Centralized and Distributed Multi-view Correspondence *

Shai Avidan[1], Yael Moses[2], and Yoram Moses[3]

[1] The Efi Arazi School of Computer Science
The Interdisciplinary Center, Herzliya, Israel
avidan@idc.ac.il
[2] The Efi Arazi School of Computer Science
The Interdisciplinary Center, Herzliya, Israel
and National ICT, Australia
yael@idc.ac.il
[3] Department of Electrical Engineering,
Technion, Haifa, 32000 Israel
moses@ee.technion.ac.il

Abstract. A probabilistic algorithm is presented for finding correspondences across multiple images in systems with large numbers of cameras and considerable overlap. The algorithm employs the theory of random graphs to provide an efficient probabilistic algorithm that performs Wide-baseline Stereo (WBS) comparisons on a small number of image pairs, and then propagates correspondence information among the cameras. A concrete mathematical analysis of its performance is given. The algorithm is extended to handle false-positive and false-negative failures of the WBS computations. We characterize the detectability of the existence of such failures, and propose an efficient method for this detection. Based on this, we propose a heuristic method for discarding false matches, and demonstrate its effectiveness in reducing errors.
Since in many multi-camera applications cameras are attached to processors that handle local processing and communication, it is natural to consider *distributed* solutions that make use of the local processors and do not use a central computer. Our algorithm is especially suited to run in a distributed setting. If the local processors are sufficiently powerful, this allows an order of magnitude increase in computational efficiency. More importantly, a distributed implementation provides strong robustness guarantees, and eliminates the existence of a single point of failure that is inherent when the application is coordinated by a central computer. We show how to efficiently overcome processor crashes and communication failures with a minimal reduction in the quality of the algorithm's results.

1 Introduction

Settings with large numbers of cameras are spreading in many computer vision applications such as surveillance, tracking, smart environments, etc. [13, 8, 22, 5]. Common to all these tasks is the need to aggregate information across multiple sensors. A central problem involving such aggregation is the correspondence problem, in which points, blobs, objects, or events that depict in different images the same items in the real world are matched. For example, a reconstruction application requires finding the matching points of the 2D projections of a 3D point, whereas a surveillance algorithm is interested in matching 2D blobs, which may correspond to moving pedestrians, across multiple views. This paper focuses on the problem of computing multi-image *point* correspondence in a large multi-camera setting

* This paper is an extension of [1], which appeared in ECCV-04.

with considerable overlap. Such correspondence forms the basis of many important visual tasks such as calibration, 3D scene reconstruction, and tracking. We explicitly treat point correspondence, but our techniques can be extended to other forms of correspondence.

One way to compute multi-image correspondence is by computing correspondence between pairs of images, using a *Wide-baseline Stereo* (WBS) algorithm.[4] Computing WBS on *all* pairs can require significant computational resources. Being quadratic in the number of images, such an approach does not scale well, and it can become intractable for a large setting with hundreds or even thousands of cameras. An alternative is to perform WBS computations on only some of the pairs, and then use the transitivity of correspondence to obtain the correspondence information among images that were not compared directly. As a result, the required computation can be reduced considerably. The choice of an appropriate set of pairs of images on which to perform WBS is a key to ensuring that correspondence information is not lost. Note that it is not always possible to reduce the number of pairwise comparisons and still obtain *all* of the correspondence information. For example, if a given feature point appears in exactly two images, then the correspondence of these image points will be discovered only if this specific pair of images is compared directly. We define the *exposure (degree)* of a given 3D point P to be the number of cameras that view P. Often, however, feature points with low exposure (appearing in few cameras) are less interesting. Many computer vision applications in a multi-image setting that make use of correspondence information obtain much greater benefit from feature points with high exposure than from ones with low exposure (e.g., Bundle Adjustment [29]). Moreover, in monitoring and surveillance applications covering a large area such as a sports stadium, camera images will have considerable overlap, and hence the exposure degree of feature points will be high.

In this paper we employ the theory of random graphs to obtain a randomized algorithm for multi-view correspondence. The algorithm accepts a user-defined exposure parameter k. It performs $\frac{(\log k+4.6)}{k}$ as many comparisons as the naive algorithm, which compares all pairs of images, would perform. For every 3D feature point Q with exposure k or more, the algorithm is guaranteed to recover the full correspondence information for Q with probability 0.99. Thus, for values of $k \geq 8$ the algorithm provides a definite improvement. A much greater improvement is obtained if capturing most but not all correspondence information suffices. This is illustrated empirically by the exeriments presented in Section 6. As we discuss below, when the algorithm is implemented in a distributed fashion, the WBS comparisons are performed in parallel by many cameras, and the total time is further reduced by a factor of the number of cameras n.

Our algorithm compensates for reducing the number of WBS computations by propagating correspondence information which is justified by the transitivity of correspondence. Such transitive inference is very sensitive to errors in the WBS computations. Due to propagation, a false positive error can result in mistakenly identifying many unrelated pairs of image points. The second part of this paper provides a characterization of when the existence of false positive and false negative matches can be detected, accompanied by an efficient heuristic method for computing this fact. False negative matches are easy to overcome, while false positive matches are more delicate. Significantly, however, our heuristic error correction mechanism is very effective in identifying and removing a majority of false-positive errors. The error correction phase typically brings the portion of false positive errors to be a fraction of the probability at which false-positive errors are produced by the original WBS computation. This is achieved based solely on the structure of the matching relation generated by the WBS comparisons, making strong use of propagation, which in this case plays a positive role. Our heuristic can, in prin-

[4] We use the WBS computation as a black box; a better solution to WBS will improve the performance of our scheme.

ciple, be enhanced and improved by the use of additional standard geometric constraints to identify false-positive matches.

One of the central motivations for this paper is an investigation of distributed solutions to the correspondence problem. Many vision applications in multi-camera settings 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. Such use of a central server has a number of distinct disadvantages. One is the problem of scalability of the computational task as the number of cameras grows. Another is that 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. Additional temporary failures are more prevalent when the system is dynamic and, for example, cameras occasionally join or leave the system, or move from one place to another. This makes central control even more problematic. The disadvantages of the central server motivate distributed solutions that are *not* based on a central server. In a distributed solution, reasonably powerful processors are attached to the cameras. They communicate among themselves and perform all necessary computations. This scenario is quite realistic given contemporary technology, which further motivates the distributed approach.

While our algorithm is applicable to a centralized solution, it is easily adapted to a distributed implementation. Its modular structure makes it easily parallelizable, so that a distributed implementation among the computers attached to the different cameras yields significant savings in parallel time. Every camera is then involved in a limited amount of communication and performs only a *small* number of WBS computations. Propagation of the correspondence is performed by local communication between cameras. Nevertheless, we obtain the correspondence information as in the centralized case.

An important feature of the distributed setting is the fact that the system is not always reliable. Communication may be intermittent and processors and cameras can fail either because of malicious reasons or for administrative reasons such as maintenance or relocation. These issues are inherent in multi-camera systems whether the computation is performed centrally or in a distributed fashion. However, a distributed computation can provide much stronger robustness guarantees. We have already mentioned that a central server is a single point of failure. In this paper we show how our distributed algorithm can be tuned to tolerate communication failures and processor failures without losing correspondence information. 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. In order to overcome a failure rate of $f < 1$ of the communication channels (resp. a portion of $f < 1$ of the actually corresponding image points end up being false-negative errors of the WBS), an increase of roughly $\frac{1}{1-f}$ in the number of 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!

This paper makes four main contributions:

- We present a new algorithm for multi-image correspondence in a setting with a large set of cameras. This algorithm is based on the theory of random graphs, and it uses propagation of correspondence information to reduce the number of WBS computations needed.
- A *distributed* version of the algorithm, is described, which solves multi-image correspondence problem in a distributed system. This solution is efficient and robust against system failures, and it employs no central server and has no single point of failure.
- Techniques are developed for detecting and correcting false positive and false negative errors of the WBS computations. This plays an important role in improving the quality of the correspondence

information obtained from our algorithm. These techniques make essential use of the transitivity of correspondence to detect and later correct correspondence conflicts. Similar techniques may be applicable to handling false-positive and false-negative errors in the output of other correspondence algorithms.
- The algorithms and analysis presented are based on the fact that correspondence is an equivalence relation. They can equally well be applied to the computation of equivalence relations over data at different sites of a distributed system, given a (possibly noisy) component that finds matches over pairs of sites.

This paper is organized as follows. The next section surveys related work on correspondence and on distributed multi-camera applications. Section 3 presents the basic (centralized) **Match** algorithm for multi-camera correspondence and analyzes its properties. In Section 4 a distributed version of the algorithm, called **D-Match**, is presented and its robustness against crashes and communication failures is demonstrated. Section 5 presents an analysis of false-positive and false-negative errors of the WBS computations, and presents effective ways to detect them and overcome them. Section 6 presents simulation data illustrating our algorithm and failure detection scheme.

2 Previous work

There have recently been significant advances on the subject of Wide-baseline Stereo, in which computations involving a small number of images are used to extract correspondence information among the images [30, 2, 23, 26, 7, 18, 19].

Schaffalitzky & Zisserman [27] and then Ferrari *et. al.* [11] 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. Ferrari *et. al.* [11] propose a method for correcting errors of the WBS computations. Their method makes use of a similarity measure among corresponding points. In this paper we present a method that is not based on such a similarity measure. The method presented here is in some ways more general, as explained in Section 5. The question of how to reduce the number of WBS computations performed has been addressed by Schaffalitzky & Zisserman [27]. 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 WBS, resulting in run-time complexity of $O(n)$ WBS computations, where n is the number of cameras. Our approach is probabilistic, rather than heuristic. Moreover, its success is guaranteed with high probability for every given set of images.

