2010

An automatic calibration method for stereo-based 3D distributed smart camera networks

Aaron Mavrinac

Xiang Chen

Kemal Tepe
University of Windsor

Recommended Citation

Mavrinac, Aaron; Chen, Xiang; and Tepe, Kemal. (2010). An automatic calibration method for stereo-based 3D distributed smart camera networks. *Computer Vision and Image Understanding*, 114 (8), 952-962.
http://scholar.uwindsor.ca/electricalengpub/8

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

An automatic calibration method for stereo-based 3D distributed smart camera networks [☆]

Aaron Mavrinac, Xiang Chen [*], Kemal Tepe

University of Windsor, Department of Electrical and Computer Engineering, 401 Sunset Ave., Windsor, Ontario, Canada N9B 3P4

ARTICLE INFO

Article history:
Received 31 August 2009
Accepted 18 March 2010
Available online 27 March 2010

Keywords:
Camera network
Calibration
Collaborative
Distributed
Registration
3D vision
Vision graph
Collaborative vision
Distributed computing

ABSTRACT

Stereo-based 3D distributed smart camera networks are useful in a broad range of applications. Knowledge of the relative locations and orientations of nodes in the network is an essential prerequisite for true 3D sensing. A novel spatial calibration method for a network of pre-calibrated stereo smart cameras is presented, which obtains pose estimates suitable for collaborative 3D vision in a distributed fashion using two stages of registration on robust 3D point sets. The method is initially described in a geometrical sense, then presented in a practical implementation using existing vision and registration algorithms. Experiments using both software simulations and physical devices are designed and executed to demonstrate performance.

1. Introduction

Most distributed smart camera (DSC) network applications make use of three-dimensional scene information to some extent. This is not surprising; the fundamental purpose of any multi-camera system is to view more than is possible with a single camera, and this almost always implies taking advantage of space in the same sense that a video takes advantage of time. DSC networks may, for example, simply use multiple views to augment human pose analysis, or perform tracking between non-overlapping views; in more direct cases, they may track by triangulation among overlapping views, or even perform full scene reconstructions. True 3D sensing networks provide this information naturally, replacing the traditional basic primitive of 2D images with 3D scene structure. It is our belief that such networks offer potential improvements to a variety of existing applications and open entirely new possibilities for exploration.

The bulk of research in DSC networks to date has focused on the fusion of data from multiple monocular camera nodes. While a number of distributed partial and full 3D reconstruction approaches using monocular nodes and wide-baseline multi-camera geometry have been proposed, the obvious approach for a robust system in the general case is to employ short-baseline stereo camera pairs at each node. For the associated costs of one additional imaging sensor per node, full 3D information is available locally prior to any distributed analysis. This greatly simplifies the 3D reconstruction problem and generally yields more robust results, in turn simplifying and relaxing constraints on subsequent tasks.

The basic common data object in monocular smart camera networks is the 2D image captured by a node. While many applications share similarities in the subsequent distributed analysis, there is no universal “next step” and thus no higher-level data common to all applications. In a direct structural comparison, the analog to the image in a 3D sensing network is the local 3D structure estimated by a node (based on short-baseline stereo reconstruction of its own field of view). While this is a useful starting point in itself, the nature of the data lends itself to a further step. A global reconstruction of the scene, or at least a partial one, is a natural precursor to virtually every conceivable application. While calibration is certainly possible in the monocular case (see Section 2.2 for a survey of methods), the benefits of knowing the locations and orientations of the cameras are somewhat limited when the data itself and its associated operations are still rooted in two-dimensional projections. By contrast, in the stereo case, the data

[☆] A prior draft of this work was presented at the 2nd ACM/IEEE International Conference on Distributed Smart Cameras [1]. Funding for this research was provided in part by the Natural Sciences and Engineering Research Council of Canada.

[*] Corresponding author.
E-mail addresses: mavrin1@uwindsor.ca (A. Mavrinac), xchen@uwindsor.ca (X. Chen), ktepe@uwindsor.ca (K. Tepe).

and operations are inherently three-dimensional, and therefore naturally benefit from global structure information.

Thus, the 3D visual sensor network paradigm does not simply replace local 2D images with local 3D reconstructions, but in fact extends the underlying data to a single space-time model of the scene. This shifts the focus of distributed processing to a more natural content-centric approach.

We present here a novel method for automatic spatial self-calibration of 3D visual sensor networks composed of stereo smart camera nodes. The algorithm itself is fully distributed and scalable, and as it operates purely on geometric data, it makes only the most basic explicit assumptions about node deployment and environment.

