Probabilistic Multi-view Correspondence in a Distributed Setting with No Central Server

Shai Avidan[1], Yael Moses[1], and Yoram Moses[2]

[1] The Efi Arazi school of Computer Science
The Interdisciplinary Center, Herzliya, Israel
{avidan,yael}@idc.ac.il
[2] Department of Electrical Engineering,
Technion, Haifa, 32000 Israel
moses@ee.technion.ac.il

Abstract. We present a probabilistic algorithm for finding correspondences across multiple images. The algorithm runs in a distributed setting, where each camera is attached to a separate computing unit, and the cameras communicate over a network. No central computer is involved in the computation. The algorithm runs with low computational and communication cost. Our distributed algorithm assumes access to a standard pairwise wide-baseline stereo matching algorithm ($\mathcal{WBS}$) and our goal is to minimize the number of images transmitted over the network, as well as the number of times the $\mathcal{WBS}$ is computed. We employ the theory of random graphs to provide an efficient probabilistic algorithm that performs $\mathcal{WBS}$ on a small number of image pairs, followed by a correspondence propagation phase. The heart of the paper is a theoretical analysis of the number of times $\mathcal{WBS}$ must be performed to ensure that an overwhelming portion of the correspondence information is extracted. The analysis is extended to show how to combat computer and communication failures, which are expected to occur in such settings, as well as correspondence misses. This analysis yields an efficient distributed algorithm, but it can also be used to improve the performance of centralized algorithms for correspondence.

1 Introduction

Settings with large numbers of cameras are spreading in many applications of computer vision, such as surveillance, tracking, smart environments, etc. [11, 7,16,5] Existing vision applications in a multi-camera setting are based on a central computer that gathers the information from all cameras, and performs the necessary computations. In some cases, part of the computation is performed locally at the cameras' sites (e.g., feature detection or local tracking), and then the overall solution is computed by the central computer.

Controlling a large application involving many cameras by a central server has the advantage that the computation, once performed, is reliable and can utilize all of the information in one place. But it has disadvantages that often

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3024, pp. 428–441, 2004.

outweigh the advantages. First, since many vision applications require a significant amount of computation, centralized solutions are often not scalable: their performance degrades as the number of sites grows. In addition, the server can become a communication hot-spot and possible bottleneck. Finally, the central server is a single point of failure. If it fails or is unreachable for a while, the applications it governs may fail. Moreover, the possibility of temporary failures grows when the system is dynamic and, for example, cameras occasionally join or leave the system, or move from one place to another. These disadvantages of the centralized approach motivate an investigation of techniques for solving computer vision applications that are *not* based on a central server. The processing units at different cameras communicate among themselves and perform whatever computations may be needed in the application. The scenario in which many of the cameras are attached to reasonably powerful computing devices is quite realistic, and supports this approach.

In this paper we present a distributed approach to computing multi-image correspondence in a multi-camera setting. Such correspondence forms the basis of many important visual tasks, such as calibration, 3D scene reconstruction, and tracking. One way to compute multi-image correspondence is by computing correspondence between pairs of images, using a *Wide-baseline Stereo* ($\mathcal{WBS}$) algorithm.[1] Computing $\mathcal{WBS}$ for *all* pairs, which clearly guarantees obtaining full correspondence, is costly in terms of both communication and computation. Moreover, the computation becomes intractable when a large setting with hundreds or even thousand of cameras is considered. An alternative is to perform $\mathcal{WBS}$ computations on only some of the pairs, and then use the transitivity of correspondence to obtain further correspondence information among images that were not compared directly. A key aspect of such an algorithm is the choice of image pairs to which the $\mathcal{WBS}$ algorithm will be applied.

Our solution is distributed: every camera is involved in a limited amount of communication and performs only a *small* number of $\mathcal{WBS}$ computations. Propagation of the correspondence is performed by local communication between cameras. Nevertheless we are guaranteed that, with high probability, the full correspondence information is obtained for the vast majority of points at the end of the propagation process. Our solution can be tuned so that it will tolerate communication failures, processor failures, and failure of the $\mathcal{WBS}$ computations to identify corresponding points in overlapping images.