Levi & Werman [14] 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, and can be regarded as propagation of 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.* [8, 9] 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 [9], was performed on a central server. The virtual-reality technology introduced by Kanade *et al.* [22, 25]

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, 15] 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.* [13] 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.

A multi-camera tracking application was designed as a distributed system by Cai & Aggarwal [6]. They presented a method for tracking humans across multiple synchronized video streams, using a module that predicts the position of the subject along a spatial-temporal domain, and then selects the camera that is best suited to continue tracking. In more recent studies, approaches to multi-camera tracking (e.g., [20, 6, 12]) were developed for a distributed setting. The tracking of objects is performed locally at each camera site, and some collaboration between cameras enables better tracking, or event detection. In all of these studies only a small number of cameras are used, and the cameras are assumed to be calibrated and reliable.

A self-configuring camera network was presented in [31], where a network of cameras observe pedestrian flow in a building. The system can auto-calibrate by learning the co-occurrences of motion events. Unfortunately, the computations are not done in a distributed fashion. Lately, the Cyclops project [24] aims at covering a large region with a network of hundreds of low-quality cameras, with the goal of observing the environment.

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 [28, 21, 16].

3 The Match algorithm

This section presents the basic scheme by which a randomized algorithm is used to solve the correspondence problem. It starts by defining the setting and specifying the problem. Consider a set of n images. Two image points (p, i) in image i and (q, j) in image j are said to *correspond*, denoted by $(p, i) \approx (q, j)$, if they are projections of the same 3D point in space. Informally, the task of a multi-image correspondence algorithm is to detect correspondence between as many image points as possible in the given set of images. Our approach to this problem will make use of a procedure, such as a Wide-based Stereo algorithm, that computes the correspondence among points in a pair of images. This procedure will be called a *WBS*.

A WBS algorithm accepts a pair of images and outputs a sparse set of pairs of matched image points. If a WBS computation outputs a match of two image points (p, i) and (q, j), then the points are said to be *directly matched*, which we denote by $(p, i) \approx_m (q, j)$.

A naive computation of corresponding points between the images in a multi-camera system performs $\binom{n}{2}$ WBS computations between all possible pairs of images. The **Match** algorithm presented below performs WBS computations between a strict subset of the possible pairs of images. It makes use

of the transitivity of correspondence to deduce that points in images that are not compared directly correspond. If there is a path of $\approx_m$ matches between (p,i) and (q,j) then we say that the two points are *transitively matched*. Assuming that the WBS computations only match (truly) corresponding points, transitively matched points are guaranteed to correspond, since correspondence is a transitive relation. Taking this into consideration, it suffices to perform enough WBS comparisons to ensure that all relevant correspondence information is obtained at least in the form of transitive matches.

Let V_Q denote the set of image points that are projections of the 3D point Q. An algorithm in the multi-image case *computes the full correspondence information* regarding Q, if it matches all pairs of points in V_Q (and matches none of the points of V_Q with points that are not in this set). Clearly, we cannot expect to obtain correspondence information regarding 3D points whose images are not matched by the WBS component that is used by the algorithm. A 3D point Q is *detectable* by a WBS computation if its projection is matched by the WBS in at least one pair of images.

Formally, we define the *multi-view correspondence* problem for a set of images to be the task of computing the full correspondence information regarding all (detectable) 3D points in these images. The **Match** algorithm will provide a good approximate solution to the multi-view correspondence problem.

The randomized **Match** algorithm for correspondence, accepts as input an exposure parameter k and a set of n images. It consists of three main steps:

Match algorithm**:**

1. **Initialization:** Randomly select each of the $\frac{n(n-1)}{2}$ pairs of images with an independent probability $\rho(k) = \frac{4.61+\log k}{k}$.
2. **Pairwise Comparison:** Perform WBS computations between each pair of images chosen in the first step. This provides direct matches among image points in the chosen pairs of images.
3. **Propagation:** Compute and output the transitive closure of the "direct match" relation obtained in the second step.

In general, the **Match** algorithm performs $\frac{4.61+\log k}{k}$ as many WBS comparisons as the naive algorithm does. Therefore in absence of information on the possible overlapping images, the number of WBS comparisons are $\frac{n(n-1)}{2}\frac{(4.61+\log k)}{k}$. However, when such information does exist in the system (e.g., via geographical considerations) the number of comparisons is reduced accordingly. For concrete values of k, **Match** provides genuine savings for all exposure parameters $k \geq 8$. For example, for $k = 8, 12, 16$ and 20 it saves 5%, 32%, 46% and 55%, of the WBS comparisons, respectively.

A probabilistic analysis can be used to prove bounds on the success of the **Match** algorithm in computing correspondence. We say that a WBS algorithm is *reliable* with respect to a set of images if it satisfies the following two conditions:

(i) No false-positive matches: If $(p,i) \approx_m (q,j)$ then $(p,i) \approx (q,j)$, meaning that only corresponding points are matched.

(ii) No false-negative matches: Assume that a 3D point Q is detectable and let $(p,i), (p',j) \in V_Q$ be projections of Q on two images i and j. Then, if a WBS computation is performed on the image pair $\{i,j\}$ a match $(p,i) \approx_m (p',j)$ is found.

With this notion, we shall soon prove the following theorem:

Theorem 1. *Assume a reliable WBS computation, and let* $8 \leq k \leq 40,000$. *Then, for every detectable point* Q *with exposure degree* k *or larger, the full correspondence information regarding* Q *is recovered by the* **Match** *algorithm with probability at least* 0.99.

The proof of Theorem 1 will be given below. Notice that Theorem 1 implies that the expected portion of points with exposure k or more for which full correspondence information is recovered is at least 99%. While the assumption of a reliable WBS computation may seem unrealistic, we show in Section 5 how to modify this algorithm to handle errors of the WBS computations.

Let us first focus on the matching information computed for a single 3D feature point Q. Recall that V_Q is the set of image points that are projections of Q. We define an undirected graph $G_Q = (V_Q, E_Q)$, where E_Q is the restriction of $\approx_m$ to V_Q. Thus, each edge in E_Q denotes a direct match found among two image points in V_Q. This graph represents the matching information regarding Q that is collected in the second step of the algorithm. The algorithm extends this matching information in the propagation step by computing the transitive closure of G_Q. Since G_Q is undirected, it follows that two points of V_Q are matched by the algorithm exactly if they are connected by a path in G_Q. Moreover, the algorithm will recover the full correspondence information regarding Q exactly if G_Q is connected. We will show that the choice of images in the first step makes G_Q be a random graph that is connected with high probabilty if Q's exposure degree is sufficiently large.

Assume that $(p,i),(q,j) \in V_Q$ where (p,i) is a point in image i, while (q,j) is a point in j. Our assumption that the WBS computations are reliable implies that (p,i) and (q,j) will be directly matched in the second step exactly if the pair of images $\{i,j\}$ is chosen in step one (and thus a WBS computation is performed on this pair in step two). Since a given pair of images is chosen with probability $\rho(k)$, we obtain that the probability that (p,i) and (q,j) will be matched is also $\rho(k)$. We can now conclude that the graph G_Q is a random graph, where E_Q is obtained by choosing every pair of nodes of V_Q with the probability $\rho(k)$.

It remains to show that if $|V_Q| \geq k$ then the graph G_Q is connected with high probability. To this end, we apply results from the theory of random graphs, which was initiated by Erdös and Rényi [10]. 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 $\mathcal{G}(N,\rho)$. Let $\hat{\rho}(N) = \frac{0.577+\ln N}{N}$. A classical result in the theory of random graphs shows that as N tends to ∞ the probability that $\mathcal{G}(N,\hat{\rho}(N))$ is connected tends to 1. This is instructive but is of little use for concrete and small values of N. Interestingly, Bollobás and Thomason [4] have proven useful bounds for concrete and relevant values of N (see the excellent textbook by Bollobás [3]):

Lemma 1. *Let* $\rho(N) = \frac{4.61+\log N}{N}$. *For every* $N \leq 40,000$ *and* $\sigma \geq \rho(N)$, *the probability that* $\mathcal{G}(N,\sigma)$ *is connected is greater than* 0.99.

The concrete bounds of Lemma 1 are relevant to practical applications. Indeed, we can use this lemma to formally prove Theorem 1:

Proof (Theorem 1). Assume that the algorithm is executed using exposure parameter k. For every feature point Q with exposure degree N the second step of the algorithm creates a random graph $G_Q = \mathcal{G}(N,\rho(k))$ in which every edge is chosen with probability $\rho(k) = \frac{4.61+\log k}{k}$. Lemma 1 guarantees that $G_Q = \mathcal{G}(N,\rho(N))$ is connected. By definition of ρ we have that if $N \geq k$ then $\rho(k) \geq \rho(N)$. Setting $\sigma = \rho(k)$ we have that $\sigma \geq \rho(N)$ and we obtain by Lemma 1 that G_Q is connected with probability at least 0.99. When G_Q is connected, the algorithm correctly computes correspondence for Q. □

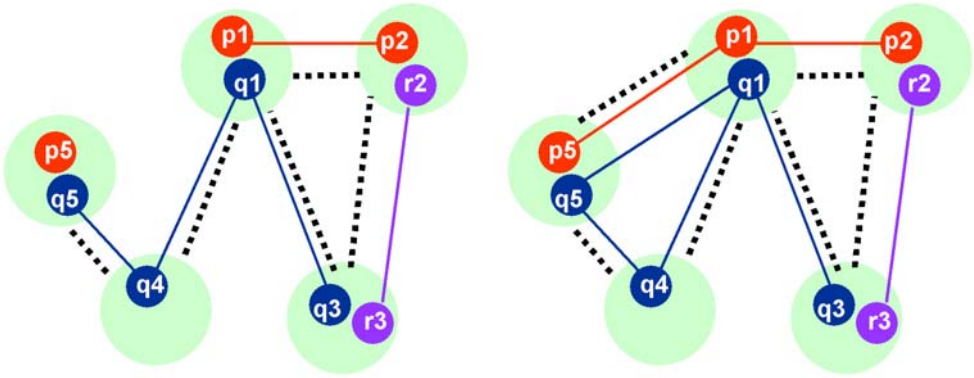


Fig. 1. Five cameras viewing a scene with 3D feature points P, Q and R. Dashed lines represent WBS computations and solid lines show direct matches. In (a) the WBS computations span the cameras but not the subset viewing P. In (b) the graphs of all points are connected.

