A Consensus-Based Framework for Distributed Bundle Adjustment

Anders Eriksson[1], John Bastian[2], Tat-Jun Chin[2] and Mats Isaksson[3]
[1]School of Electrical Engineering and Computer Science, Queensland University of Technology
[2]School of Computer Science, The University of Adelaide
[3]Electrical and Computer Engineering Department, Colorado State University,

Abstract

In this paper we study large-scale optimization problems in multi-view geometry, in particular the Bundle Adjustment problem. In its conventional formulation, the complexity of existing solvers scale poorly with problem size, hence this component of the Structure-from-Motion pipeline can quickly become a bottle-neck. Here we present a novel formulation for solving bundle adjustment in a truly distributed manner using consensus based optimization methods. Our algorithm is presented with a concise derivation based on proximal splitting, along with a theoretical proof of convergence and brief discussions on complexity and implementation. Experiments on a number of real image datasets convincingly demonstrates the potential of the proposed method by outperforming the conventional bundle adjustment formulation by orders of magnitude.

1. Introduction

With the number of images and video available over the internet reaching several billions of terabytes and growing, the need for novel tools capable of handling such vast amount of data becomes apparent. In this paper we present a framework for solving a wide range of large-scale optimization problems in multi-view geometry in a distributed manner.

Our work focuses on Structure-from-Motion and in particular the Bundle Adjustment problem [23, 9], which is the process of iteratively refining the camera parameters as well as the 3D points of a scene reconstruction through non-linear optimization. Bundle adjustment constitutes a core component in most state-of-the-art multi-view geometry systems, and is typically invoked as a final refinement stage to approximate initial scene estimates as well as a means for removing drift in incremental reconstructions. The Levenberg-Marquardt algorithm has proven to be the most successful method for solving this formulation, as it is simple to implement, robust to initialization, and its framework makes it very amenable to taking advantage of the forms of sparsity that typically arise in multi-view geometry problems. Each step of this algorithm produces an estimate of the parameters that improves upon the previous and the resulting series of iterates can be shown to converge to a local minimum of the objective function at hand.

Despite its long history within the field of computer vision, bundle adjustment still receives a significant amount of research attention. In its conventional formulation, the complexity of bundle adjustment scales quite poorly with the size of the problem considered; as we transition from small and moderate reconstructions to truly large scale settings, bundle adjustment quickly becomes a computational bottle-neck. Hence, recent efforts in this topic have primarily been focussed towards improving the computational efficiency of bundle adjustment, in particular when applied to very large-scale reconstruction problems, see [16, 14, 24, 6, 15, 3], with a number of publicly available software packages as a result [16, 24, 1]. These previous approaches typically boost performance of bundle adjustment by raising the efficiency of specific computational steps taken within the algorithm itself. For example, a number of methods have been introduced in attempts at improving the efficiency of solving the linear systems that rise within the bundle adjustment algorithm. In [6] the authors advocated the use of conjugate gradients along with a novel preconditioning to replace the Cholesky factorization typically employed. In the paper [2] a method was proposed using a combination of sparse methods, preconditioned conjugate gradient and approximation methods with some very impressive results.

Other research directions include the work of [18]. Here the authors attempt to split the problem into a number of smaller subproblems, solve these individually and in parallel and then subsequently merge them into one large final reconstruction. However, because this method relies on decoupling the full scene into sub-maps that are as independent as possible, the degree of parallelism that can be achieved is limited to particular classes of scenes.

The paper [24] proposed a very efficient method for solving large scale 3D scene reconstruction problems by capi-

talizing on multicore GPUs and CPUs commonly available in today's computers. Some extremely impressive improvements in run-time was reported here. However, this work is more concerned with achieving speed-ups of the conventional bundle adjustment algorithm, exploiting multiple processors on a single computer. Less attention was paid to addressing out-of-core issues and inherent limitations caused by the algorithmic complexity of this conventional formulation.

This paper focuses on the problem of efficiently solving large-scale bundle adjustment problems in a distributed manner. The proposed method is ultimately intended as an external memory algorithm that fundamentally relies on message passing between resources, and is therefore capable of handling datasets exceeding the memory capacity of a single computer. Furthermore, no assumptions are made in regards to the sparsity or structure of the scene and no limitations on the degree of parallelism are imposed.

This present work is based on consensus optimization [5], a simple but powerful class of methods with the appealing property of being able to operate in a distributed fashion with minimal coordination between nodes. Consensus based algorithms are also extremely straightforward to implement while typically being able to avoid the overhead involved in other approaches, such as aggregating up spanning trees. They belong to a family of optimization algorithms known as proximal splitting methods; this very general class of methods have commonly been used in control and signal processing, applications [8] but has recently also received attention in the computer vision community [10, 11]. To our knowledge this is the first study into their use for distributed Structure-from-Motion recovery in multi-view geometry.

The paper is organized as follows. In the subsequent section, a brief review of multi-view geometry and the bundle adjustment problem is given, along with an introduction to proximal splitting methods and the Douglas-Rachford method. This is then followed by a section containing the derivation of the proposed algorithm, a theoretical convergence analysis and a discussion on implementational details. Finally, experimental results are given in section 5 followed by concluding remarks.