A key element in the efficiency of an algorithm such as ours is in the choice of which $\mathcal{WBS}$ computations to perform. We employ the theory of random graphs in order to obtain a drastic reduction in the number of such computations each camera performs. Further reduction is obtained when there is information regarding cameras that do not view overlapping regions. Finally, we tune our algorithm to capture correspondence information for points that are seen by many cameras. In a multi-image setting, the number of images in which a feature point p appears is important. We call this the *degree of exposure*, or exposure for short,

[1] We use the $\mathcal{WBS}$ computation as a black box; a better solution to $\mathcal{WBS}$ will improve the performance of our scheme.

of p. Applications that use correspondence information typically obtain much greater benefit from points with high exposure than from ones with low exposure (e.g., Bundle Adjustment [23]). Accordingly, our algorithm is designed to accept an *exposure parameter* k and will be tuned to find the correspondence information of points with degree of exposure that is greater than or equal to k. The algorithm has the following features:

1. Every camera i performs a small number s_i of $\mathcal{WBS}$ computations;
2. cameras exchange correspondence information for at most $\log_2 k$ rounds; and
3. for every feature point p with exposure degree k or more, with probability at least 0.99 all cameras that view p obtain the full correspondence information regarding p.

As we show, when there is sufficient information about the relative locations of cameras, s_i may be $c \cdot \log_2 k$ for some constant c. Since $\mathcal{WBS}$ computations are dominant in this application, the algorithm will then terminate in time that is proportional to $\log_2 k$. Our approach is probabilistic, rather than heuristic. Moreover, its success is guaranteed with high probability for every given set of images (provided that the $\mathcal{WBS}$ algorithm is error-free).

The algorithm is designed in such a way that no single failure can impact the quality of the correspondence information obtained in a significant way. Moreover, it is robust in the sense that it degrades gracefully as the number of failures grows. We extend the algorithm to handle unreliable systems with communication failures, processor failures, and failure of the $\mathcal{WBS}$ computations to identify corresponding points in overlapping images. Roughly speaking, in order to overcome a failure rate of $f < 1$ of the communication channels (resp. a portion of $f < 1$ of the cameras crashes, or a portion of $f < 1$ of the matches are false-negative errors by the $\mathcal{WBS}$), an increase of roughly $\frac{1}{1-f}$ in the number of $\mathcal{WBS}$ computations leads to the same performance as in a system with no failures. Hence, to overcome a high failure rate of 10%, the cameras need to perform only 12% more work!

While originally motivated by the quest for a distributed solution, our probabilistic analysis can be applied to reduce the number of $\mathcal{WBS}$ computations even when correspondence is computed on a single computer. That is, our algorithm can be simulated on a centralized computer (replacing the propagation step by a simple transitive closure computation) to improve the efficiency of computation of existing centralized algorithms.

The question of how to reduce the number of $\mathcal{WBS}$ computations performed in the centralized setting has been addressed by Schaffalitzky & Zisserman [21]. They suggested a heuristic approach to this problem: first single-view invariants are computed and mapped to a large feature vs. views hash table. The hash table can then guide the greedy choice of the pairs on which to compute $\mathcal{WBS}$, resulting in run-time complexity of $O(n)$ $\mathcal{WBS}$ computations, where n is the number of cameras.

This paper makes two main contributions. One is in providing a reasonably efficient solution to the multi-image correspondence problem in a distributed system with no central server and no single point of failure. The second is in

employing the theory of random graphs in order to reduce the number of $\mathcal{WBS}$ computations needed to obtain a useful amount of correspondence information.

2 Previous Work

There have recently been a number of significant advances on the subject of *Wide-baseline Stereo* (WBS), in which computations involving a small number of images are used to extract correspondence information among the images [24, 1,17,20,6,15].

Schaffalitzky & Zisserman [21] and then Ferrari *et. al.* [10] suggested methods for wide baseline matching among a large set of images (on the order of 10 to 20 cameras). They both suggested methods for extending the correspondence of two (or three) views to n views, while using the larger number of views to improve the pairwise correspondence. Their algorithms are designed to run on a central computer. Levi & Werman [12] consider the problem of computing the fundamental matrices between all pairs of n cameras, based on knowing only the fundamental matrices of a subset of pairs of views. Their main contribution is an algebraic analysis of the constraints that can be extracted from a partial set of fundamental matrices among neighboring views. These constraints are then used to compute the missing fundamental matrices.

In recent years various applications of multi-camera settings with a central computer are considered. These include various tasks such as surveillance, smart environments, tracking and virtual reality. Collins *et al.* [7,8] report on a large surveillance project consisting of 14 cameras spread over a large compound. The algorithm they used for calibration, which was based on known 3D scene points [8], was performed on a central server. The virtual-reality technology introduced by Kanade *et al.* [16,19] uses a multi-camera setup that can capture a dynamic event and generate new views of the observed scene. Again, the cameras were calibrated off-line using a central computer. Smart environments [5,13] consist of a distributed set of cameras spread in the environment. The cameras can detect and track the inhabitants, thus supporting higher-level functions such as convenient man-machine interfacing or object localization. Despite the distributed nature of these systems, the calibration of the cameras is usually done off-line on a central processor.