Observe that two kinds of graphs have played a role so far. The first step of the algorithm can be considered as constructing a random graph whose nodes are images, where an edge connects a pair of images on which WBS computations are performed. Let us denote this graph by W (standing for WBS). Our analysis did not make direct use of W. Rather, for each detectable point Q, we used a graph G_Q whose nodes are image points and whose edges represent direct matches produced by the WBS computations. It would have been easy to ensure the connectivity of the graph W using a simple deterministic algorithm, such as one computing a ring or a spanning tree over the images. However, to compute correspondence for all points Q of sufficient exposure degree, we need each of the graphs G_Q to be connected. Interestingly, the connectedness of W does not ensure connectedness of all the graphs G_Q. An example demonstrating this is given in Figure 1. While the graph W of WBS computations among images in Figure 1(a) is connected, the graph $G_P = (V_P, E_P)$, where $V_P = \{p_1, p_2, p_5\}$ is disconnected. Since we do not know in advance the set of images that contain the projections of a given 3D point Q, there is no deterministic way to guarantee connectedness of G_Q without performing a large number of WBS computations. The **Match** algorithm, on the other hand, does not require prior knowledge about the images that contain projections of Q. By choosing the edges of W with a sufficiently high probability, the **Match** algorithm is guaranteed to produce a connected graph G_Q with high probability. Moreover, it does so *at once* for all points Q with exposure degree k or more.

4 The distributed algorithm

The **Match** algorithm presented in the previous section is easily implementable on a central computer once this computer obtains the image data from all cameras. However, such use of a central server has many disadvantages. One is the problem of scalability of the computational task as the number of cameras grows. Another issue is that a central server can become a communication hot-spot and possible bottleneck. Finally, the central server is a single point of failure. If it crashes or is unreachable

for a while, the applications it governs may fail. The disadvantages of the central server motivate distributed solutions that are *not* based on a central server.

We next suggest a distributed version of our algorithm. The setting assumed for the distributed solution is 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. The algorithm's goal in this case is to provide each camera with the complete correspondence information for every feature point in its image. The distributed version has two main advantages over the centralized one. The first is that by using n computers rather than one, the overall computation time is considerably reduced. As a result, the system is easily scalable to a large number of cameras. The second is that the distributed algorithm can be tuned to handle communication and cameras failures. As a result, the multi-camera system becomes a robust fault-tolerant system. In particular, no single failure (indeed no small set of failures) can impact the quality of the correspondence information obtained in a significant way.

The distributed algorithm is a straightforward implementation of the randomized **Match** algorithm described in Section 3. The choice of images to compare by WBS in the Initialization step and the WBS computations in the Pairwise Comparison step are performed in a distributed fashion. Both of these require local computations and communication between camera pairs whose images are being compared. The transitive closure computation is performed using $\log_2 k$ rounds of communication to propagate correspondence information.

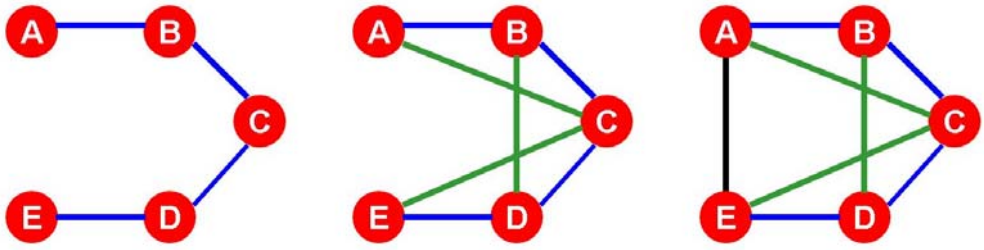
More formally, given an exposure parameter k and a set of cameras n, each camera i executes the following algorithm.

D-Match algorithm**:**

1. **Initialization:** Randomly choose a subset S_i of cameras of size $|S_i| = n\,\tau(k)$ and request their images, where $\tau(k) = 1 - \sqrt{1-\rho(k)}$ and the value of $\rho(k)$ is taken from the **Match** algorithm. (Hence $\tau(k) \cong \rho(k)/2$.)
2. **Pairwise Comparison:** For every $j \in S_i$, perform a WBS computation between image i and image j, store the results and send them to j.[5]
3. **Propagation:** This stage proceeds in rounds of communication. Its role is to propagate the matching information to all sites (i.e., to compute the transitive closure of $\approx_m$). Denote by $[p,i](m)$ the set of points that are recorded at camera i as (directly or transitively) matching (p,i) at the end of propagation round m. The set $[p,i](0)$ is initialized to be the set of image points that are found to directly match (p,i) by the WBS computations performed in the second (pairwise comparison) step of the algorithm. In every round $m \geq 1$ of the propagation step, if $m = 1$ or $[p,i](m-1) \neq [p,i](m-2)$ then camera i updates all members of $[p,i](m-1)$ regarding newly found matches in $[p,i](m-1)$. The propagation phase terminates at i when no new matching information is received.

Notice that choosing a subset of the neighbors of a predetermined size in the initialization stage has the advantage that we can control the number of WBS computations that every node performs. We will analyze the propagation phase of this algorithm below and show that it terminates in at most

[5] Clearly, if both i and j chose each other in the Initialization phase, then they can detect this, and only one of them performs the WBS computation.



a **b** **c**

Fig. 2. (**a**) In G_Q the distance between A and E is 4. (**b**) After one step of propagation the distance is reduced to 2, and (**c**) after two steps the distance is 1.

$\log n$ steps, and is expected to terminate faster. First, we state the analogue of Theorem 1. We say that the correspondence information for a 3D point Q has been fully recovered in the distributed setting if every camera containing a point in V_Q obtains the list of all points in V_Q.

Theorem 2. *Assume a reliable WBS computation, and let* $8 \leq k \leq 40,000$. *Then, for every detectable point with exposure degree* k *or larger, the full correspondence information is recovered by the* **D-Match** *algorithm with probability at least* 0.99.

Proof. Let Q be a detectable point with exposure degree k or more. If G_Q is connected, then every camera containing a node of V_Q will obtain the list of members of V_Q within a finite number of rounds of the propagation phase. It suffices to show that an appropriate proportion of the graphs G_Q are connected. Following the proof of Theorem 1, it is sufficient to show that in the distributed version of the algorithm, each edge is chosen with probability of at least $\rho(k)$. In the distributed algorithm, each camera randomly chooses a subset S_i of size $\tau(k) = 1 - \sqrt{1-\rho(k)}$. An edge is chosen if one of its nodes selects it. Since the nodes' choices are independent events, this guarantees that each edge is chosen with probability $2\tau(k) - \tau^2(k)$. The value of $\tau(k)$ was chosen to be the solution of the quadratic equation $\rho(k) = 2\tau(k) - \tau^2(k)$, and hence each edge is chosen with probability $\rho(k)$ as desired.

Since $\tau(k)$ tends to $\frac{\rho(k)}{2}$ in the limit and $\tau(k) < 0.6\rho(k)$ for all $k \geq 10$, the value of $\tau(k)$ is approximately $\frac{\rho(k)}{2}$. □

Observe that in every propagation step, any two nodes u and v that are connected to the same node w become directly connected (see Figure 2). As a result, such a step cuts the diameter of the graph by half. It follows that if the diameter of G_Q is d, then it will become fully connected within $\lceil log_2(d) \rceil$ steps. In fact, even if the exposure degree of a point Q is greater than k, the fact that we are choosing edges with probability $\rho(k)$ implies that most subsets of V_Q of size k are connected, and hence the diameter of G_Q will typically be less than k and so propagation will terminate in $\log k$ rounds or less in practice.

4.1 Communication failures and crashes

The implementations we have considered so far were shown to work well provided the multi-camera system and the software are reliable. No errors, crashes or failures were considered possible. In reality, we should expect both hardware and software problems to occur. A multi-camera setting is in particular a distributed system and it is therefore prone to camera and processor crashes as well as to communication failures. If the centralized solution is implemented, then a crash of the central server can cause complete failure of the application. As we now show, the distributed implementation of Section 4 is much more robust, and can be tuned to tolerate failures at a fairly small cost. False positive and false negative failures of the WBS software will be considered in Section 5.

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 camera i (or its processor) 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 Q to be its exposure degree if we ignore image points in cameras that crash. The original algorithm, unchanged, is guaranteed to discover all information for points with a surviving degree of k or more, since crashed cameras do not participate in the algorithm and therefore do not affect the interactions among the surviving cameras. This is as much as can be expected, because there is no way to extract image data from crashed cameras. If a crashed camera's image reaches a surviving camera then this can be used to improve the output and generate transitively computed matches faster than otherwise. Observe that because the application is computed in a distributed fashion, the failure of a single camera or processor has limited impact on the overall outcome.