2. Projective Geometry and Bundle Adjustment

Multi-view geometry typically concerns the problem of estimating camera and structure parameters that minimize some aspect of the reprojection error of a number of measured image points. Here we restrict ourselves to problems on the following form.

Let $\pi(P_i, X_j) : \mathcal{Q} \times \mathbb{R}^3 \mapsto \mathbb{R}^2$ denote the projection, according to the pinhole camera model, of point $X_j \in \mathbb{R}^3$ in image i given its camera matrix $P_i \in \mathcal{Q} \subseteq \mathbb{R}^{3\times4}$ and $u_{ij} = \left[\, u_{ij}^x \; u_{ij}^y \,\right]^T$ the observed image location of the same point. Here the structure of the set $\mathcal{Q}$ depends on the setting and camera model used; for full projective reconstruction $\mathcal{Q}$ is simply $\mathbb{R}^{3\times4}$ but in the case of calibrated or uncalibrated euclidean reconstructions this set becomes slightly more involved.

When the error minimized is the total sum-of-squares reprojection error,

$$\min_{\substack{P_i\in\mathcal{Q}\\ X_j,\\ i=1,..,m\\ j=1,..,n}} \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}\left(u_{ij} - \pi(P_i, X_j)\right)^2 \tag{1}$$

given n measured image points in m images, we arrive at the well-known Bundle Adjustment problem. Here w_{ij} denotes the elements of the visibility matrix, a binary variable that equals 1 if point j is visible in image i and 0 otherwise.

With

$$P = \left[\begin{array}{ccc} P_1^T & \dots & P_m^T \end{array}\right]^T,\; X = \left[\begin{array}{ccc} X_1 & \dots & X_n \end{array}\right],$$
$$\Pi(P,X) = \left[\begin{array}{ccc} \pi(P_1,X_1) & \pi(P_1,X_2) & \dots \quad \pi(P_m,X_n) \end{array}\right],$$
$$U = \left[\begin{array}{ccc} u_{11} & u_{12} & \dots \quad u_{mn} \end{array}\right],$$

we can write (1) compactly, using the Hadamard (or elementwise) product denoted $\circ$, as

$$\min_{\substack{P\in\mathcal{Q}^m\\ X\in\mathbb{R}^{3\times mn}}} ||W \circ (U - \Pi(P,X))||_F^2. \tag{2}$$

Bundle adjustment solves (2) using iterative methods for non-linear least squares optimization. The Levenberg-Marquardt algorithm is arguably the most successful method for this formulation. The memory complexity for the conventional formulation of this algorithm is $\mathcal{O}(m^2)$, quadratic in the number of cameras, and a time complexity of $\mathcal{O}(m^3)$, cubic in the number of cameras. Hence, the computational costs for this algorithm rapidly becomes prohibitive as the size of the scene grows.

3. Proximal Splitting Methods

Consider a general convex optimization problem on the following form

$$\min_{x\in H} \underbrace{f_1(x) + ... + f_N(x)}_{=\Phi(x)}, \tag{3}$$

where $f_i,\ i = 1, ..., N$ are proper, convex and lower semi-continuous functions and H a Hilbert space. *Proximal splitting* methods are a general class of meta-algorithms used for solving (3). These methods proceed by splitting the objective functions and evaluating the summands f_i individually so as to yield an efficient and easily implementable algorithm.

The *proximity operator* $\text{prox}_f : H \to H$ of a proper, convex and lower semi-continuous function $f : H \to \mathbb{R}$, with $\rho > 0$ and H a Hilbert space, is defined as

$$\text{prox}_{f/\rho}(y) = \underset{x \in H}{\arg\min} \left(f(x) + \frac{\rho}{2}||x - y||^2 \right) \quad (4)$$

This notion was first introduced by [17] as a generalization of the concept of orthogonal projections onto convex sets. The proximity operator plays a central role in the development of the proximal splitting methods. These splitting schemes are first order optimization methods that are particularly aimed at minimizing a sum of functionals for which it is possible to efficiently compute its proximity operator.

For any $y \in H$, $\bar{x}$ is a stationary point of (4) if and only if the inclusion $0 \in \partial f(\bar{x}) + \rho(\bar{x} - y)$ holds, or equivalently

$$\bar{x} \in \left(I + \frac{1}{\rho}\partial f \right)^{-1} y, \quad (5)$$

with I the identity operator.

Returning to our original problem (3), where, for the sake of brevity, we let $N = 2$, if we further assume that a solution to this problem exists then it follows by Fermat's theorem that $\bar{x}$, a minimizer of (3) must satisfy

$$0 \in \partial f_1(\bar{x}) + \partial f_2(\bar{x}). \quad (6)$$

By exploiting its additive structure the above inclusion can then equivalently be recast as a fix-point iteration in a number of different ways, resulting in a variety of different proximal splitting algorithms, most notably the *Forward-Backward* and *Douglas-Rachford* splitting schemes [7].