This work is organized in the following fashion. Prior related work is discussed in Section 2. Section 3 covers the basic concepts and mathematical definitions used throughout the work. In Section 4, we define and analyze the problem in geometric terms, and present the conceptual construction of the solution. Section 5 presents an overview of our final algorithm. Details of an actual hardware and software implementation are given in Section 6, and some experiments demonstrating its performance are discussed in Section 7. Concluding remarks are given in Section 8.

2. Related work

2.1. 3D (stereo) DSC networks

While the volume of distributed smart camera research has grown markedly over the past several years, monocular nodes are the dominant paradigm [2]. Jannotti and Mao [7], Heath and Guibas [4], Malik and Bajcsy [6], and Englebienne et al. [5] present cases where fusion of 3D data from multiple short-baseline stereo cameras has advantages over analogous methods using 2D monocular data. Notably, in [4,6], several issues unique to stereo camera nodes are discussed.

Here, we are further interested in the fundamentally different model of 3D visual sensor networks largely made possible by stereo camera nodes. Akdere et al. [3] elucidate this concept in presenting a distributed query mechanism for the space-time scene abstraction.

2.2. Distributed calibration

Functional calibration methods for monocular distributed smart cameras are presented by Devarajan and Radke [8] and by Mantzel et al. [9]. These are based on wide-baseline stereo methods, which are generally not robust due to the matching problem [23], and require unwieldy initialization schemes or dictate deployment constraints. Funiak et al. [10] use the motion of objects in the scene for more robust matching. The method of Medeiros et al. [15] also uses a moving object of known geometry to find correspondences; this method also uses a grouping scheme similar to that presented here. Yu and Sharma [16] present a robust feature descriptor acknowledging the network constraints of distributed smart cameras, but their calibration method requires central processing. Potentially more robust methods are presented by Beall and Qi [11], by Taylor and Shirmohammadi [12], by Barton-Sweeney et al. [13], and by Kurillo et al. [14]; however, these require the use of markers or beacons placed in the environment, which is infeasible in many cases and may constrain deployment or extension to dynamic calibration.

Jannotti and Mao [7] present a calibration method for stereo DSC networks called Lighthouse, which uses 3D point sets and geographic hash tables (GHTs) [17] to localize and orient nodes. Their method is conceptually similar to ours, up to the grouping process: it matches "point packs" constructed from 3D points, categorizing them according to a geometric hash function (similar to our geometric descriptor) via the GHT, and then merges the nodes into ever-larger groups by propagating point packs to other groups. Our method has two key advantages over Lighthouse. First, we use a more efficient grouping scheme which allows composition of poses within groups, thereby eliminating much redundant matching and pose computation (important for scalability). Second, we provide a full pairwise pose refinement stage, which is non-trivial yet absent from the description of Lighthouse.

2.3. Graph model

The concept of the vision graph was originally put forth by Devarajan and Radke [8] to model their multi-camera calibration algorithm. Kurillo et al. [14] later used it for this purpose as well. The idea was refined by Cheng et al. [18] recently, and is becoming a useful general tool for describing the directionality of nodes in distributed smart camera networks. Lobaton et al. [19] present a closely-related algebraic model for representing camera network topology, though no explicit graph formalism is employed. It is worth noting that similar graph formalisms have been used in other computer vision work; for example, by Huber [20] and by Stamos and Leordanu [21].

3. Preliminary concepts

It is useful at this point to formally define some core concepts used throughout the remainder of this work. In some cases, the purpose may not be immediately clear, but we will refer back to this section as they are further developed and employed.

3.1. Nodes and groups

A *node* is the abstract or physical smart stereo camera device itself; nodes shall be denoted by sequential capital letters (A, B, and so forth). The set of all nodes in the network shall be denoted $\mathbb{N}$ (where $|\mathbb{N}|$ represents the total number of nodes). A *group* is a set of nodes which agree on a single *leader* node; a group led by node A shall be denoted G_A (where $|G_A|$ represents the number of nodes in the group). Every group is a subset of the full set of nodes ($G_A \subseteq \mathbb{N}$), and every node is a member of exactly one group (so, if G_A and G_B are two separate groups, $|G_A \cap G_B| = 0$).