Somewhat more interesting is the situation with regards to communication failures. Suppose 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. Assume that we choose edges with a probability $\rho'(k)$. Since communication line failures are independent of the choices of WBS computations made by the camera, the probability that an edge is both chosen and does not fail is $(1-f)\rho'(k)$. Hence, if we define $\rho'(k) = \frac{\rho(k)}{1-f}$ then each edge will be useful with probability $\rho(k)$. Since we are running a distributed version of the algorithm, every camera should select edges with probability $\tau'(k) = 1 - \sqrt{1-\rho'(k)}$ to match this value. Overcoming 10% failures requires 20% overhead for $k = 10$ and drops to 14.5% at $k = 15$. Moreover, to overcome a huge 25% probability of failure in the case of $k = 12$ it suffices to choose at most 60% more cameras than in the fully reliable case, while 48% suffice for $k = 15$. As we show in the experimental results, in practice significantly lower numbers suffice. This discussion is summarized as follows.

Theorem 3.

(a) Assume a reliable WBS computation in a system with camera crashes, and let $8 \leq t \leq 40,000$. *Let the* **D-Match** *algorithm be executed unchanged with* $\tau(t)$ *cameras chosen in the Initialization step. Then, for every detectable point with surviving degree* t *or larger, the full correspondence information is recovered by the* **D-Match** *algorithm with probability at least* 0.99.

(b) Assume a reliable WBS computation in a system in which communication channels fail with probability $f < 1$, *and let* $8 \leq k \leq 40,000$. *Let the* **D-Match** *algorithm be executed with* $\tau'(k) = 1 - \sqrt{1 - \frac{\rho(k)}{1-f}}$ *instead of* $\tau(k)$ *cameras chosen in the Initialization step. Then, for every de-*

tectable point with exposure degree k or larger, the full correspondence information is recovered by the **D-Match** *algorithm with probability at least* 0.99.

5 Handling Errors of the WBS Computations

So far we have been assuming that the WBS computations were reliable. In practice, this assumption is unrealistic. In this section we relax the reliability assumption and consider both positive and negative failures of the WBS computations. Considerations based on epipolar geometry and other well-known computer vision techniques can provide insight into such failures. However, we focus here on the failure information inherent in the graph describing the matches produced by the WBS computations. Our objective is to detect, locate and correct failures. Detecting means identifying an inconsistency in the graph. Localization means identifying a small set of edges that contains a faulty edge, and correction means removing a faulty edge.

A WBS computation can err in two ways: It can produce a *false-positive* match, in which it matches two images points (p,i) and (q,j) that do not in fact correspond (so $(p,i) \approx_m (q,j)$ but $(p,i) \not\approx (q,j)$), or it can produce a *false-negative* match, in which it fails to match two detectable feature points that are corresponding (so $(p,i) \not\approx_m (q,j)$ while $(p,i) \approx (q,j)$). In the context of the **Match** algorithm, the impact of a false-negative error is limited, since the match that is missed may still be obtained as a transitive match. The impact of a false-positive, however, is amplified by the use of propagation.

The **Match** algorithm makes use of the positive matching information obtained by WBS computations, by computing the transitive matches between points based on the direct match information obtained from the WBS computation. For the analysis of the WBS failures we will also consider the negative matching information available from our algorithm. We say that two image points, (p,i) and (q,j), are *negatively matched* if a WBS is performed on images i and j and it does not output this pair as a match. Most of the negative matches are expected to be correct, since many image points indeed do not match. However, a negative match can also be false-negative if (p,i) and (q,j) are indeed corresponding points.

A false-positive error will match an image point from a set V_Q of the projections of a 3D point Q with one from V_P for $P \neq Q$. To represent this fact, it is necessary to go beyond the graphs G_P and G_Q used to analyze the reliable case. When the WBS computations are error-prone, we represent the matching information they generate by a graph $\mathcal{G} = (V, E^+, E^-)$ defined as follows. The graph nodes V are image points, and $\mathcal{G}$ contains two disjoint sets of edges—E^+ representing direct matches and E^- representing negative matches. Formally,

$$\begin{aligned} V &= \{(p,i) |\ \ (p,i) \text{ appears in at least one direct match}\} \\ E^+ &= \{\{(p,i),(q,j)\} |\ \ (p,i) \approx_m (q,j)\} \\ E^- &= \{\{(p,i),(q,j)\} |\ \ (p,i) \text{ and } (q,j) \text{ are negatively matched}\} \end{aligned}$$

Observe that E^+ is the relation $\approx_m$. The set E^- is rather large, and it contains almost all pairs of image points in images that are compared by a WBS computation. However, we shall never need to construct E^- or keep it in memory, since it is easily computable from the much smaller E^+ and the graph W of pairs of images on which WBS comparisons were performed. E^- is used to simplify our exposition in the analysis of false matches.

Recall that in the absence of false-positive WBS failures, a path of positive edges connecting two points in $\mathcal{G}$ implies that the points correspond. In the absence of false-negative failures, a negative edge implies that its nodes do *not* correspond. We now consider how failures of the WBS can be overcome.

5.1 Overcoming false-negative failures

The effect of a false negative error is similar to that of a failed communication line: It keeps us from observing a match between two nodes in V_Q, and can therefore cause the graph G_Q to remain disconnected. We say that a WBS algorithm *fails to identify a match* between (p, i) and (q, j) if the two points correspond but the WBS component does not produce them as a match. The same arguments as in Theorem 3 can therefore be used to prove the following analogous theorem:

Theorem 4. *Assume that WBS computations may fail to identify a match with independent probability $f_N < 1$ and let $8 \leq k \leq 40,000$. Let the* **Match** *(resp.* **D-Match***) algorithm be executed using in the Initialization step $\rho''(k, f_N) = \frac{\rho(k)}{1-f_N}$ instead of $\rho(k)$ (resp. $\tau''(k, f_N) = 1 - \sqrt{1-\rho''(k)}$). Then, for every detectable point with exposure degree k or larger, the full correspondence information is recovered with probability at least* 0.99*.*

It follows that with high probability, despite negative failures of the WBS, a transitive match between any two corresponding points will be found by our algorithm.

5.2 Overcoming false-positive matches

When WBS computations are not fully reliable, they will typically give rise to both false-negative and false-positive errors. False-positive errors appear to be especially harmful in the **Match** algorithm, because correspondence information is extended by propagation. Consequently, a false-positive error between image points p and q will give rise to new false-positive errors between all points connected to p and all points connected to q in $\mathcal{G}$. As a result, false-positive errors require careful examination and treatment in the **Match** algorithm, or else the correspondence information it produces will have low quality. In this section we develop an approach to dealing with the case that both false-positive and false-negative errors arise from WBS computations. As we show in Section 6, after error correction, the quantity of false matches after propagation can typically be reduced to be significantly smaller than the probability of false-positive errors that the WBS component guarantees. Indeed, the error correction scheme presented here should in principle also be usable to improve the quality of correspondence information obtained without propagation, when all image pairs are compared.

Detecting matching conflicts We first consider how to detect the existence of failures. In **Match**, correspondence between two image points can be infered if the points are connected via a path of direct matches (E^+ edges) in $\mathcal{G}$. Often, more than one such path can exist between a given pair of image points. When WBS computations may produce false-positive and/or false-negative errors, information from different sources can be inconsistent. It will be convenient to call a path of positive edges containing a false-positive error a *faulty path*.

There are two straightforward indications for the occurrence of a WBS error (see Figure 3):

Local conflict: This occurs when two distinct points (p, i) and (p', i) in the same image are transitively matched (see Figure 3(a)). Since two points in the same image cannot be projections of the same 3D point, the path connecting them must be a faulty path, indicating the existence of a false-positive error.

Mismatch cycle: In this case, two points in different images are transitively matched, while a WBS computation comparing their images fails to produce them as a direct match (see Figure 3(b)). Formally, a *mismatch cycle* is a cycle in $\mathcal{G}$ consisting of a path of positive edges and a *single* negative edge. At least one edge of such a cycle must originate from a WBS error.

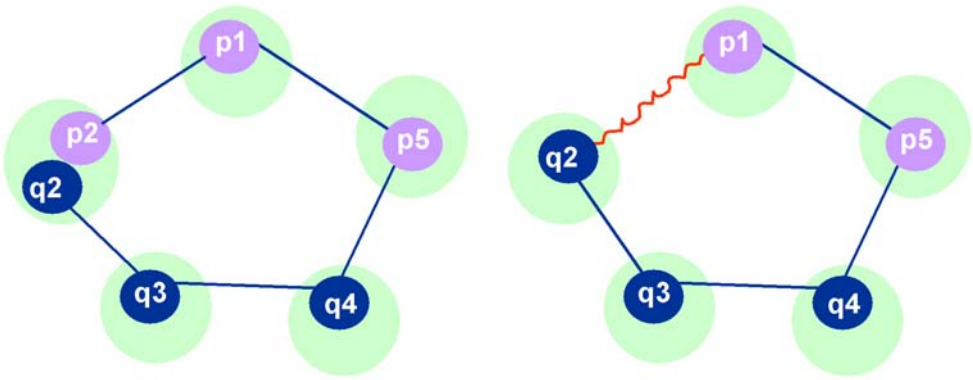


Fig. 3. A local conflict (a) and a mismatch cycle (b).