Karuppiah *et al.* [11], have already discussed the value of solving multi-camera computer vision problems in a distributed manner. They constructed a four-camera system and experimented with tracking and recognition in this system, showing the potential for fault-tolerance and avoiding a single point of failure.

While the literature contains little in the way of distributed solutions to computer vision applications, the literature on distributed systems and distributed computing has addressed many issues that are relevant to a task of this type. They involve methods for failure detection and fault-tolerant execution of computations, algorithms for leader election and consensus, etc. A good overview of the issues can be found in Tanenbaum and van Steen [22] and in the collection

by Mullender [18]. A comprehensive source for distributed algorithms can be found in Lynch [14] and a useful treatment of issues relating to data replication is the book by Bernstein *et al.* [2].

3 The Algorithm

We assume a set $\{1, \ldots, n\}$ of cameras overlooking a scene. Each camera has a processing unit attached to it, and the cameras can communicate over a point-to-point communication network. We further assume that the communication network is complete so that every camera can communicate directly and reliably with every other camera. We denote by M_i the number of cameras with which camera i can have corresponding points and let m_i denote the size M_i. Initially, we assume that the $\mathcal{WBS}$ computations are noise and error free: A $\mathcal{WBS}$ computation performed on a pair of images identifies two locations in the images as being corresponding exactly if there is a genuine feature point p that appears in the stated coordinates in the respective images. We relax these reliability assumptions in Section 5.

Our distributed algorithm is defined in terms of an exposure parameter k, and is designed to discover the vast majority of points with an exposure size k or more. Each camera maintains a list of its own feature points and their corresponding points in other cameras. At each propagation step, each camera, propagates any new correspondence information to all the cameras with which it has established corresponding points. Each camera has to run the following algorithm, given the exposure parameter k.

1. **Initialization**
   Randomly choose a set $S \subset M_i$ of cameras of size
   $$s_i = m_i \tau(k) \approx m_i \frac{\log k + 5}{2k}, \tag{1}$$
   and request their images.
2. **Pairwise Matching**
   For every camera j from which an image has been received, perform a $\mathcal{WBS}$ computation between i's image and j's , record its results in the local correspondence lists, and send the results to j. Concurrently, for every request for an image, send you image to the requesting camera and later record the result when you receive them in the correspondence lists.
3. **Correspondence Propagation**
   This stage proceeds in rounds of communication. In the first round, for every point $p_i = (x_i, y_i)$ in camera i's image that has been matched with more than one point by the $\mathcal{WBS}$ computations, i performs a propagation step. In every subsequent round, i performs propagation steps for every point p_i for which it received new correspondence information in the most recent round. The propagation is terminated when no new correspondence information received.

One of the main contributions of the algorithm is in equation 1 that expresses the number of $\mathcal{WBS}$ computations as a function of the exposure parameter k. As for m_i, it is equal to n in case no prior information is available, but in many cases, we do initially have information regarding which images potentially have corresponding points, and which do not. Reducing the value of m_i means a smaller number of $\mathcal{WBS}$ computations, as is evident from equation 1. Consider, for example, a situation in which cameras are located around a hill or a rooftop. They may cover the surrounding scene quite effectively, while every camera has a limited number of relevant neighbors to consider for correspondence. There is another source that may reduce the size of M_i. Some recent approaches to computing multi-view correspondence contain a preprocessing stage in which images are ranked for likelihood of correspondence (e.g., [21]). The result of such a stage can reduce the sets M_i.

4 Probabilistic Analysis of the Algorithm

The probabilistic analysis will show that the above algorithm will detect all points with exposure factor great or equal to k with probability 99%. Furthermore, it will show that only $log_2(n)$ propagation steps are needed, at most, for the algorithm to terminate.

We represent the state of information that the cameras attain regarding the multi-view correspondence of feature points by a labeled multi-graph, which we denote by G. There is a node in G for each camera. There is a labeled edge, $(\{i, j\}, p)$, between nodes i and j if p is an established corresponding point of the images of i and j. Initially, the graph has no edges. After the first phase, in which $\mathcal{WBS}$ computations are performed, the graph contains edges only among images that were compared directly by a $\mathcal{WBS}$ computation. Additional edges are added to G in the propagation phase.