In this paper we restrict ourselves to the latter, for which the inclusion (6) can be expressed as

$$\bar{x} + \frac{1}{\rho}\partial f_2(\bar{x}) \in \left(I + \frac{1}{\rho}\partial f_1 \right)^{-1} (I + \partial f_2)(\bar{x}) + \frac{1}{\rho}\partial f_2(\bar{x}), \quad (7)$$

arriving at the fix-point iterations,

$$z^{t+1} = \text{prox}_{f_1/\rho}(x^t), \quad (8)$$

$$x^{t+1} = x^t - z^{t+1} + \text{prox}_{f_2/\rho}(2z^{t+1} - x^t) \quad (9)$$

In addition to its simplicity these above expressions have the further desirable property of preserving separability. If f_1 or f_2 are separable functions then so are the corresponding proximity operators in (8)-(9). It is this attribute that allows for the distributability of the algorithm proposed in the following section.

It should also be mentioned that strong connections between proximal splitting methods and a number of already existing classes of algorithms have been established. For instance, the popular Alternating Direction Method of Multipliers [4] can be shown to be equivalent to a Douglas- Rachford iteration applied to the dual problem. In fact methods such as the Split Bregman and alternating Split Bregman algorithm [13], the augmented Lagrangian methods [4] and Projected Landweber algorithm, to name a few, can all be viewed as special instances of the classical proximal splitting methods; see [7, 21] for more details.

In some cases it is of interest to only realize a Douglas-Rachford splitting scheme on a subset of the variables. This can be shown to benefit both computational and memory requirements of the standard Douglas-Rachford algorithm, for instance if one or both of the entering functions in (3) are separable or invariant to some of the entries of x.

Let H_1 and H_2 be some partition of the Hilbert space H, i.e. $H = H_1 \times H_2$. Then the partial proximity operator $\text{prox}^\dagger{}_f : H_2 \to H$, of $f : H \to \mathbb{R}$, is defined as

$$\text{prox}^\dagger{}_{f/\rho}(y) = \underset{\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right] \in H}{\arg\min} \left(f(x_1, x_2) + \frac{\rho}{2}||x_2 - y||^2 \right). \quad (10)$$

Following as in (4), for $\bar{x} = \text{prox}^\dagger{}_{f/\rho}(y)$ the equivalent inclusion becomes

$$\bar{x} = \left(\left[\begin{smallmatrix} 0 \\ I \end{smallmatrix}\right] + \frac{1}{\rho}\partial f \right)^{-1} \left[\begin{smallmatrix} 0 \\ y \end{smallmatrix}\right]. \quad (11)$$

Here we assume that f is strongly convex in x_1, ensuring the existence of the operator $(\left[\begin{smallmatrix} 0 \\ I \end{smallmatrix}\right] + \frac{1}{\rho}\partial f)^{-1}$.

The inclusion (7) can be modified accordingly (omitted here), thus arriving at a *partial Douglas-Rachford* splitting

$$z^{t+1} = \text{prox}^\dagger{}_{f_1/\rho}(x_2^t), \quad (12)$$

$$x^{t+1} = x^t - z^{t+1} + \text{prox}^\dagger{}_{f_2/\rho}(2z_2^{t+1} - x_2^t) \quad (13)$$

The convergence properties for this partial formulation can be shown to be similar to that (8)-(9) with only some minor additional assumptions, see [7].

4. Distributed Bundle Adjustment

We are now ready to derive our proposed formulation of a distributed bundle adjustment algorithm for solving the Bundle Adjustment problem (1). We will show that reformulating the problem in a very specific way allows us to apply the partial proximal splitting framework of section 3 in a straight forward and distributable manner.

First we split the m images into l disjoint partitions. Let $c_k \in \{1, ..., m\},\ k = 1, ..., l$ with $\bigcup_k c_k = \{1, ..., m\}$ and $c_i \cap c_j = \emptyset,\ \forall i \neq j$. Next, introduce ml additional latent variables denoted $\bar{X}_j^k \in \mathbb{R}^3,\ j = 1, ..., m, k = 1, ..., l$. To simplify notation it will prove convenient to also define the following latent visibility matrix

$$\bar{w}_j^k = \begin{cases} 1, & \exists i \in c_k \ s.t.\ w_{ij} = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (14)$$

Using the indicator function (for a set $\mathcal{S}$), here defined as

$$\iota_{\mathcal{S}}(x) = \begin{cases} \infty, & x \notin \mathcal{S}, \\ 0, & x \in \mathcal{S}, \end{cases} \tag{15}$$

we can now rewrite (1) as the equivalent problem,

$$\min_{\substack{P_i,\in\mathcal{Q} \\ \bar{X}_j^k,}} \sum_{\substack{1\leq k\leq l \\ 1\leq j\leq n \\ i\in c_k}} w_{ij}||u_{ij} - \pi(P_i, \bar{X}_j^k)||_2^2 + \sum_{\substack{1\leq k_1\leq l-1 \\ k_1+1\leq k_2\leq l \\ 1\leq j\leq n}} \iota_{\vec{0}}\left(\bar{w}_j^{k_1}\bar{w}_j^{k_2}(\bar{X}_j^{k_1} - \bar{X}_j^{k_2})\right). \tag{16}$$