We will show that matching conflicts completely characterize the possibility of detecting the existence of failures based on the matching information. We first show that the existence of a local conflict implies that a mismatch cycle exists.

Lemma 2. *If $\mathcal{G}$ contains a local conflict, then there is a mismatch cycle in $\mathcal{G}$.*

Proof. Assume that two points (p,i) and (p',i) in the same image are connected by a path of positive edges in $\mathcal{G}$. Let $\{(q,j),(p',i))\}$ be the last edge on the path of positive edges from (p,i) to (p',i). Then a WBS was performed on the image pair $\{i,j\}$ and it found a direct match between (p',i) and (q,j). Since this computation can not match (q,j) with two distinct points in the image i, it follows that (q,j) was not matched with (p,i) and we obtain that $\{(p,i),(q,j)\} \in E^-$. This negative edge and the path of positive edges connecting (p,i) to (q,j) together form a mismatch cycle. □

The notion of mismatch cycles can be used to characterize the detectability of failures as follows.

Lemma 3. *(a) Every mismatch cycle contains at least one false edge.*
(b) If there is no mismatch cycle in $\mathcal{G}$, then $\mathcal{G}$ is compatible with an equivalence relation $\approx'$, and it is impossible to detect the existence of a false edge based solely on the information in $\mathcal{G}$.

Proof. (a) Assume that the negative edge in a mismatch cycle connects the points (p,i) and (q,j). If one of the positive edges in the cycle is a false positive, we are done. Otherwise, a straightforward induction on the length of the path of positive edges connecting these points shows that $(p,i) \approx (q,j)$. In this case, the negative match computed between (p,i) and (q,j) must be a false-negative.

(b) Assume that there is no mismatch cycle in $\mathcal{G} = (V, E^+, E^-)$. Define the relation $\approx'$ over the nodes of $\mathcal{G}$ so that $(p,i) \approx' (q,j)$ if the two points are E^+-connected in $\mathcal{G}$. The relation $\approx'$ is clearly an equivalence relation. We claim that if $\{(p,i),(q,j)\} \in E^-$ then $(p,i) \not\approx' (q,j)$. Suppose that this is not the case. Then $\{(p,i),(q,j)\} \in E^-$ for two points where $(p,i) \approx' (q,j)$. By definition of $\approx'$, this implies that there is a path of positive edges connecting (p,i) and (q,j). The negative edge between them completes a mismatch cycle in $\mathcal{G}$, contradicting the assumption that there is no mismatch cycle in $\mathcal{G}$. □

Lemma 3 provides a complete characterization of when it is possible to detect the existence of a faulty edge in $\mathcal{G}$. Part (b) of the lemma states that in the absence of a mismatch cycle, the matching information in $\mathcal{G}$ is consistent with there being no faulty edge in $\mathcal{G}$. Hence, the existence of a mismatch cycle is necessary for the existence of faulty edges to be detectable. Recall that the existence of mismatch cycles in the graph is computable, and hence detectable. By Part (a) we have that a mismatch cycle must contain a faulty edge, and thus such a cycle implies that a faulty edge exists. It follows that it is possible to *detect* the existence of a faulty edge based on the structure of $\mathcal{G}$ if and only if there is a mismatch cycle in the graph.

We remark that Ferrari *et. al.* [11] propose an error-detection method that is based on identifying local conflicts and eliminating them. They do not use negative matches to deduce inconsistencies in the data computed by the wide balseline stereo computations. As mismatch cycles are strictly more general than local conflicts, Lemma 3 shows that some detectable errors will go unnoticed (and will not be fixed) by the method proposed there.

Notice that a given false-positive edge need not necessarily participate in mismatch cycles. It can, for example, connect an isolated point in $\mathcal{G}$, or can be cancelled in a cycle by other false-positive edges. However, in many cases a false-positive edge will give rise to a mismatch cycle. To see why, consider a false positive edge $e = \{(p,i),(q,j)\}$. Suppose that (p,i) is a projection of the 3D point P and (q,j) is the projection of Q. If V'_P is the subset of V_P connected to (p,i) in $\mathcal{G}$ and V'_Q the analogous subset for (q,j), then any (correct) negative match between points in the two sets V'_P and V'_Q is guaranteed to close a mismatch cycle containing e. The number of pairs of points from these two sets is $|V'_P| \cdot |V'_Q|$. At least $|V'_P| \cdot (|V'_Q| - 1)$ of them are pairs of points that reside in different images. Since our algorithm performs a WBS computation on a pair of images with probability $\rho(k)$, it follows that the expected number of edges between points of the two sets is at least $t = \rho(k) \cdot |V'_P| \cdot (|V'_Q| - 1)$. For the false positive edge e not to be on a mismatch cycle, all of these edges must be also false-positives. Assume that the probability of a false-positive edge among image points of V_P and V_Q, given that there is a false-positive edge among points of these sets, is π. It follows that if these are independent events then the probability that e does not appear on any mismatch cycle at all is smaller than π^t. Recall that if the exposure degree of P (resp. Q) is $m \geq k$, then with high probability $V'_P = V_P$ and therefore $|V'_P| = m$. In this case, $t > \log k |V'_Q|$.

We can conclude that a false positive error of the WBS connecting points of sufficiently large exposure degree is extremely likely to cause a mismatch cycle. Theorem 4 implies that a false-negative error between image points in a set V_P where the 3D point P has exposure degree at least k will also generate a mismatch cycle with high probability, since V_P is likely to be connected.

Locating and Correcting matching conflicts We have seen that mismatch cycles are effective indicators for the existence of failures of the WBS computations. As we show below, detecting a mismatch cycle is fairly straightforward and efficient. Locating which of the edges in the cycle is the faulty edge will allow us to remove it and correct the matching. In principle, however, any edge in a mismatch cycle might be the faulty edge. Furthermore, as long as both positive and negative failures can occur, it is in general impossible to uniquely determine a faulty edge based on the graph alone. We therefore suggest a method for *shrinking* a mismatch cycle to a *mismatch triangle*—a mismatch cycle of length 3—thereby reducing the number of candidate faulty edges in a mismatch cycle to at most three. Finally, to determine which of the three edges is faulty, we present a heuristic algorithm.

Our method is based on performing additional tests of correspondence on pairs of image points. We call such tests *probes*. A probe can be obtained either by applying a new WBS computation between the two images or by other more specific vision techniques that can verify whether two chosen points

correspond. The success of a probe is considered to be another positive match, and a negative outcome of the probe is considered a negative match. We do not assume here that probes are more reliable than the WBS computations were, although for particular pairs it may be feasible to increase the chances of a correct answer in such cases. Additional methods for correcting errors may be based on direct vision techniques (e.g., directly comparing local image patches or using three rather than two images), but here we study error correction based only on the information available from the matching information in the graph $\mathcal{G}$.

The general structure of a failure correction algorithm is as follows:

Failure correction algorithm:

While mismatch cycles exist do

1. **Compute a mismatch**
2. **Shrink the mismatch cycle** to length ≤ 3
3. **Remove faulty edge**
4. **Recompute transitive closure** as in the Propagation step of **Match**.

Steps one and two are about locating the edges that cause the mismatch conflict to occur, while step three performs error correction. For ease of exposition, we use the term *connected component* in the sequel as shorthand for E^+ *connected component*. We now describe the steps of this algorithm in greater detail.

Computing a mismatch cycle: Any negative edge among two points that are in the same connected component appears in a mismatch cycle. It is convenient to start constructing a mismatch cycle from such negative edges. Let $e = \{(p,i),(q,j)\} \in E^-$ be a negative edge among two transitively matched image points. By performing a breadth-first search on (V, E^+) starting from, say, (p,i), we can find a shortest path connecting it with (q,j). This path and the negative edge e form a mismatch cycle. This runs in time linear in the size of the connected component, and hence is extremely efficient. Observe that if two points (p,i) and (p',i') from the same image are transitively matched (forming a local conflict), then a similar breadth-first search (BFS) computation will find a faulty path of E^+ edges connecting these points.

Shrinking the mismatch cycle: Here we use probing on pairs of image points to shrink a mismatch cycle to a mismatch triangle. (Similarly, a faulty path will be reduced to a faulty path of length two or to a mismatch triangle). We call this procedure *binary probing*. Suppose that $(p_1,i_1),\ldots,(p_\ell,i_\ell)$ is a mismatch cycle, with the negative match being between (p_ℓ,i_ℓ) and (p_1,i_1). We repeat the following step as in Binary search until the cycle is a triangle:

Compare (p_1,i_1) `to the middle node in the cycle` $(p_{\ell/2},i_{\ell/2})$.
If `the comparison produces a` **negative match** `(Figure 4(b)) then`
$(p_1,i_1),\ldots,(p_{\ell/2},i_{\ell/2})$ `is a mismatch cycle.`
Otherwise, `the comparison produces a` **positive match** `(Figure 4(c)) and`
$(p_1,i_1),(p_{\ell/2},i_{\ell/2}),\ldots,(p_\ell,i_\ell)$ `is a mismatch cycle.`

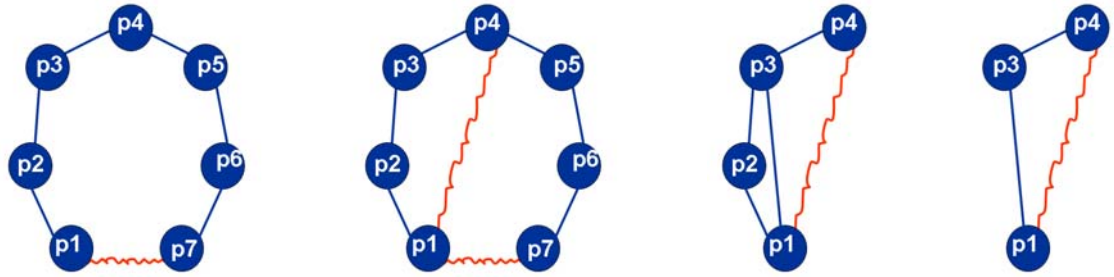


Fig. 4. Binary probing: (a) A mismatch cycle, (b) A probe yielding a negative match between p_1 and p_4 creates a shorter mismatch cycle, (p_1, p_2, p_3, p_4). (c) A probe between p_1 and p_3 yields a positive match and creates the mismatch triangle in (d).