Let us begin by considering the behavior of the algorithm in terms of discovering the correspondence information of a single 3D feature point p. Let us call the set of cameras that view the point p the *p-set*. All of the correspondence information regarding p will be uncovered exactly if, at the end of the propagation process, every pair of cameras in the p-set will share a p-edge. To analyze the algorithm's behavior with respect to p it is convenient to consider the *p-graph* G_p derived from G that is defined by the p-set and the p-edges of G. More formally, $G_p = (V_p, E_p)$ where V_p is p-set—the set of cameras that view p, and E_p consists of the edges $\{i, j\}$ for which $(\{i, j\}, p)$ is in G. We refer to the state of G_p after the matching step of the algorithm by $G_p(0)$, and after $r \geq 1$ rounds of propagation by $G_p(r)$.

4.1 Analysis of Propagation

We now prove that if $G_p(0)$ is connected, then propagation will uncover the full correspondence information regarding p. Moreover, this will be done within a small number of rounds of propagation.

Lemma 1. *If the distance between nodes i and j in $G_p(0)$ is d, where $2^{r-1} < d \leq 2^r$, then their distance in $G_p(r)$ is 1.*

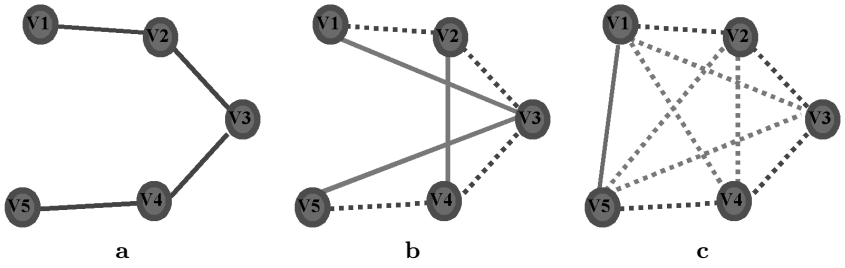

Fig. 1. (**a**) $G_p(0)$—the distance between v_1 and v_5 is 4. (**b**) $G_p(1)$—the distance is reduced to 2, and (**c**) in $G_p(2)$ the distance is 1.

Proof. Consider two vertices v_1 and v_{d+1} that are distance d apart. Let $v_1, ..., v_{d+1}$ be a path connecting these points in $G_p(0)$. In the first round of the propagation algorithm node v_2 will update node v_1 that v_3 also views the point p, and similarly node v_2 will also update node v_3 that v_1 views the point p (see Figure 1). As a result, nodes v_1 and v_3 both update their local p-lists, and the edge $\{v_1, v_3\}$ is added to G_p. In a similar manner, all edges between v_i and v_{i+2}, $i \leq d+1$, are added to G_p. It follows that after a single round of propagation, the path $v_1, v_3, v_5, ..., v_{d+1}$ connects v_1 and v_{d+1} in the graph. As a results, the distance between v_1 and v_{d+1} is shortened by a factor of two, and it is $\lceil \frac{d}{2} \rceil$. A straightforward induction shows that the distance between v_1 and v_d is reduced to 1 after $\lceil log_2(d) \rceil = r$ steps.

Corollary 1. *Suppose that $G_p(0)$ is connected. If the diameter of $G_p(0)$ is d, then $G_p(\lceil log(d) \rceil)$ is a complete graph (its diameter is 1).*

Since $d \leq k \leq n$ is guaranteed, Corollary 1 implies that there is no need to *ever* run the propagation algorithm for more than $\lceil log(n) \rceil$ rounds.

Corollary 2. *If camera i does not receive a new update regarding the point p in round r of the propagation phase, then i will never send or receive any further updates about p.*

Corollary 1 proves that propagation is guaranteed to terminate for all points within a small logarithmic number of rounds. Moreover, by Corollary 2 every camera can easily detect when its propagation phase is done.

4.2 The Number of $\mathcal{WBS}$ Computations

As we have seen, if $G_p(0)$ is connected then the propagation phase of the algorithm will discover the correspondence information regarding p. Clearly, if $G_p(0)$

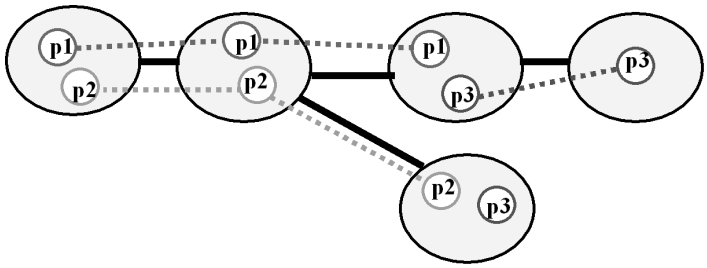

Fig. 2. The graph G with cameras that view the points p_1, p_2, and p_3. The graphs G_{p_1}, G_{p_2}, and G_{p_3} are marked with red, black, and green edges, respectively. The edges of G span each of the derived graphs G_{p_i}.

