(T, e) = \sum_{f \in P(T,e)} \frac{w(e)}{w(f)}, \quad \text{for } e \in E \tag{9}$$

where $P(T, e)$ denotes the edges on the unique path between u and v in T. The stretch of G with respect to T is defined as the sum of the stretches of all the edges in G:

$$\mathbf{st}(T, G) = \sum_{e \in E} \mathbf{st}(T, e). \tag{10}$$

Intuitively speaking, the higher the stretch of a tree, the more time it takes for information to percolate, negatively affecting convergence.

If we relax the restriction and consider a general subgraph, a common practice is to use a *low-stretch* spanning tree as a skeleton and augment it with additional edges to further reduce the stretch. However, when additional edges are added to the subgraph, not only may the subgraph take longer to build, but also the preconditioners will become more expensive to apply in the conjugate gradient method. Clearly, there is a trade-off between the quality of the preconditioner and the time required to build and apply it.

5.3. The Proposed Algorithm

Here we propose a greedy algorithm to find a family of subgraph preconditioners based on GSP and the ideas developed in the combinatorial preconditioners.

The bundle adjustment graph is a bipartite graph $G = (X, L, E)$, where X and L denote the vertices of cameras and 3-D points respectively on the two sides of G. Each edge in E denotes a measurement that connects the corresponding camera and point vertices. The goal of this algorithm is to find a subset E_S of E, such that (1) the resulting subgraph G_S has low stretch with respect to G, and (2) the maximum size of the induced cliques does not exceed a predefined parameter n. By the maximum size of the induced

cliques we actually mean the clique number in the factorization phase, which can indirectly affect the computational complexity. A straightforward strategy would be to use a low-stretch spanning tree in G as the subgraph, but this does not exploit the bipartite and unbalanced nature of G.

We make some notations to facilitate the explanation. We denote $X(l)$ as the set of cameras associated with a 3-D point l, and $E(l)$ as the corresponding set of edges (measurements). Note that by picking t edges from $E(l)$ into the subgraph, we will induce a clique of size t between the corresponding cameras after eliminating the 3-D point l in the factorization phase. Moreover, if the edges and the elimination ordering are not chosen appropriately, even larger cliques will appear in the factorization phase.

Here we propose a greedy algorithm to construct a family of subgraphs. We first build a camera graph G_X where the vertices consist of all cameras and the edge weight between two cameras is defined as the number of 3-D points that are observed by both of them. Then we find a low-stretch spanning tree T_X in G_X. This tree aims to preserve the structural information of G, and provide a reference to control the size of induced cliques when adding the edges.

Now we explain how to pick edges into the subgraph. Initially the set E_S is empty. For each point l, we sort $X(l)$ according to their average distance to the other cameras in $X(l)$ with respect to T_X. Then we pick the edges of $E(l)$ into the subgraph according to this ordering. An edge is added into E_S if it does not induce a camera clique of size greater than n. To this end, we also maintain an array (initially set to 0, whose length is the number of cameras) which holds the size of the maximum clique that a camera belongs to. The array is updated whenever an edge is added. Repeating this process for all 3-D points results in edge set E_S, which we use to construct the subgraph preconditioner.

6. Results

6.1. Configurations

Here we compare the sparse factorization method (DBA) and the conjugate gradient (CG) method with three preconditioners: (1) the block-Jacobi preconditioner (JACOBI), (2) the subgraph preconditioner (SPCG), and (3) the generalized subgraph preconditioner (GSP-n). The number attached to "GSP-n " indicates the maximum clique size allowed in the greedy algorithm.

We use the Levenberg-Marquardt method as the nonlinear solver. The stopping criteria are (1) the number of iterations exceeds 20, (2) the average reprojection error is less than 0.8, or (3) the relative decrease of total reprojection error is less than 10^{-2}.

For the linear solvers, DBA uses the *cholmod* package with an approximate mininum degree ordering [9]. For the solvers using the CG method, we solve Eq. (5) by using the least-squares variant of CG [4] without forming the normal equation (see Algo. 1). The stopping criteria for the CG method are (1) the number of iterations exceeds 2000, (2) the relative decrease of residual is less than 10^{-2}.

Algorithm 1:
Preconditioned Conjugate Gradient Least-Squares Method

Input: let x_0 be an initial, and ϵ be the tolerance
$r_0 = b - Ax_0$, $p_0 = s_0 = R^{-T}(A^T r_0)$, $\gamma_0 = \|s_0\|_2^2$
for k = 0 **to** maximum iterations **do**
  if $\gamma_k < \epsilon$ **then** break $t_k = R^{-1}p_k$
  $q_k = At_k$
  $\alpha_k = \gamma_k / \|q_k\|_2^2$
  $x_{k+1} = x_k + \alpha_k t_k$
  $r_{k+1} = r_k - \alpha_k q_k$
  $s_{k+1} = R^{-T}(A^T r_{k+1})$
  $\gamma_{k+1} = \|s_{k+1}\|_2^2$
  $\beta_k = \gamma_{k+1}/\gamma_k$
  $p_{k+1} = s_{k+1} + \beta_k p_k$
end