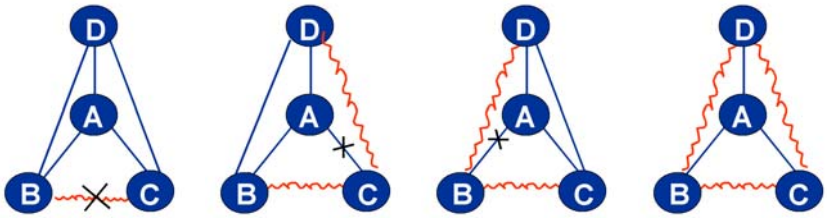


Fig. 5. Removing faulty edges: Given the mismatch triangle ABC, and the comparison results from these points to D. (a) Positive matches of BD and DC indicates that BC is a false negative. (b) A negative match of CD and a positive match BD indicate that CA is a false negative, and symmetrically (c) A Negative match BD and a positive match CD indicate that BA is a false negative. Finally, (d) if both BD and CD are negative matches, then there must be more than one failure among ABCD.

Shrinking a faulty path $(p_1, i_1), \ldots, (p_\ell, i_\ell)$ is similar. A particular case of binary probing is illustrated in Figure 4.

Correcting false edges: After shrinking, we have a mismatch triangle or a faulty path of length 2. At least one of the edges involved must be false. We proceed to perform some more probing and heuristically correct a single edge that is likely to be faulty. We describe how to do this for a mismatch triangle. The treatment of a faulty path of length 2 is analogous.

Suppose that ABC is a mismatch triangle, where BC is the negative edge. Choose an image point D that is directly matched to A and probe the pairs BD and CD. A likely faulty edge can be determined based on these probes, as shown in Figure 5. Three of the four possible outcomes are each consistent with a particular edge of ABC being false, while the fourth implies that at least two match results are false. In particular, if both probes produce a positive match (Figure 5(a)), then consider the negative match of BC to be false. If BD (resp. CD) yields a positive match and CD (resp. BD) a negative match,

then AC (resp. AB) is the candidate faulty edge. See Figure 5(b) (resp. (c)). Finally, if both probes yield negative matches, then both ABD and ACD are mismatch triangles (Figure 5(d)). Given that ABC is a mismatch triangle, it follows that between A, B, C, and D there are at least two false edges. In each of the first three outcomes, our failure correction algorithm will remove the corresponding edge from E^+ or E^-. In practice, E^+ will generally be explicitly represented, with E^- induced by E^+ and W. In this case, the removal of an edge from E^- is achieved by adding this edge to E^+. In the case of two failures, we can search for an alternative D node, which may be connected to any of A, B or C, and apply similar considerations. If all of these fail, we can discard all three edges involved. (Alternatively, additional probabilistic considerations can be developed for this case, but this is beyond the scope of this paper.)

Notice that the fault correction algorithm has one heuristic step—removing an edge from a mismatch triangle or a faulty path. If the probability of false matches is small, then the suggested scheme is likely to correctly identify a faulty edge and remove it. The order in which errors are corrected can be an important factor in whether error correction converges. Recal that our error-correction is typically based on a noisy WBS component. Since our error correction involves both adding matches to correct false-negative errors and removing errors to correct false-positives, it is, in principle, possible for error correction to encounter cycles leading to non-convergence. This can easily be handled by removing an edge that is repeatedly added and removed from the graph (beyond a threshold number of times). Similarly an image point that participates in too many mismatch-triangles can also be discarded. Since such cycles are rare, this ensures convergence without having a significant impact on the final results, as demonstrated in Section 6.

Given probabilities for false-positive and false-negative matches, we can define the *support* of an edge e of $\mathcal{G}$ to be the conditional probability that e is a false edge, given $\mathcal{G}$. The problem of computing the support of an edge appears to be nontrivial in general. Ideally, given a mismatch triangle (or even a larger mismatch cycle), it appears that a good candidate for a faulty edge to remove is the edge with lowest support in the cycle. The use of the node D in the last step of the fault correction algorithm can be viewed as a way of increasing the *support* of some match results in the triangle and reducing the support of others. In fact, there is a rich mathematical structure underlying this matter, whose analysis is beyond the scope of this paper.

Distributed implementation: Let us briefly discuss how fault correction can be implemented in the distributed setting. In this case, the existence of a local conflict is observed by the camera i whose image contains the local conflict at the end of the Propagation phase of **D-Match**, because i receives notice of all nodes in the connected component of its image points. As for a mismatch cycle, recall that the matching results of a WBS computation that is performed on images i and j are kept at cameras i and j in the Pairwise Comparison step of **D-Match**. If camera i is informed during Propagation that a point (q, j) is transitively matched to (p, i), and a WBS between i's and j's images produced a negative match, then i (and similarly j) can detect the existence of a mismatch cycle. It follows that cameras with points on a negative edge of a mismatch cycle are guaranteed to discover the existence of a mismatch cycle. In order to compute a mismatch cycle containing the negative edge $e = \{(p, i), (q, j)\}$ camera i can initiate a BFS computation among the cameras in (p, i)'s connected component. Another, often more efficient, possibility is to modify the contents of messages in the Propagation step so that they pass on topology information consisting of the E^+ edges. At the end of Propagation, each camera i would have a copy of the connected components of each of its feature points, and i then can compute a mismatch cycle containing $e = \{(p, i), (q, j)\}$ by a local BFS on i's copy of the connected component.

The modifications in the distributed setting required for handling local conflicts are analogous to those discussed for mismatch cycles.

6 Experiments

The experiments described in this section are divided into two parts. We first evaluate the performance of the **Match** algorithm with reliable WBS computations. This part demonstrates the effectiveness of propagation as a means to save WBS computations while still recovering the correspondence information. Our theoretical results showed that **Match** is guaranteed to recover an overwhelming portion of the correspondence information in the absence of errors or even in the presence of system failures. Our experimental results show that the theoretical analysis is conservative, and in practice smaller numbers of WBS computations suffice to recover a significant amount of the correspondence information. The second part of our experiments considers unreliable WBS computations, and focuses on the performance of our error correction techniques for overcoming false-positive and false-negative matches. The experiments demonstrate that the great majority of false-positive errors can be corrected based solely on the graph-theoretical considerations.

Since our approach is not based on any specific WBS tool, the experiments consist of extensive simulations in MATLAB on synthetic data. Our scenario is a surveillance system in an urban setting and thus our simulated testbed contains a collection of orthographic cameras that are mounted on rooftops looking down. Each camera observes all the feature points within a predefined 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.

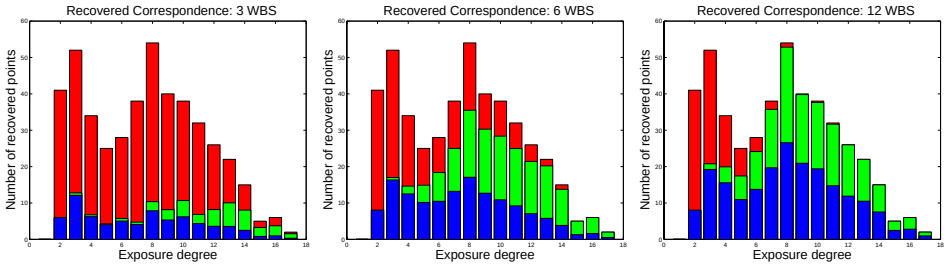


Fig. 6. The first experiment on a reliable system of 50 cameras and 500 points. For exposure degree d, the full height of the bar at d is the number of points with exposure degree d. The blue bar is the value of $T_E(d)$ after WBS computations are performed but before propagation. The green bar is the final value of $T_E(d)$ after propagation. Note how the exposure degree of the fully recovered points decreases as the number of WBS computations performed increases. (**a**) The results with three WBS per camera, (**b**) with six WBS per camera and (**c**) with twelve WBS per camera.

6.1 The Match algorithm with reliable WBS

The first experiment was designed to verify the theoretical bounds on the number of required WBS operations. We generated a setup of 50 cameras and 500 points (Figure 7(a)). In the first experiment, we

simulated the algorithm with precisely the same data, but with different numbers of WBS operations. The experiment shows that the predicted number of required 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 degree. Denote by $L_i(p)$ the size of camera i's correspondence list for p. A point p with exposure degree d is fully recovered if $L_i(p) = d$ for every camera i that views p. It follows that

$$\frac{1}{d^2} \sum L_i(p) = 1$$

if p is fully recovered, and this value is smaller than 1 if p is only partially recovered.

Let $E(d)$ be the set of feature points with exposure degree d. Ideally, if all the points in $E(d)$ are recovered, then $|E(d)|$ is equal to:

$$T_E(d) = \frac{1}{d^2} \sum_{p \in E(d)} \sum_i L_i(p)$$

We use the measure $T_E(d)$ to evaluate the success of extracting the correspondence. Intuitively, the ratio of $T_E(d)$ and $|E(d)|$ represents the average completeness of the local correspondence lists for points of exposure degree d.