is *not* connected, then only partial information will be discovered. In this section we determine the precise number of cameras that are selected in the initialization step of the algorithm by showing that it will ensure connectedness. We base the discussion here on the theory of random graphs, which was initiated in a paper by Erdös and Rényi [9]. Consider a random process in which each of the $\binom{N}{2}$ undirected edges of a graph on N nodes is chosen independently with probability $\rho > 0$. The resulting graph is denoted by $\mathcal{G}(N, \rho)$.

Lemma 2. *(a) Let* $\hat{\rho}(N) = \frac{0.577+\ln N}{N}$. *The probability that* $\mathcal{G}(N, \hat{\rho}(N))$ *is connected tends to* 1 *as* N *tends to* ∞. *More concretely, for small values of* N *we have*

(b) Let $\rho(N) = \frac{4.61+\log N}{N}$. *The probability that* $\mathcal{G}(N, \rho(N))$ *is connected is greater than* 0.99 *for all values of* $N \leq 40,000$

The first part of the lemma is a classical result in the field, while the results in the second part are from Bollobás and Thomason[4], as they are quoted in the excellent textbook by Bollobás [3]. Since we are unlikely to be interested in computing correspondence for points that are seen by more than 40,000 cameras, the second part gives us very good bounds to work with: For our purposes, if p has exposure degree k and each pair of nodes in the p-set is chosen with probability at least $\rho(k)$ for a $\mathcal{WBS}$ computation, then we have high assurance (over 0.99 probability!) that $G_p(0)$ will be connected.

For independent probabilistic events A and B, we have that $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A)\Pr(B)$. An edge is chosen if one of its nodes selects it. In the algorithm, if every node selects the edge with probability $\tau(k)$, then we need to ensure that $2\tau(k) - \tau^2(k) \geq \rho(k)$ in order to guarantee edges are chosen with sufficient probability. The exact formula for $\tau(k)$ is thus $\tau(k) = 1 - \sqrt{1 - \rho(k)}$. However, $\tau(k)$ tends to $\frac{\rho(k)}{2}$ in the limit, and for all $k \geq 10$ we have $\tau(k) < 0.6\rho(k)$. So $\tau(k)$ is essentially $\frac{\rho(k)}{2}$.

Our analysis so far has been in terms of connectivity of the graph $G_p(0)$. Indeed, working with G_p rather than G is crucial since guaranteeing that G is connected would not immediately yield G_p's connectivity. (Figure 2 is an

example showing that not every spanning graph of G induces spanning graphs of all of the graphs G_{p_i}.) However, the correspondence algorithm works at the level of the graph G on all cameras, with no a priori knowledge about the identity of the p-sets. Working at this level, if every edge is chosen with probability at least $\tau(k)$ then, in particular, every edge among members of the p-set is chosen with this probability. The desired property for ensuring connectivity of $G_p(0)$ is thus satisfied. Actually, much more is true. Connectivity of $G_q(0)$ is ensured with high probability *at once for all points q with exposure degree k or more*! In particular, Lemma 2(b) implies that in this case the algorithm will find the correspondence information for at least 99% of these points.

In the algorithm, we guarantee that a camera i chooses each edge with probability at least $\tau(k)$ by having it randomly choose a subset of M_i of size s_i where $\frac{s}{m_i} \geq \tau(k)$. Choosing a subset of the neighbors of a predetermined size has the advantage that we can control the number of $\mathcal{WBS}$ computations that every node performs. In summary, we have

Theorem 1. *Executing the correspondence algorithm with parameter k will, with high probability, yield the full correspondence information for at least 99% of the points that have exposure degree k or larger.*

5 Dealing with System Failures

In this section we consider the properties of our algorithm when executed on an unreliable distributed system. We start with a classical analysis of processor crashes and communication failures but show that the analysis can be naturally extended to handle mis-matches by the $\mathcal{WBS}$ as simply another type of failure.