3.2. Point sets and point packs

A *point set* is the full set of 3D interest points detected locally at a node; the point set of node A shall be denoted S_A. The *overlap* between point sets S_A and S_B refers to the size of the intersection of the two sets $|S_A \cap S_B|$, said intersection occurring where a point in S_A corresponds to the same physical point as a point in S_B. The *percent overlap* is defined as follows:

$$\%O(S_A, S_B) = \frac{|S_A \cap S_B|}{\max(|S_A|, |S_B|)} \times 100\% \tag{1}$$

A *point pack* is any subset of the point set of fixed size (determined by a parameter of the algorithm); when discussing a single arbitrary point pack from node A, it shall be denoted F_A, where $F_A \subseteq S_A$. Two point packs F_A and F_B, from nodes A and B, respectively, are considered to *match* (denoted $F_A \approx F_B$) if each point in F_A corresponds to the same physical point as a point in F_B.

In the context of the algorithm, it is impossible to ascertain this correspondence, so the term *match* implies rather a presumed match based on a criterion of geometrical similarity.

3.3. Pose

Each node is considered to have its own local coordinate system. The *relative pose* of node A with respect to node B is denoted P_{AB}, and is the rigid transformation in 3D Euclidean space from the coordinate system of A to that of B. The inverse of pose P_{AB}, denoted P_{AB}^{-1}, reverses the pose transformation (so that $P_{AB}^{-1} = P_{BA}$). A succession of pose transformations $P_{BC}(P_{AB}(x))$ can be composed into a single pose, denoted $(P_{AB} \circ P_{BC})(x) = P_{AC}(x)$.

The error between two poses P_A and P_B is denoted $E(P_A, P_B)$. The actual value for this error depends on the method of computation, as it is used for comparison only; see Section 7.1.2 for a practical case.

3.4. Graph model

We make extensive use of a graph-based model in describing the calibration problem, our algorithm, and our experimental results. It consists of three undirected simple graphs, each containing vertices representing the nodes in the DSC network. The important interrelations between these graphs are mentioned here, but will become more clear in Section 4.

3.4.1. Vision graph

In the *vision graph* $\Gamma_V = (\mathbb{N}, E_V)$, an edge $e \in E_V$ indicates overlap between the fields of view of the endpoint nodes. The term *overlap* implies that sufficient information – the nature and extent of which is application-specific – is measurable by both nodes.

In our case, an edge represents a sufficient point set overlap for both the matching and fine registration components of our algorithm to yield results. We do not actually determine this graph quantitatively here; it is rather discussed as a qualitative guideline for deployment (see the shared view assumption, Section 4.2.2) and as a theoretical upper bound for the connectivity of the grouping and calibration graphs.

3.4.2. Grouping graph

In the *grouping graph* $\Gamma_G = (\mathbb{N}, E_G)$, an edge $e \in E_G$ indicates the availability of a direct coarse pose estimate between the endpoint nodes. Each component (maximally connected subgraph) of the grouping graph constitutes a group as defined in Section 3.1.

In our algorithm, $\Gamma_G \subseteq \Gamma_V$, since direct coarse pose estimates can only be obtained between nodes sharing scene points.

3.4.3. Calibration graph

In the *calibration graph* $\Gamma_C = (\mathbb{N}, E_C)$, an edge $e \in E_C$ indicates the availability of a direct fine pose estimate between the endpoint nodes. The weight of an edge is a function of the error in the associated pose estimate; this allows one to construct, via pose composition, the most accurate pose estimate for a given pair nodes using a weighted shortest path algorithm (such as Dijkstra's algorithm [31]).

In our algorithm, similarly to Γ_G, $\Gamma_C \subseteq \Gamma_V$. Also, our algorithm performs pairwise calibration within groups; therefore, a minimum condition for an edge to exist between two nodes in Γ_C is mutual reachability in Γ_G.

4. Problem analysis

4.1. Problem statement

The overall objective is to spatially calibrate a series of homogeneous smart stereo camera nodes, with no *a priori* knowledge and using only the nodes' 3D visual data, in a distributed fashion. Assuming the visual data consists of a set of 3D points triangulated from stereo images of the environment, the problem may be reduced to geometrical terms:

> Given a set of nodes $\mathbb{N}$, each node $X \in \mathbb{N}$ having a point set S_X, estimate the pose P_{XY} for enough node pairs (X, Y) such that the calibration graph for $\mathbb{N}$ is connected.

In other words, for convergence, we seek at least a spanning tree in Γ_C. Finding additional edges, up to the theoretical limit of isomorphism with Γ_V, will generally improve accuracy (see Section 7.1.2); thus, in actuality, we pursue as many direct pairwise pose estimates as possible.

4.2. Assumptions

We make the following basic assumptions about the DSC network to be calibrated.

4.2.1. Pre-deployment offline access

It is assumed that, prior to deployment of the network, there is a period during which each node may be accessed without restriction in a controlled environment, in order to perform certain essential modifications to software (such as assignment of a unique identifier, network configuration, and intrinsic/stereo calibration of the cameras).