Here $\vec{0}$ denotes the zero vector. Letting

$$f_1(P, \bar{X}) = \sum_{\substack{1\leq k\leq l \\ 1\leq j\leq n \\ i\in c_k}} w_{ij}||u_{ij} - \pi(P_i, \bar{X}_j^k)||_2^2, \tag{17}$$

and

$$f_2(P, \bar{X}) = \sum_{\substack{1\leq k_1\leq l-1 \\ k_1+1\leq k_2\leq l \\ 1\leq j\leq n}} \iota_{\vec{0}}\left(\bar{w}_j^{k_1}\bar{w}_j^{k_2}(\bar{X}_j^{k_1} - \bar{X}_j^{k_2})\right). \tag{18}$$

we obtain a formulation in line with the proximal splitting of the previous section. Allowing us to apply the partial Douglas-Rachford iterations (12)-(13) to solve (16). The partial formulation was employed since, with f_2 being independent of P, the resulting subproblems then becomes particularly simple to solve, as we will show next.

First we turn to solving the proximal operators relevant to the partial Douglas-Rachford Splitting (12). With f_1 given by (17) then ${\text{prox}^{\dagger}}_{f_1/\rho} : \mathbb{R}^{3\times n\times l} \mapsto \mathcal{Q}^m \times \mathbb{R}^{3\times n\times l}$ is obtained as the minimizer of,

$$\text{prox}^{\dagger}{}_{f_1/\rho}(Z) = \arg\min_{\substack{P,\in\mathcal{Q}^m \\ \bar{X}}} \sum_{\substack{1\leq k\leq l \\ 1\leq j\leq n \\ i\in c_k}} w_{ij}||u_{ij} - \pi(P_i, \bar{X}_j^k)||_2^2 + \frac{\rho}{2}||\bar{X} - Z||_F^2. \tag{19}$$

Since the partitions c_k are disjoint we can write (19)

$$\sum_{k=1}^{l} \min_{\substack{P_i,\in\mathcal{Q}^{|c_k|}, \\ \bar{X}^k\in\mathbb{R}^{3\times n}}} \sum_{j=1}^{n}\sum_{i\in c_k} w_{ij}||u_{ij} - \pi(P_i, \bar{X}_j^k)||_2^2 + \frac{\rho}{2}\bar{w}_j^k||\bar{X}_j^k - Z_j^k||^2 \tag{20}$$

Note that the inner summand of the above problem is completely independent over k and that each of these subproblem is a total sum-of-squares problem to which a standard bundle adjustment solver is directly applicable. Consequently ${\text{prox}^{\dagger}}_{f_1/\rho}$ can be evaluated by solving l smaller, independent SfM problem in parallel. This formulation thus allows us to efficiently solve (20) by distributing this problem across an arbitrary number of processors.

The second iteration (13) with f_2 as in (18) is given by the solution to,

$$\text{prox}_{f_2/\rho}(Z) = \arg\min_{\bar{X}} \sum_{\substack{1\leq k_1\leq l-1 \\ k_1+1\leq k_2\leq l \\ 1\leq j\leq n}} \iota_{\vec{0}}\left(\bar{w}_j^{k_1}\bar{w}_j^{k_2}(\bar{X}_j^{k_1} - \bar{X}_j^{k_2})\right) + \frac{\rho}{2}||\bar{X} - Z||_F^2 \tag{21}$$

$$= \sum_{j=1}^{n}\left(\min_{\bar{X}_j} \sum_{\substack{1\leq k_1\leq l-1 \\ k_1+1\leq k_2\leq l}} \iota_0\left(\bar{w}_j^{k_1}\bar{w}_j^{k_2}(\bar{X}_j^{k_1} - \bar{X}_j^{k_2})\right) + \frac{\rho}{2}\sum_{k=1}^{l}(\bar{X}_j^k - Z_j^k)^2\right). \tag{22}$$

Here we again have a problem that is separable across j and can consequently also be solved in a distributed manner. In addition, here ${\text{prox}^{\dagger}}_{f_2/\rho}$ has a particularly convenient closed form solution. Since, for all j, it must hold that

$$\bar{w}_j^{k_1}\bar{w}_j^{k_2}(\bar{X}_j^{k_1} = \bar{X}_j^{k_2}), \;\; k_1, k_2 = 1, ..., l, \tag{23}$$

at the solution of (22). Then this problem can be rewritten equivalently as a equality constrained least-squares problem for which the following close form solution holds

$$\left[\text{prox}_{f_2/\rho}(Z)\right]_j^k = \begin{cases} \frac{\sum_{k=1}^{l}\bar{w}_j^k z_j^k}{\sum_{k=1}^{l}\bar{w}_j^k}, & \bar{w}_j^k = 1, \\ z_j^k, & \text{otherwise} \end{cases} \tag{24}$$

The above expression can be obtained through variable substitution in (22) followed by setting the gradient of the resulting expression to zero. We summarize our proposed method in algorithm 1.

4.1. Convergence Analysis

Directly applying proximal splitting to a non-convex problem does in general not guarantee convergence as the proximity operator is no longer firmly non expansive, a property which the theoretical analysis of this class of algorithms rests heavily on. An exception is the work in [12, 22], which analyses the Forward-backward splitting on non-convex problems. However, very little seems to have been published regarding the convergence of Douglas-Rachford splitting applied to non-convex problems.