5.1 System Crashes

Let us first consider crashes. Assume that some of the cameras may crash during the operation of the algorithm. We assume further that the cameras use a timeout mechanism to identify that a processor is down. Clearly, if i is in a p-set and it crashes early on, we do not expect to necessarily discover the correspondence information regarding i's image. Define the *surviving degree* of a feature point p to be its exposure degree if we ignore the cameras that crash. Crashed cameras do not participate in the algorithm, and their crashing does not affects the interactions among the surviving cameras. The original algorithm, unchanged, is thus guaranteed to discover all information for the point with surviving degree of k.

5.2 Communication Failures

Now consider communication failures. We assume that each channel between two cameras can fail with independent probability $f < 1$, after which it stays down for the duration of the algorithm. Again, our timeout mechanism can allow the cameras to avoid being hung waiting for messages on failed lines.

The worst-case behavior of a failing communication line is to be down from the outset. Since communication line failures are independent of the choices of $\mathcal{WBS}$ computations made by the camera, the probability that an edge that is chosen by the cameras with probability ρ will also be up is $(1-f)\rho$. Hence, we can increase ρ by a factor of $\frac{1}{1-f}$ and make the random graph resulting from the joint behavior of the cameras choices and the adversary's failures have the exact same structure as it originally. This translates into the choice of a neighbor with probability $\tau'(k) = 1 - \sqrt{1 - \frac{\rho(k)}{1-f}}$ instead of $\tau(k)$. In the range of $20 \leq k \leq 50$, overcoming 10% failures requires between 13% and 11% overhead, and to overcome a huge 25% probability of failure it suffices to choose between 40% and 36% more cameras than in the fully reliable case.

This discussion is summarized as follows.

Theorem 2. *(a) Executing the correspondence algorithm unchanged with parameter k when camera crashes are possible will, with high probability, yield the full correspondence information for at least 99% of the points that have* **surviving** *exposure degree k or larger.*

(b) When communication channels may fail with probability $f < 1$, executing the algorithm with $\tau'(k) \approx \frac{1}{1-f}\tau(k)$ instead of $\tau(k)$ will, with high probability, yield the full correspondence information for at least 99% of the points that have exposure degree k or larger. Moreover, it will not require more computation than the original algorithm does.

5.3 Failure of $\mathcal{WBS}$ to Detect Matches

We next consider failures of the $\mathcal{WBS}$ computation to identify the fact that a feature point appears in two images being compared. Here we suppose that our $\mathcal{WBS}$ algorithm will fail to identify a match with independent probability $f < 1$. The situation here is very similar to the case of communication failures. Again, the probability that an edge of G_p will be discovered by the first part of the algorithm is $(1-f)\rho$ if edges of G are chosen with probability ρ. By the analysis we performed in the case of communication failures, choosing $\tau'(k)$ instead of $\tau(k)$ images to compare with will provide us with the original guarantees. This time, however, all $\tau'(k)$ computations and communications must be carried out. The total overhead is then roughly $\frac{1}{1-f}$:

Theorem 3. *When $\mathcal{WBS}$ computations may fail to identify a match with independent probability $f < 1$, executing the algorithm with $\tau'(k) \approx \frac{1}{1-f}\tau(k)$ instead of $\tau(k)$ will, with high probability, yield the full correspondence information for at least 99% of the points that have exposure degree k or larger.*

We remark that this analysis is applied to false-negative errors. Coping with false-positives—mistaken matches reported—can be done using distributed systems' techniques for handling malicious failures. This analysis is beyond the scope of this paper and is left as a topic for future work.

6 Experiments

Our analysis ensures that the algorithm will indeed recover the correspondence. However, the analysis is very conservative, and in practice smaller numbers of $\mathcal{WBS}$ computations should suffice. We validated this expectation through extensive simulations in MATLAB. Our scenario is a surveillance system in an urban setting and thus our simulated test-bed consists of a collection of orthographic cameras that are mounted on roof-tops looking down. Each camera observes all the feature points within a pre-defined distance from its position. To ensure that all the cameras form a single connected component, we enforce overlap between the image footprint of the different cameras, on the ground.

In every experiment we run the algorithm with precisely the same data, but with a different number of $\mathcal{WBS}$ operations. The experiments show that the predicted number of required $\mathcal{WBS}$ operations is indeed sufficient, but even smaller numbers can be used.

To evaluate the success of each run, we define the average number of recovered points for each exposure. A given point p is recovered if each camera in the p-set knows the identity of all cameras in the p-set. In particular, let p be a point with exposure degree k, and for every i denote by $L_i(p)$ is the size of i's correspondence list for p. Then observe that $\frac{1}{k^2}\sum L_i(p) = 1$ if p is fully recovered, and this value is smaller than 1 if p is only partially recovered.