4.2.2. Shared view

For full convergence, it is assumed that Γ_V is connected. This imposes a minimum qualitative constraint on node deployment that the shared field of view of the entire network be continuous and have substantial internal pairwise overlap.

4.2.3. Fixed nodes

It is assumed that each node is fixed in its location and orientation relative to all other nodes. It is also assumed that, once internally calibrated for stereo vision, no node changes the relative motion between its cameras or the internal parameters (e.g. focal length) of either of its cameras.

4.2.4. Static scene

It is assumed that the contents of the scene are fully static for the purposes of acquiring calibration point sets. This is solely for simplicity, and could easily be relaxed by employing background estimation techniques or accurate temporal synchronization.

4.2.5. Abstract network

It is assumed that the nodes are capable of autonomously forming an ad-hoc network of some kind, wherein each node can be addressed by a unique identifier. From the algorithm's point of view, the network is assumed to be *fully connected* [32], or in other words, the communication graph is assumed to be complete. Additionally, it is assumed that arbitrary amounts of data can be sent with assured delivery.

4.3. Two-stage registration

Bringing the point sets, and thereby the node coordinate systems, into alignment with one another can be accomplished by *registration*. Registration algorithms may be divided into two types: coarse registration, which can align points without an initial estimate but are generally not very accurate; and fine registration, which require an initial estimate to align points but are very accurate [28].

For our purposes, no alignment estimate is initially available, yet high accuracy is desirable. The typical solution when presented with such a problem is a two-stage approach, using coarse registration to initialize fine registration. However, there is more to the problem in our case: it is not even known which point sets overlap or to what degree. We use a process of *point pack matching* to determine how to proceed with registration between nodes.

4.4. Point pack categorization

We wish to evenly distribute the processing and storage of the data throughout the network based on some metric of the data itself, a process known as *data-centric storage*. Our scheme is intended to work with the generic network implied by our assumptions in Section 4.2.5; better routing efficiency and redundancy may be achievable in practice on some networks using GHT [17] or other methods.

To describe the data, we define a continuous, deterministic *geometric descriptor* function as follows:

$$g(F) = \sum_{i=1}^{f} \|F_i - F_c\| \tag{2}$$

where F_i is the ith point in the point pack, F_c is the centroid of the pack, and $\|\cdot\|$ denotes the Euclidean norm.

The solution space of this descriptor function is divided as evenly as possible among the nodes in the network, with some overlap (see below), and point packs detected locally at each node are sent to the appropriate node for matching to other geometrically similar point packs.

The difference between the descriptors of two point packs F_A and F_B describes the degree of difference d between those point packs:

$$d(F_A, F_B) = |g(F_A) - g(F_B)| \tag{3}$$

Based on the measurement accuracy of a node and the specific coarse registration algorithm used, there is a similarity threshold t_d, such that it is necessary to compare two point packs F_A and F_B if $d(F_A, F_B) < t_d$, and unnecessary otherwise; this will be termed the *similarity condition*. The desirable overlap for categorization, then, is $t_d/2$ in each direction.

Note that categorization, and thus the nodes where point packs are matched, does not depend on which nodes originated the point packs. When a match is found, the result is returned to one of the two source nodes, based on the following selection function:

$$W(A,B) = \begin{cases} A & \text{if } e_B - e_A = 1 \bmod 2 \text{ and } e_B > e_A, \\ & \text{or } e_B - e_A = 0 \bmod 2 \text{ and } e_B < e_A \\ B & \text{otherwise} \end{cases} \tag{4}$$

where e_X is a sequential integer enumeration of the node ID X. This function has been chosen to deterministically choose between two nodes in as unbiased a fashion as possible, so as to evenly distribute processing.

4.5. Point pack matching

The matching process itself compares point packs by attempting to apply the coarse registration algorithm (e.g. fully-contained DARCES [29], see Section 6.3.1); if the registration error falls below a certain threshold t_{ec}, the point packs are considered to match. The actual registration result may then be used as a coarse pose estimate between the source nodes.