Here we are able to make provable statements regarding the convergence of the proposed framework. It can be

shown that with a lower bound on the depth of the reconstructed scene and with ρ sufficiently large then algorithm 1 will converge to a local minima of (1). We have the following main theorem, the proof can be found in the supplementary material.

Theorem 4.1 *(Convergence) With f_1 and f_2 as in (17)-(18), let $\{(P^{(t)}, X^{(t)})\} \subset R^{3\times4\times m} \times R^{3\times n}$ denote a sequence generated by Algorithm 1. Assuming that local minimizers of (1) exists, are unique and that the scene depth d is bounded from below by $d = P_{i3}^{(t)} \begin{bmatrix} X_i^{(t)} \\ 1 \end{bmatrix} \geq d_{\min} > 0$, $i \in [1, m]$. Then there exists a $\mathbb{R} \ni \rho_{\min} > 0$ such that if $\rho^{(t)} > \rho_{\min}$ (with $t \geq T$ for some fixed T) then Algorithm 1 is guaranteed to converge and every limit point of $\{(P^{(t)}, X^{(t)})\}$ is a local minimizer of (1).*

As there is no restriction on the permissible partitions of the cameras we can thus construct subproblems as small as desirable, down to a single camera per process, regardless of scene structure and camera connectivity. If solved in a distributed manner with sufficient nodes, this means that the complexity of this bundle adjustment formulation is now only $\mathcal{O}(\max_i(|c_i|)^3)$, a significant improvement over $\mathcal{O}(m^3)$, the complexity of the conventional bundle adjustment formulation. However, this improved complexity does come at a cost. The convergence rates of proximal splitting methods are typically decidedly inferior to those of algorithms such as the Levenberg-Marquardt. In addition, this difference in convergence speed consistently grows with the degree of parallelization. This trade-off between complexity and rate of convergence will be further discussed in section 5.

4.2. Implementation Details

In this section we will briefly discuss some of the practical aspects of implementing algorithm 1. As mentioned in the previous section, the convergence of the proposed method does not necessarily hold in the non-convex setting, unless ρ is sufficiently large, as shown in theorem 4.1. Hence, correctly setting the value of ρ is of importance, a task that is made more demanding owing to the established relationship between the value of this parameter and the rate of convergence of proximal splitting algorithms. Setting ρ too small will not result in a convergent algorithm and setting it too large will render the algorithm intractably slow. A commonly accepted approach is to modify this penalty parameter during the progress of the algorithm and start with a small initial value of ρ to ensure fast convergence in the initial stages and then gradually increase it to certify overall convergence to a stationary point of the problem at hand. This simple scheme was used in this work $\rho^{t+1} = (1+\eta)\rho^t$, where typical choices of these parameters were $\rho^0 = 10^{-3}$ and $\eta = 0.01$.

Algorithm 1 Distributed Bundle Adjustment

input:
u (image measurements),
$\{\rho^{(t)}\}_{t=0}^{\infty}$, $\rho^{(t)} \in \mathbb{R}^+$ (proximal weighting),
$\{c_1, ..., c_l\}$ (camera partitions)
initialize:
$P^{(0)}$, $X^{(0)}$ (initial estimate of cameras & 3D points)
$\bar{X}^{k(0)} \leftarrow X^{(0)}$, $k = 1, ..., l$ (initial latent variables)
$t = 0$
repeat
- $\{P^{(t+1)}, \bar{X}^{(t+1)}\} \leftarrow \text{prox}^{\dagger}_{f_1/\rho^{(t)}}(Z^{(t)})$
  Solve by evaluating,
  $\{P^{(t+1)}_{i\in c_k}, \bar{X}^{k(t+1)}\} \leftarrow \text{prox}^{\dagger}_{f_1/\rho^{(t)}}(Z^{k(t)})$,
  in parallel for all $k \in [1, l]$, as in (20).
- $\{Z^{(t+1)}\} \leftarrow Z^{(t)} - \bar{X}^{(t+1)} + \text{prox}_{f_2/\rho^{(t)}}(2\bar{X}^{(t+1)} - Z^{(t)})$
  Here the proximity operator is separable in j and $\text{prox}_{f_2/\rho}$ is given in closed form by (24).
- $X_j^{(t+1)} \leftarrow \bar{X}_j^{k(t+1)}$, for any $k \in [1, l]$ such that $\bar{w}_j^k = 1$, $j = 1, ..., n$.
- $t \leftarrow t + 1$

until convergence

As stated in section 4 algorithm 1 might at the onset give the appearance of having quite substantial memory requirements, since the introduction of the latent variables $\bar{X}$ effectively increases the number of variables by a factor of (almost) l. These memory requirements can however be greatly reduced by noting that the individual subproblems (20) are in fact invariant to any latent 3D point not visible in the current camera partition. That is, for any given $k = 1, ..., l$, we do not need to store the value of $\bar{X}_j^k$ unless $\bar{w}_j^k = 1$. Consequently, not only does each sub-problem here involve a smaller number of cameras, it only needs to consider the subset of the structure visible from its reduced set of cameras.

Finally, we point out the fact that since our proposed method is iterative in nature, it is not necessary to solve each bundle adjustment subproblem to machine precision. In fact, in our current implementation we terminate after a single bundle adjustment iteration has resulted in an improved reprojection error.