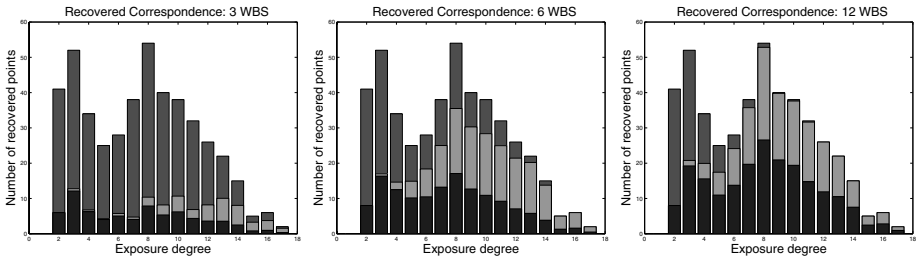

Fig. 3. The number of recovered points for each exposure degree, $\tilde{E}(k)$. Red is the number of points with given exposure. Blue is the average portion of correspondence found after the matching phase, and green is the portion after propagation. The experiment was run on a reliable system with 50 cameras and 500 points. (**a**) the results when using 3 $\mathcal{WBS}$ per camera, (**b**) when using 6 $\mathcal{WBS}$ and (**c**) when using 12.

Let $E(k)$ be the set of 3D points with an exposure degree k. Ideally, if all the points in $E(k)$ are recovered, then the number of points with exposure k is given by:

$$\tilde{E}(k) = \frac{1}{k^2} \sum_{p \in E(k)} L_i(p)$$

We use the measure $\tilde{E}(k)$ to evaluate the success of extracting the correspondence: when all the connections are recovered then $|E(k)| = \tilde{E}(k)$.

For each exposure degree k, we present three values of $\tilde{E}(k)$. The first, red bar, is the real number of points for each exposure degree. The second, green bar, is the final result of our algorithm. It is the number of computed recovered points at the end of the run. If all points for a given exposure were uncovered, then the green bar will cover the red bar. Finally, blue bar, is the value after $\mathcal{WBS}$ operation, that is only direct edges in the graph are considered.

The first experiment was designed to verify the bound on the number of required WBS operations. We generated a setup of 50 cameras and 500 points (Figure 3**a**) and simulated the behavior of the algorithm five times, changing the number of $\mathcal{WBS}$ each time. As can be seen in Figure 3**a**, using just three $\mathcal{WBS}$ does not generate enough matching points and hence the algorithm does not fully recover any of the p-sets. As the number of $\mathcal{WBS}$ performed grows, the number of fully recovered p-sets grows. In Figure 3**b** and 3**c**, we present the results of running 6 and 12 $\mathcal{WBS}$. As can be seen, the exposure degree from which full correspondence is obtained is reduced when we use the number $\mathcal{WBS}$ a camera performs increases. In Figure 3**b**, we present the smallest degree which all the p-sets of the degree were fully recovered.

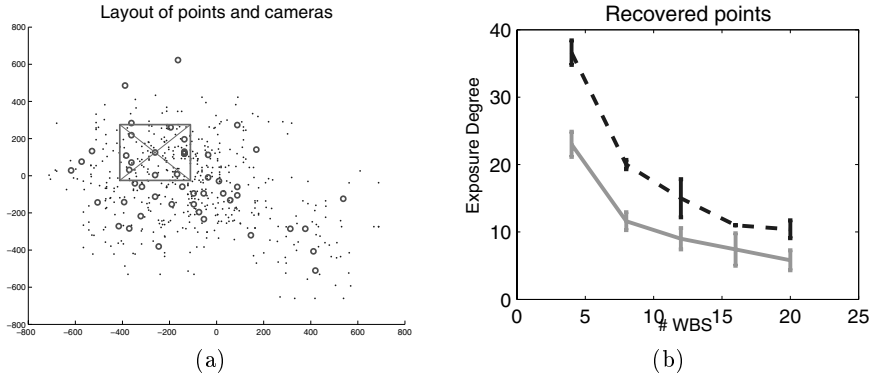

Fig. 4. (a) The setup with 50 cameras and 500 points. Each point is marked in blue, and each camera center is marked in red. The field of view of one of the cameras is marked in pink. (b) Full recovery with 20% errors of the $\mathcal{WBS}$ in red, and with no errors in black.

In the second experiment we evaluated our algorithm when the $\mathcal{WBS}$ algorithm failed to find 20% of the natchings. The same set of cameras and points as in the first experiments were used, in order to compare the performance using perfect and imperfect $\mathcal{WBS}$. The results are presented in Figure 4.