For JACOBI, we accumulate all unary factors for each variable (i.e., the diagonal blocks of $\mathbf{A}^T\mathbf{A}$) and solve them independently. For SPCG, we use the Sparse QR factorization package [10]. For GSP-n, we use the proposed greedy algorithm to pick the subgraph in the Hessian factor graph and factorize it to build the preconditioners by using *cholmod* with an ordering in which 3-D points are eliminated first and the cameras are eliminated according to the topological ordering of the camera low-stretch spanning tree. We use Alon et al.'s algorithm to find a low-stretch spanning tree in the camera graph [3]. Note that for SPCG and GSP-n, the topology of the subgraph is determined at the beginning, and never changed during the optimization.

Although GSP offers the flexibility to use various number of unary factors for each variable, we choose to use all of the unary factors for simplicity. In this case, GSP-0 and is mathematically equivalent to JACOBI. Also, GSP-∞ is like DBA if CG runs only one iteration.

We run the experiments on the *bal* datasets released by Agarwal et al. [1]. Since *bal* contains many datasets and some of them cannot fit into the memory of a regular PC, we select ten proper datasets from *bal* which have 100K to 500K points (see Table. 2). We run all of the experiments on a Core2 Duo PC with 8G RAM.

6.2. The Performance of GSP-n

We first investigate the performance of GSP-n for different values of n, and show the timing results in Fig. 6. We exclude the linearization time and focus on comparing the linear solvers. The results show that GSP-n converges faster than JACOBI by 10-30% in most cases.

We also observe that as n increases, the overall time decreases at first, but increases if n is set too high. To better

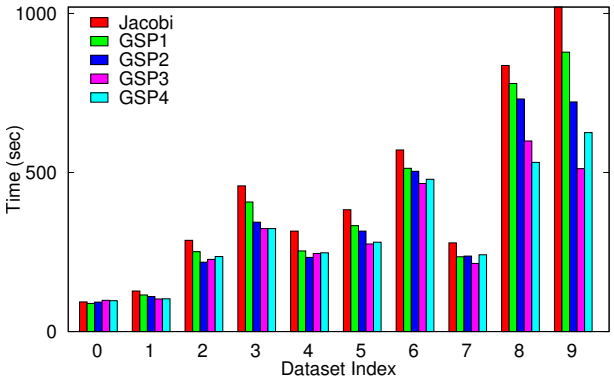

Figure 6: Timing results of JACOBI and GSP-n on *bal*.

Table 1: Timing results of GSP-n on the "F-05" dataset. We only show the components relevant to the linear solvers. The columns indicate (1) the maximum clique size in GSP-n, (2) the percentage of edges used in the subgraph, (3) the time of building the subgraph, (4) the time per CG iteration, and (5) the number of total CG iterations, and (6) the total time.

n	edges (%)	build (s)	time/iter (s)	#iters	total (s)
0	0.0	27.2	0.48	1438	732.6
1	19.8	33.4	0.53	1130	648.8
2	26.6	48.7	0.56	866	550.5
3	32.5	69.1	0.62	631	**473.7**
4	39.0	101.5	0.78	526	512.8

understand the behavior of GSP-n, we break down the timing results of one dataset and show the major components of the linear solvers in Table 1. We can see that as n increases, the subgraph becomes less sparse and hence harder to solve, but the time spent on building the subgraph preconditioner is not significant when n is still small. Here the more important parts are (1) the time to apply the preconditioner per CG iteration, and (2) the number of total CG iterations. The former increases because the preconditioner becomes more dense and hence more computation is involved in the back substitution. The latter decreases because the linear systems become better conditioned. We can see that their product dominate the total time. Clearly there is a trade-off between these two factors.

6.3. Timing Results

Here we compare the timing results of four linear solvers on the *bal* datasets. According to Sec. 6.2, we use $n = 3$ to build subgraphs for both SPCG and GSP-n. The timing results in Table 2 are sorted according to the DBA time, which reflects the intrinsic difficulty of the datasets. The results confirm that sparse factorization methods are efficient for small datasets, but preconditioned CG methods are better alternatives for large datasets. Comparing JACOBI and GSP, the results show that by adding extra edges (binary factors) to the subgraph preconditioners, GSP provide better preconditioners than JACOBI in most of the cases. Comparing SPCG and GSP, the results show that being able to add more unary factors to the graph is crucial to improve the convergence speed of the CG method.

Table 3: The Condition numbers of the SPCG, JACOBI, and GSP-3 preconditioners on three *bal* datasets.

set	Original	SPCG	JACOBI	GSP-3
D-15	5.58e+21	1.87e+06	5.94e+04	4.36e+03
V-02	6.54e+21	6.46e+09	6.35e+05	1.38e+05
F-01	3.68e+11	1.92e+08	7.54e+06	8.71e+05

6.4. The Condition Numbers

In addition to the running time, the quality of a preconditioner can be evaluated by the condition number of the preconditioned systems. Here we show the condition numbers of the SPCG, JACOBI and GSP-3 preconditioners on several medium *bal* datasets in Table 3. We can see that the original condition numbers (of Eq. 4) are huge, which indicate the slow convergence of using a plain CG solver. The SPCG precondtitioner works to some extent, but is not as good as JACOBI and GSP-3. The condition numbers of GSP-3 are 5-10 times smaller than JACOBI.