5. Experimental Validation

In this section we present an empirical validation of our proposed algorithm. In our experiment we were mainly interested in investigating the computational gain achieved by the presented formulation over that of conventional formulations. This proposed algorithm is a meta-algorithm, one that does not rely on a specific bundle solver, and we are

Dataset	Stockholm City Hall	UWO	Alcatraz Courtyard	Vercingetorix Statue	Eglise du Dome	Arc de Triomphe
Cameras	43	57	67	68	85	173
3D-Points	47,833	8,880	23,674	10,789	84,792	35,165
Observations	162,782	27,309	68,615	49,378	615,227	369,765
Conventional BA						
# Iter	9	9	12	9	12	10
Time	19.01s	6.79s	76.04s	16.35s	167.51s	114.33s
Distributed (#cams/node)	21	28	33	34	28	34
# Iter	13	9	41	14	18	18
Time	12.35s	2.62s	38.37s	6.68s	49.73s	14.41s
Distributed (#cams/node)	7	7	11	17	12	17
# Iter	15	22	60	14	23	19
Time	4.99s	1.59s	16.92s	2.21s	19.24s	5.92s
Distributed (#cams/node)	3	4	6	4	4	10
# Iter	17	23	69	17	33	32
Time	2.92s	0.94s	11.83s	0.69s	9.36s	5.35s

Table 1: Summary of the datasets and results of section 5.1.

primarily interested in determining the potential speed-ups gained from the distributed formulation over its conventional counterpart. While it would be interesting to test the speed-up given by our meta-algorithm on more efficient implementation of bundle solvers [1, 14, 24], we leave this as future work. We argue, nonetheless, that the *relative* gain in computational efficiency would essentially remain unchanged and that this experimental protocol is a legitimate and suitable proxy for highlighting the potential of the proposed method.

The initialization of refinement methods such as bundle adjustment is of crucial importance to the results they produce. As the structure-from-motion problem addressed in this work are non-convex, there are multiple local undesirable minimas present. If the starting point is not good enough, the final result will inevitably be poor, but this is just as true for the conventional bundle adjustment as it is for the distributed variant proposed here. However, as we are mainly concerned with computational aspects of (1), we will simply assume that a sufficiently good initialization has been made available to us by some other means. In the below we construct initial estimates by manually adding moderate amounts of Gaussian noise to the reconstructions supplied with each the datasets considered.

5.1. Synthetic Experiments

In order to eliminate outside influences on the computational resources, such as external jobs demanding access to the distributed system, our initial experiments are conducted in a synthetic setting. We emulated a distributed architecture by solving algorithm 1 in serial and subsequently obtaining the timing information as the maximum of the time requirements over all the individual subproblems. This allows us to study the computational requirements separate from the communication overhead under controlled and repeatable conditions.

We carried out experiments on six different real world datasets, all available from [19, 20]. A summary of these datasets are given in table 1.

A full and calibrated Euclidean reconstructions of each of these image sequences was carried out using our proposed algorithm. We compared our approach to that of a standard bundle adjustment solver. Both algorithms were implemented in Matlab and run on a standard Intel i7, 3GHz machine with 256GB of RAM. The same Matlab implementation of bundle adjustment was employed to solve the conventional formulation as well as the subproblems of algorithm 1. The results of these reconstructions[1] are shown in figure 1 and table 1.[2]

These initial results appear very convincing. The left column of graphs in figure 1 shows how the reprojection error decreases over time (in seconds); for the conventional bundle adjustment algorithm and the distributed formulation, with varying numbers of nodes employed.

In these plots, the distributed bundle adjustment formulation outperforms the conventional formulation by orders of magnitude. It also appears evident that the more nodes used the faster convergence is achieved. Note that although this outcome might be unsurprising and intuitively correct, it is neither an obvious result nor does it follow trivially from the parallelization of the problem. Recall from section 4.1

[1] Averaged over 100 iterations with different initializations.

[2] Convergence is achieved once the magnitude of the normalized gradient of (1) is less than 10^{-3}