7 Summary

Visual systems consisting of a large number of geographically distributed cameras are, in particular, distributed computing systems. Information is generated and gathered at different sites, and a communication medium is used for integrating the data being gathered. We have shown how a particular application, namely image correspondence across multiple cameras, can be done in a distributed manner.

This approach allows us to use results from distributed systems theory to analyze the complexity of the distributed algorithm. In particular, we have shown what is the number of pair-wise stereo matching computations required to detect, with high probability, all points that appear in a given number of cameras. Moreover, the analysis carries naturally to the centralized case as well. Our distributed approach combines naturally failures in the communication lines, processing units and the stereo matching algorithm in a single, coherent framework. So, we have started with a distributed approach which, we believe, should be the natural way to approach large scale camera settings and ended with contributions to centralized algorithms. We plan to apply this distributed analysis to other problems in computer vision.

References

1. Baumberg, A. Reliable feature matching across widely separated views. In *Proc. of IEEE Computer Vision and Pattern Recognition* (2000), pp. I: 774–781.
2. Bernstein, P. A., Hadzilacos, V., and Goodman, N. Concurrency control and recovery in database systems. Addison Wesley, 1987.
3. Bollobas, B. *Random Graphs, Second Edition.* Cambridge University Press, 2001.
4. Bollobas, B., and Thomason, A. G. Random graphs of small order, 1985.
5. Brumitt, B., Krumm, J., Meyers, B., and S., S. Ubiquitous computing and the role of geometry, 2000.
6. Chetverikov, D., and Matas, J. Periodic textures as distinguished regions for wide-baseline stereo correspondence. In *Texture* (2002), pp. 25–30.
7. Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., and Hasegawa, O. A system for video surveillance and monitoring. *CMU-RI-TR* (2000).
8. Collins, R., and Tsin, Y. Calibration of an outdoor active camera system. In *Proc. of IEEE Computer Vision and Pattern Recognition* (1999), pp. 528–534.
9. Erdos, P., and Renyi, A. On random graphs i, 1959.
10. Ferrari, V., Tuytelaars, T., and Gool, L. V. Wide-baseline muliple-view correspondences. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* (June 2003).
11. Karuppiah, D., Zhu, Z. Shenoy, P., and Riseman, E. A fault-tolerant distributed vision system architecture for object tracking in a smart room. In *In International Workshop on Computer Vision Systems* (2001).
12. Levi, N., and Werman, M. The viewing graph. In *Proc. of IEEE Computer Vision and Pattern Recognition* (2003).

13. Lopez de Ipina, D., Mendonca, P. R. S., and Hopper, A. Trip: a low-cost vision-based location system for ubiquitous computing. In *Personal and Ubiquitous Computing Journal* (2002), vol. 6.
14. Lynch, N. A. *Distributed Algorithms*. MIT Press, 1996.
15. Matas, J., Burianek, J., and Kittler, J. Object recognition using the invariant pixel-set signature. In *The British Machine Vision Conference* (2000).
16. Narayanan, P. J. Virtualized reality: Concepts and early results. In *The IEEE Workshop on the Representation of Visual Scenes, (in conjunction with ICCV'95)* (1995).
17. Pritchett, P., and Zisserman, A. Wide baseline stereo matching. In *Proc. International Conference on Computer Vision* (1998), pp. 754–760.
18. S., M., Ed. *Distributed Systems*. Addison Wesley, 1993.
19. Saito, H., Baba, S., Kimura, M., Vedula, S., and Kanade, T. Appearance-based virtual view generation of temporally-varying events from multi-camera images in the 3d room. In *Proc. of Second International Conference on 3-D Digital Imaging and Modeling* (1999).
20. Schaffalitzky, F., and Zisserman, A. Viewpoint invariant texture matching and wide baseline stereo. In *Proc. International Conference on Computer Vision* (2001), pp. II: 636–643.
21. Schaffalitzky, F., and Zisserman, A. Multi-view matching for unordered image sets, or how do I organize my holiday snaps? In *Proc. of European Conference of Computer Vision* (2002).
22. Tanenbaum, A. S., and van Steen, M. *Distributed Systems Principles and Paradigms*. Pearson Education publisher, 2001.
23. Triggs, W., McLauchlan, P., Hartley, R., and Fitzgibbon, A. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice* (1999), pp. 298–372.
24. Tuytelaars, T., and Van Gool, L. Wide baseline stereo matching based on local, affinely invariant regions. In *The British Machine Vision Conference* (2000).