For each exposure degree d, we observe three values (see Figure 6). The full height of the bar at d is $|E(d)|$—the number of points with exposure degree d. The bottom portion of the bar represents the value of $T_E(d)$ after direct matching of the WBS computations (before propagation). The next level up is the final value of $T_E(d)$ after propagation, representing the success of the algorithm in recovering correspondence information for the points of degree d.

As can be seen in Figure 6(a), using just three WBS per camera does not generate enough matching points and hence the algorithm does not fully recover any of the points. As the number of WBS performed grows, the number of fully recovered points grows. In Figures 6(b) and 6(c), we present the results of running performing six and twelve WBS computations per camera. As can be seen, the exposure degree from which full correspondence is obtained is reduced when the number WBS a camera performs increases. We note that by our theoretical bounds, for the case of $n = 50$ considered in the experiments, performing 3 WBS per camera guarantees 0.99 recovery of correspondence for points with exposure degree $k \geq 40$; performing 6 WBS per cameras recovers points with degree $k \geq 18$, and performing 12 WBS per cameras recovers points with exposure degree $k \geq 8$. The results depicted in Figure 6 are cumulative over all points, and show that useful correspondence information for practrical applications can be obtained using significantly fewer comparisons than suggested by the bounds of Theorem 1.

6.2 False negative errors

In the second experiment we ran the algorithm on five different randomly chosen data sets, each with 50 cameras and 500 points. The algorithm was run with five different values of number of WBS performed by each camera. We repeated the experiments for an unreliable system, in which each communication line has probability of 0.2 to fail, and the WBS has a probability of 0.2 to generate a false negative on each point. To evaluate the performance of the algorithm we measured the smallest degree for which all the sets of that degree and up were at least 90% recovered. In Figure 7(b) the green line shows the results for the reliable system, and the blue line the results for the unreliable system. As expected, the exposure degree recovered decreases as the number of WBS performed increases; interestingly, the algorithm performs in the unreliable system better than half as well as in the reliable one.

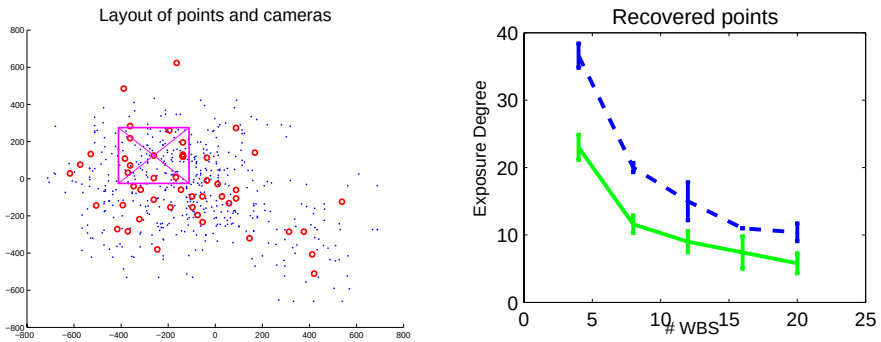


Fig. 7. (a) The setup with 50 cameras and 500 points. Each point is marked with a blue dot, and each camera center is marked with a red circle. A rectangle shows the field of view of one of the cameras. (b) Mean and standard deviation of five runs of the algorithm on 50 cameras and 500 points. The green line is for a reliable system, and blue line is for a system with probability 0.2 of communication failures and probability 0.2 of false-negative matches. The exposure degree recovered decreases as the number of WBS performed increases; the algorithm performs in the unreliable system better than half as well as in the reliable one.

6.3 Correcting WBS errors

Finally we consider a setting with false positive and false negative errors of the WBS. We ran the **Match** algorithm with faulty WBS computations, and employed the additional error-correction layer of Section 5.2 to the data collected. The error-correction computation detects local conflicts and mismatch-cycles, which are the indicators for WBS failures, and uses binary probing followed by a simple heuristic to correct these failures. We considered both negative (Fneg) and positive (Fpos) errors of the WBS computation. In the simulation, a WBS computation on a pair of images with false negative matches is obtained by first computing the correct matches between a pair of images, then removing an Fneg portion of the matches, and finally choosing from the remaining matches Fpos matches and randomly permuting their matches. At the end of this process, a pair of corresponding image points in the respective images compared by the WBS will form a false-negative error with probability $\mathsf{Fpos}+\mathsf{Fneg}-(\mathsf{Fpos}\cdot\mathsf{Fneg})$. (The total number of errors is thus Fpos false-positives and $\mathsf{Fpos}+\mathsf{Fneg}-(\mathsf{Fpos}\cdot\mathsf{Fneg})$ false negatives.) The error correction steps in the simulations are performed with simulated WBS computations that have the same error rates as those used to produce the original data. The simulation was run on a setup consisting of 50 cameras and 500 points (similar to Figure 7(a)).

We performed two sets of experiments. The first set studies the performance of the algorithm with error correction as a function of the number of WBS computations performed per camera, while the second does so as a function of the level of errors in the WBS computations. In each case, we compare the quality of correspondence information produced using propagation before and after error correction. For a given pair of images, we first consider the ratio of false-positive matches among points of these images to the total number of matches among the same pair. The average of this value, over all pairs of cameras whose images overlap is denoted by FP. A second measure, denoted TP, is also an average over all image pairs. In this case, the value for a pair of images is the ratio of the true positive matches recovered to the number of pairs of corresponding points the images share in the ground truth data.

Note that the sum of FP and TP is not necessarily 1. This sum may be considerably greater than 1 when many errors occur.

The behavior of error correction as a function of the number of WBS computations performed by each camera is depicted in Figure 8. The experiments in this graph were performed with $\mathsf{Fneg} = \mathsf{Fpos} = 0.15$, and the average exposure degree of the 3D points in this experiment is 6.5. The red dashed line in Fig 8(a) depicts the value of FP before error correction is applied and demonstrates the blowup in false-positive matches caused by propagation. The blue line depicts the value of FP after error correction. As the graphs show, FP is reduced significantly as the number of WBS performed increases. The value of FP falls below the original Fpos ratio of 0.15 of the WBS component already when 4 WBS computations are performed per image. It continues to drop rapidly and reaches below 0.03 for 10 WBS computations. Recall that our heuristic error-correction scheme may sometimes incorrectly remove true positive matches. Indeed, the elimination of false positive errors comes at a cost of reducing the number of true matches that survive error correction. In Fig. 8(b), the red dashed line shows the value of TP before error correction. This is an upper bound on the amount of data that can be recovered by the algorithm after error correction. The blue line shows the value of TP after error correction. For 8 WBS computations, 62% of the ground truth matches are recovered, where 72% is the upper bound. Note that exposure degree of 6.5 is below the thresholds considered by the theoretical results in earlier sections of the paper. The results when cameras have a broader field of view and the average exposure degree was 9.8 were qualitatively similar, and quantitatively quite a bit better.

The next set of experiments performed tested the performance of our error correction technique. We simulated the use of WBS computations (chosen ten per camera) with failure probabilities of $\mathsf{Fneg} = \mathsf{Fpos} = 0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3$, and 0.4. The average exposure degree in this experiment was 9.6. The results are depicted in Figure 9. As expected, the performance is highly correlated with the quality of the WBS component used. Nevertheless, even when $\mathsf{Fpos} = \mathsf{Fneg} = 0.25$, which means that 25% of the direct matches produced are false-positives, and 44% of the matches among the cameras tested are false-negatives, the final outcome provides 49% of the overall true matches existing among the cameras. Moreover, we are left with only 3.7% false-positive errors after propagation. This quality of output is sufficient for running bundle adjustment, for example. We checked the actual numbers (before propagation) for this extreme case. The total number of ground-truth matches in the scene was 54,000. After the initial WBS computations, we had 13,600 true-positive matches and 4,340 false-positives. After error correction, we were left with 11,200 true-positives and 424 false-positives. Thus, close to 4,000 (over 90%!) false-positive errors were eliminated, at the cost of roughly 2,400 (less than 20%) of true positives. Overall, this performance suggests that with the error correction, the **Match** algorithm becomes useful in practical contexts. After the $\mathsf{Fpos} = \mathsf{Fneg} = 0.25$ point, the performance continues to degrade linearly, and the results are no longer of any use when $\mathsf{Fpos} = \mathsf{Fneg} = 0.4$. We can see that the level of false-positive errors in this case matches the quality of the original WBS component (FP is 40%). But the portion of recovered true matches seems too small to be of practical use (TP is 14%). In this case, of course, there is limited data to begin with, and both this data and the WBS comparisons used in the error-correction phase are of low quality. Observe that in Figure 9(b), the value of TP does not diminish significantly as the number of failures increases. This happens because false positive matches increase the connectivity of $\mathcal{G}$. As a result, true positive matches that are not directly matched by the WBS may be transitively matched. Indeed, as FP approaches 1, the false positive matches may make $\mathcal{G}$ consist of a single connected component, in which case TP will be equal to 1.