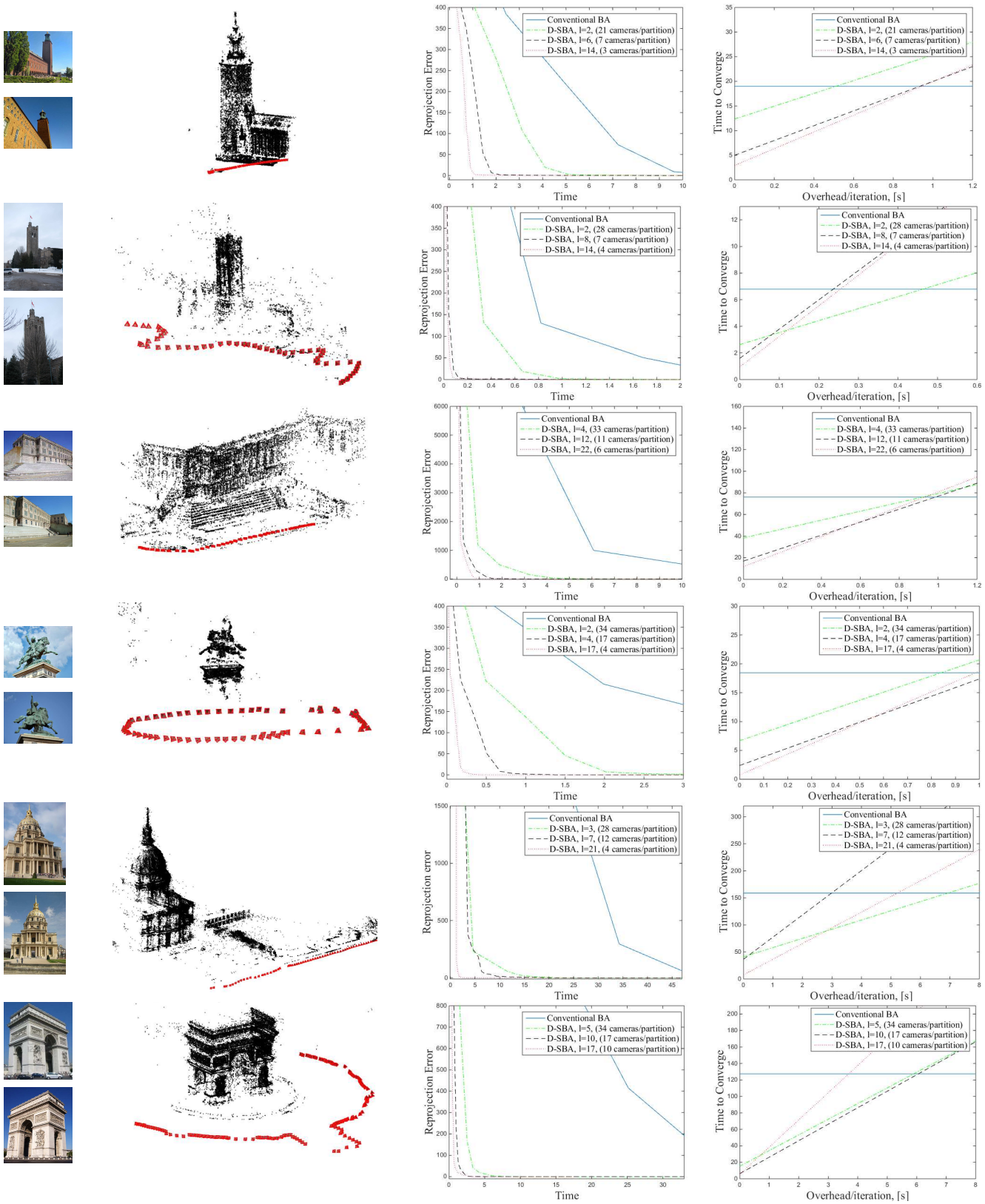

Figure 1: Euclidean reconstruction of the datasets of table 1. The left graph shows the *Reconstruction Error* vs. *Time* for Conventional Bundle Adjustment as well as the proposed Distributed Bundle Adjustment (for a varying number of nodes). The right graph shows the *Time to Convergence* vs. *Communication Overhead per Iteration*.

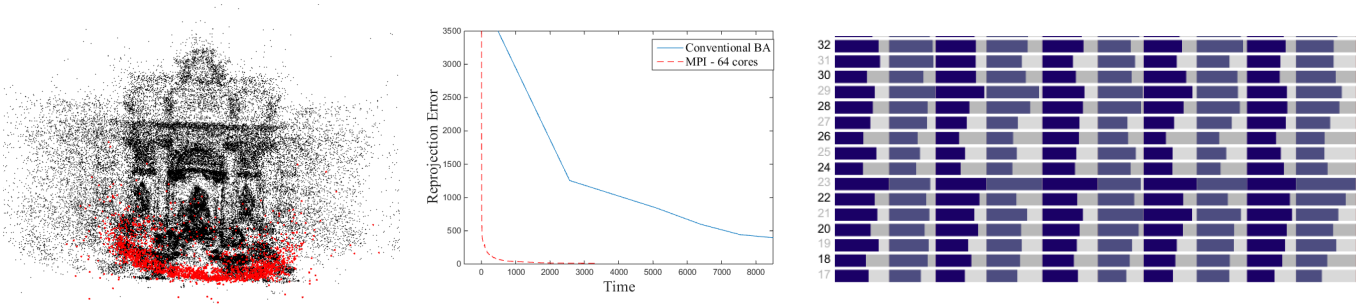

Figure 2: Euclidean reconstruction of the final-961 dataset. (Left) 3D reconstruction. (Center) Reconstruction Error vs. Time. (Right) A Gantt chart of (a portion) of the 64 nodes used. Here each row corresponds to individual nodes, each column to one iteration. The length of each blue bar correlates to the time required to solve the subproblem on that specific node.

that although the algorithmic complexity of algorithm 1 decreases with the number of nodes l, the rate of convergence is inversely proportionate to l. However, these results do seem to suggest that the speed-ups gained from parallelization clearly outweighs the reduced convergence rates.