7. Conclusions and Future Work

While direct methods are efficient for small datasets and iterative methods are more appropriate if the memory requirement is of concern, a subgraph-based preconditioning method combines their advantages and provides a better alternative for solving large-scale bundle adjustment. One such method is SPCG, which to the best of our knowledge has not been applied to the bundle adjustment problem until now. Although for large datasets SPCG is significantly better than direct methods and the plain CG method, its behavior is sub-optimal: as the bundle adjustment graph is bipartite and unbalanced, SPCG overestimates the uncertainty of the variables. In contrast, GSP avoids this problem, and is more expressive and suitable for bundle adjustment. Well-known preconditioners like Jacobi fit naturally in the GSP context. To exploit the graphical structure of the problem, we develop an efficient algorithm rooted in combinatorial preconditioning, to construct a family of subgraph preconditioners. When applied to large datasets, GSP preconditioners display promising performance.

The first possible direction for future work is to develop a more expressive subgraph preconditioner language to explain and understand other matrix preconditioners such as

Table 2: Timing results (secs) of the four methods on ten *bal* datasets. The second column corresponds to the name and index in the original *bal*: "D" for "Dubrovnik", "L" for "Ladybug", "V" for "Venice" and "F" for "Final".

set	source	cameras	points	measurements	DBA	JACOBI	SPCG	GSP-3
0	V-01	89	110,973	562,976	**42**	84	401	89
1	F-01	394	100,368	534,408	**79**	113	256	96
2	V-02	245	198,739	1,091,386	**155**	245	415	196
3	D-15	356	226,730	1,255,268	**187**	397	804	285
4	V-03	427	310,384	1,699,145	313	273	695	**212**
5	L-30	1,723	156,502	678,718	578	312	718	**223**
6	V-04	744	543,562	3,058,863	886	506	913	**407**
7	F-03	961	187,103	1,692,975	1148	252	741	**191**
8	F-02	871	527,480	2,785,977	1939	776	1154	**564**
9	F-05	3,068	310,854	1,653,812	3504	894	2035	**473**

SSOR and Incomplete Cholesky factorization in terms of graphs. The second is to develop a better algorithm to construct the subgraph preconditioners, and provide theoretical guarantees for their performance.

References

[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *European Conference on Computer Vision*, 2010. 1, 2, 6

[2] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building rome in a day. In *IEEE 12th International Conference on Computer Vision*, pages 72–79, 2009. 1, 2

[3] N. Alon, R. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24, 1995. 6

[4] Björck, A. *Numerical Methods for Least Squares Problems*. SIAM Publications, 1996. 3, 6

[5] E. Boman, D. Chen, O. Parekh, and S. Toledo. On factor width and symmetric h-matrices. *Linear algebra and its applications*, 405:239–248, 2005. 5

[6] E. Boman and B. Hendrickson. Support theory for preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 25(3):694–717, 2003. 2, 5

[7] M. Byröd and K. Åström. Bundle adjustment using conjugate gradients with multiscale preconditioning. In *British Machine Vision Conference*, 2009. 1, 2

[8] M. Byröd and K. Åström. Conjugate gradient bundle adjustment. In *European Conf. on Computer Vision*, 2010. 1, 2

[9] Y. Chen, T. Davis, W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, 35(3):1–14, 2008. 6

[10] T. Davis. Algorithm 9xx, SuiteSparseQR: multifrontal multithreaded rank-revealing sparse QR factorization. 6

[11] F. Dellaert, J. Carlson, V. Ila, K. Ni, and C. E. Thorpe. Subgraph-preconditioned conjugate gradient for large scale slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010. 1, 2

[12] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, Dec 2006. 3

[13] J.-M. Frahm, P. F. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, and S. Lazebnik. Building Rome on a cloudless day. In *European Conference on Computer Vision*, 2010. 1

[14] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1474–1481, 2010. 1, 2

[15] M. Lourakis and A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):2, 2009. 2

[16] D. MacKay. *Information theory, inference, and learning algorithms*. Cambridge Univ Press, 2003. 3

[17] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3D reconstruction. In *IEEE 11th International Conference on Computer Vision*, 2007. 2

[18] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2003. 2, 5

[19] N. Snavely, S. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. 1

[20] N. Snavely, S. M. Seitz, and R. S. Szeliski. Skeletal graphs for efficient structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 2

[21] D. A. Spielman. Algorithms, graph theory, and linear equations. In *Int'l Congress of Mathematicians*, 2010. 5

[22] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. *Vision algorithms: theory and practice*, pages 298–372, 2000. 1, 2