The experimental results suggest that the **Match** algorithm offers good performance well beyond the bounds suggested by our theoretical analysis. Moreover, the error-correction heuristic makes this approach robust against significant degrees of error by the WBS comparison module used. This can, for

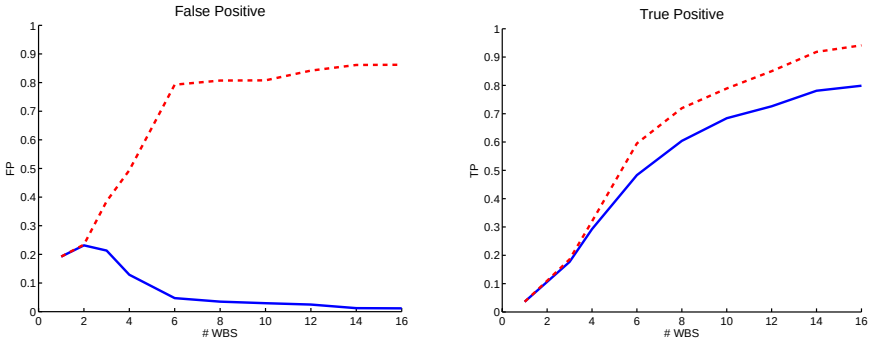


Fig. 8. True and false positive matches as a function of WBS computations performed. (a) Average ratio of false positive matches to total number of matches among pairs of cameras after propagation: Before error correction (red dashed line) and after correction (blue line). (b) Average portion of true positive matches over ground truth matches among pairs of cameras after propagation: Before error correction (red dashed line) and after correction (blue line). Tested for 50 cameras, 500 points, exposure degree 6.5 and $\mathsf{Fpos} = \mathsf{Fneg} = 0.15$.

example, have a dramatic effect on outlier rejection schemes, such as RANSAC [17], that are often used in computer vision. The RANSAC equation states that robustly fitting a model to the data requires L trials, where

$$L = \frac{log(p_{fail})}{log(1 - p_{good}^N)} \quad ,$$

where N is the number of points needed to fit a model ($N = 8$ in case we want to fit the fundamental matrix to the data), $p_{good} = 1 - \mathsf{Fpos}$ is the fraction of true positive matches and p_{fail} is a user defined parameter that specifies the probability that the algorithm will exit without finding a good fit to the model. Using the values of figure 8(a) we have that if $p_{fail} = 0.001$ and $\#WBS = 6$, then $p_{good} = 0.2$ before correction and therefore $L = 3,597,784$. After running our error correction procedure $p_{good} = 0.99$ and correspondingly L drops to 4.

These experiments cover only a small fraction of the possible questions one may raise with respect to this error-correction scheme. An extensive set of experiments is beyond the scope of this paper, and we believe that the experiments presented here indicate that the method has merit. In addition, it justifies the use of the **Match** algorithm even with unreliable WBS components and for settings with modest exposure degrees.

7 Discussion and Summary

This paper has presented algorithms for the multi-view point correspondence problem with the following features:

1. The algorithms are based on a black box, WBS, which computes correspondence between pair of images.

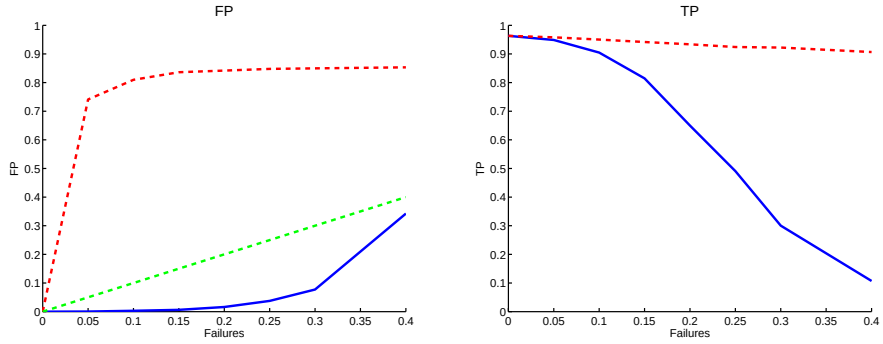


Fig. 9. True and false positive matches as a function of probability of WBS errors. All cases were tested with $\mathsf{Fpos} = \mathsf{Fneg}$, and this is the failure probability listed on the X-axis. (a) Average ratio of false positive matches to total number of matches among pairs of cameras after propagation: Before error correction (red dashed line) and after correction (blue line). The green line represents the Fpos probability of the WBS component used. (b) The portion of true positive matches after propagation: Before error correction (red dashed line) and after correction (blue line). For 50 cameras, 500 points, 10 WBS per camera and average exposure 9.6.

2. Using randomization and the transitivity of correspondence, the number of WBS computations among images is reduced, while still computing the correspondence information for all points with a chosen exposure degree. The improvement is by a constant factor in the centralized case, and the factor is a function of the exposure parameter k. In the distributed implementation, the original computation is performed in parallel by the n processors, and a further reduction by a factor of n in parallel time is obtained.
3. False-positive and false-negative failures of the WBS computations are handled efficiently. The quality of the correspondence information after error correction is sufficient for many applications including bundle adjustment.
4. Distributed implementations of the algorithms are given that further reduce the parallel computation time by an order of magnitude.
5. In the distributed implementation, camera and communication line failures are tolerated at a relatively small cost, yielding implementations that are robust and contain no single point of failure. In contrast, in a centralized solution to the multi-view correspondence problem, the central server is both a single point of failure and a computation bottleneck.

While the paper has been written with the goal of presenting effective solutions to the point correspondence problem, the results apply more broadly. Consider, for example, related correspondence problems in which the goal is to identify corresponding objects, regions, or events viewed by the different cameras. Since the algorithms in this paper make use of WBS computations as a black box, any similar computational procedure that would identify pairs of corresponding objects, regions, or events, respectively, among different cameras, can be substituted for the WBS component and yield an algorithm for the respective instance of correspondence problem. As long as the property considered is transitive, all of the technical development in this paper applies.

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 com-

munication medium is used for integrating the data being gathered. As the correspondence problem illustrates, the information in such a distributed vision system has particular properties. In contrast to, say, a distributed database, views of the same points, objects or events at different sites of the system are not labelled in a uniform manner. Rather, a nontrivial computation is needed in order to discover the correlation between the information at different sites. This has caused us to seek and develop new distributed algorithms to handle this type of problem. We expect further attempts to solve multi-camera problems in a distributed manner to give rise to additional properties worthy of study from a distributed systems point of view. We believe that the study of multi-camera applications as problems that bring together computer vision concerns with distributed systems promises to be interesting and worthwhile.

Acknowledgments

This research was supported in part by the Israel Science Foundation (grant No. 1339/05). The second author performed part of this research while on sabbatical at the School of Computer Science and Engineering, The University of New South Wales (UNSW), Sydney, Australia and NICTA.[6] The third author performed part of this research while on sabbatical at the School of Computer Science and Engineering, UNSW, Sydney Australia, supported in part by ARC Discovery Grant RM02036.

References

1. Avidan, S., Moses, Y., and Moses, Y. Probabilistic multi-view correspondence in a distributed setting with no central server. In *Proc. of European Conference of Computer Vision* (2004), pp. 428–441.
2. Baumberg, A. Reliable feature matching across widely separated views. In *Proc. of IEEE Computer Vision and Pattern Recognition* (2000), pp. I: 774–781.
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. Cai, Q., and Aggarwal, J. Automatic tracking of human motion in indoor scenes across multiple synchronized video streams. In *International Conference on Computer Vision* (1998), pp. 356–362.
7. Chetverikov, D., and Matas, J. Periodic textures as distinguished regions for wide-baseline stereo correspondence. In *Texture* (2002), pp. 25–30.
8. 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).
9. 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.
10. Erdos, P., and Renyi, A. On random graphs 1, 1959.
11. Ferrari, V., Tuytelaars, T., and Gool, L. V. Wide-baseline multiple-view correspondences. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* (June 2003).
12. Kang, J., Cohen, I., and Medioni, G. Continuous multi-views tracking using tensor voting. In *Motion02* (2002), pp. 181–186.
13. 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).

[6] National ITC Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

14. Levi, N., and Werman, M. The viewing graph. In *Proc. of IEEE Computer Vision and Pattern Recognition* (2003).
15. 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.
16. Lynch, N. A. *Distributed Algorithms.* MIT Press, 1996.
17. M. A. Fischler, R. C. B. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communication of the ACM* (1981), vol. 24, pp. 381–395.
18. Matas, J., Chum, O., Urban, M., and Pajdla, T. Robust wide baseline stereo from maximally stable extremal regions. In *The British Machine Vision Conference* (2002).
19. Mikolajczyk, K., and Schmid, C. Scale & affine invariant interest point detectors. *Int. J. Comput. Vision 60*, 1 (2004), 63–86.
20. Mittal, A., and Davis, L. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *IJCV 51*, 3 (February 2003), 189–203.
21. Mullender, S. *Distributed Systems.* Addison Wesley, 1993.
22. 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).
23. Pritchett, P., and Zisserman, A. Wide baseline stereo matching. In *Proc. International Conference on Computer Vision* (1998), pp. 754–760.
24. Rahimi, M., Estrin, D., Baer, R., Uyeno, H., and Warrior, J. Cyclops, image sensing and interpretation in wireless networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (2004), pp. 311–311.
25. 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).
26. 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.
27. 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).
28. Tanenbaum, A. S., and van Steen, M. *Distributed Systems Principles and Paradigms.* Pearson Education publisher, 2001.
29. Triggs, W., McLauchlan, P., Hartley, R., and Fitzgibbon, A. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice* (1999), pp. 298–372.
30. Tuytelaars, T., and Van Gool, L. Wide baseline stereo matching based on local, affinely invariant regions. In *The British Machine Vision Conference* (2000).
31. Wren, C., and Rao, S. Self-configuring, lightweight sensor networks for ubiquitous computing. In *Proc. of International Conference on Ubiquitous Computing (UbiComp)* (2003), pp. 205–206.