However, these results do not take any overhead, such as communication across nodes, into account. The actual amount of overhead is entirely platform dependent. We argue that it is justifiable to assume that this overhead is constant across the iterations. Hence, we can study how different levels of overhead effects the overall performance of algorithm 1. The rightmost column of graphs in figure 1 shows how the time to convergence varies with the overhead per iteration. From these graphs it is evident that the optimal choice of the number of nodes will constitute a trade-off between the complexity of the subproblems and the amount of overhead inherent to the distributed system used. It should be noted that for conventional bundle adjustment to outperform the distributed algorithm in these experiments, an overhead of about 400%-600% is required.

5.2. MPI Experiments

To verify these above results on an actual distributed platform, we implemented the proposed algorithm 1 using C++, MPI and OpenBLAS, and evaluated its performance on a Beowulf cluster comprising 33 heterogeneous nodes. Each node was connected by Gigabit Ethernet and were typically 2 x 8 core 2.7Ghz machines with 128GB of RAM, although a number of nodes had up to 48 cores and 256GB of RAM. MPI was used to distribute the problem to a varying number of processes run across the cluster. Each subproblem (20) was solved in parallel, and the minimum $\bar{X}$ sent to the root node in order to compute the global solution (24).

This implementation was evaluated on the *Final* sequence of [3], a dataset consisting of 961 cameras, 187, 103 3D-points and 1, 692, 975 observations. Here algorithm 1 was distributed across 64 machines, the conventional bundle adjustment algorithm was run on a single 32 core machine, see figure 2. As can be seen, these results are in strong agreement with the experiments in the emulated setting of the previous section. It should also be noted that in this experiment and on this system the overhead constituted approximately 0.1% of the entire computational time.

6. Conclusion

In this paper we have proposed a consensus formulation for distributed bundle adjustment. The resulting meta-algorithm, here derived from proximal splitting methods, is both computationally efficient as well as remarkably straightforward to implement. Our empirical validation clearly demonstrates the potential of our distributed approach, as it appears to be able to offer significant speed-ups of orders of magnitude over the conventional bundle adjustment formulation.

Future work includes further theoretical as well as empirical analysis and verification of the proposed formulation. We also intend to procure access to the necessary infrastructure required to carry out a comprehensive experimental evaluation on a large number of considerably more demanding datasets across a wide range of architectures.

Acknowledgements

This research was supported by the Australian Research Council through grants DE130101775 and DP160103490.

References

[1] S. Agarwal and K. Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc. 1, 6

[2] S. Agarwal, N. Snavely, S. M. Seitz, and R. 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. 1

[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Twelfth IEEE International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan, September 2009. IEEE. 1, 8

[4] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1 edition, 1996. 3

[5] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. 2

[6] M. Byrod and K. Astrom. Conjugate gradient bundle adjustment. In *Proceedings of the 11th European conference on Computer vision: Part II*, ECCV'10, pages 114–127, Berlin, Heidelberg, 2010. Springer-Verlag. 1

[7] P. L. Combettes and J.-C. Pesquet. Proximal Splitting Methods in Signal Processing. In H. Bauschke, R. Burachik, P. Combettes, V. Elser, D. Luke, and H. E. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011. 3

[8] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione. Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11):1847–1864, 2010. 2

[9] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *Photogrammetric Computer Vision (PCV)*. ISPRS, Sept. 2006. 1

[10] A. Eriksson and M. Isaksson. Pseudoconvex proximal splitting for l-infinity problems in multiview geometry. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 4066–4073. IEEE, 2014. 2

[11] A. Eriksson, M. Isaksson, and T.-J. Chin. High breakdown bundle adjustment. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 310–317. IEEE, 2015. 2

[12] M. Fukushima and H. Mine. A generalized proximal point algorithm for certain non-convex minimization problems. *International Journal of Systems Science*, 12(8):989–1000, 1981. 4

[13] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Science*, 2(2):323–343, Apr. 2009. 3

[14] Y. Jeong, D. Nistér, D. Steedly, R. Szeliski, and I.-S. Kweon. Pushing the envelope of modern methods for bundle adjustment. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(8):1605–1617, 2012. 1, 6

[15] Y.-D. Jian, D.-C. Balcan, and F. Dellaert. Generalized subgraph preconditioners for large-scale bundle adjustment. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, editors, *ICCV*, pages 295–302. IEEE, 2011. 1

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

[17] J. J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes Rendus de l'Académie des Sciences (Paris), Série A*, 255:2897–2899, 1962. 3

[18] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3d reconstruction. In *ICCV*, pages 1–8. IEEE, 2007. 1

[19] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. In *Scandinavian Conference on Image Analysis, SCIA 2011*, 2011. 6

[20] C. Olsson, A. Eriksson, and R. Hartley. Outlier removal using duality. 2010. 6

[21] S. Setzer. Split bregman algorithm, douglas-rachford splitting and frame shrinkage. *Scale space and variational methods in computer vision*, pages 464–476, 2009. 3

[22] S. Sra. Scalable nonconvex inexact proximal splitting. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 530–538. Curran Associates, Inc., 2012. 4

[23] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment ? a modern synthesis. 1883:298–372, 2000. 1

[24] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 3057–3064, Washington, DC, USA, 2011. IEEE Computer Society. 1, 6