4.5.1. Point pack size

Consider point sets from two nodes, S_A and S_B, from which, according to the coarse matching algorithm, each node randomly selects a point pack of size $f \geqslant 3$ (since at least 3 points are required to fix a rigid pose in 3D Euclidean space), resulting in $F_A \subseteq S_A$ and $F_B \subseteq S_B$ where $|F_A| = |F_B| = f \leqslant |S_A \cap S_B|$. The performance of the matching scheme depends on the probability of a match between F_A and F_B, $P(F_A \approx F_B)$, which can be calculated as follows (a full derivation is given in Appendix A):

$$P(F_A \approx F_B) = \frac{|S_A \cap S_B|!f!(|S_A| - f)!(|S_B| - f)!}{|S_A|!|S_B|!(|S_A \cap S_B| - f)!} \tag{5}$$

It is therefore desirable to increase $|S_A \cap S_B|$ relative to $|S_A|$ and $|S_B|$ (i.e., increase the percent overlap), which translates into repeatability in interest point detection (see Section 6.2.1). There is a trade-off in the value of f, where a lower value yields more matches and improved speed, but a higher value reduces the occurrence of false positives; generally, a relatively low value such as $f = 4$ is adequate.

4.6. Grouping

In order to allow all node pairs adjacent in Γ_V to attempt pairwise pose refinement without the need for exhaustive point pack matching, we introduce a grouping scheme wherein nodes are merged into ever-larger groups within the same coordinate system, albeit with only coarse estimates. Through pose composition on Γ_G, any node in a group can determine its coarse pose with respect to any other node.

Each node is aware of its current group leader and the set of nodes comprising its group. Within a group (Section 3.1), each node has a coarse pose estimate relative to the group leader, called its *group coarse pose estimate*, and denoted C_A for a node A. Relative coarse pose estimates (e.g. C_{AB} for node A relative to node B) can be computed from these, either directly or through one or more compositions. Initially, each node begins in a singleton group, of which it is the leader, with its group coarse pose estimate initialized to P_I.

A *merge* is initiated when two nodes have detected a certain minimum number t_m of consistent matches with each other. Consistency is enforced via a threshold t_c specifying the minimum Euclidean distance between the pose estimates' mappings of a given point (such as the centroid μ_S of the computing node's point set). Once a node has stored at least t_m matches with a particular other node, each time a new match is detected for that node, an average coarse pose estimate is computed for every combination M_i of matches containing the new match, and checked for consistency:

$$\|C_m(\mu_S) - C_{avg}(\mu_S)\| \leqslant t_c, \quad \forall\, m \in M_i \tag{6}$$

If a consistent average is found, it is considered a reliable relative coarse pose estimate, and is forwarded to the source nodes' group leaders and composed as necessary to merge the nodes' respective groups (see Fig. 1).

Fig. 2 illustrates a typical group merge. Node D, of group G_A, and node G, of group G_F, find a relative coarse pose estimate through point pack matching, and initiate a merge. The nodes in group G_A do not modify their group coarse pose information, while those formerly of group G_F now honor A as their group leader and modify their pose information accordingly. Node G's new group coarse pose estimate (C'_G) is the composition of its estimated pose relative to node D with node D's group coarse pose estimate:

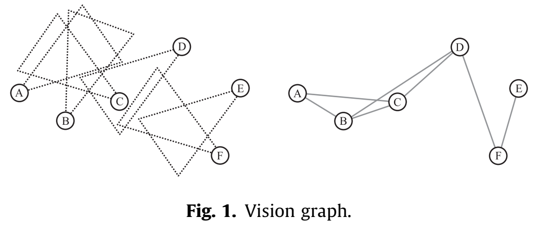


Fig. 1. Vision graph.

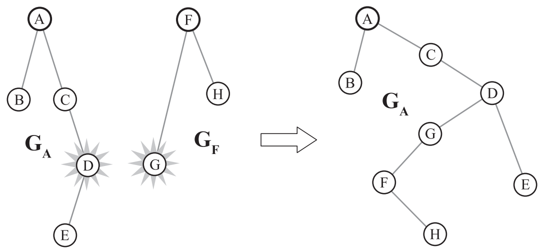


Fig. 2. Group merging.

$$C'_G = C_{GD} \circ C_D \tag{7}$$

The new group coarse pose estimates for the merging group's leader (C'_F) and any other nodes in the merging group (in this case, C'_H) can similarly be calculated as compositions of known pose estimates:

$$C'_F = C_G^{-1} \circ (C_{GD} \circ C_D) \tag{8}$$

$$C'_H = C_H \circ (C_G^{-1} \circ (C_{GD} \circ C_D)) \tag{9}$$

Since merging consists of composition operations, it is a transitive operation which can occur based on matches (and the resultant relative coarse pose estimates) between any pair of nodes in different groups. Fig. 2 illustrates this by showing the actual history of merges leading to the groups as arrows between the node pairs; in reality, of course, every node in the group has a direct pose estimate to the leader (group coarse pose estimate).

4.7. Pairwise pose refinement

Once a given pair of nodes belong to the same group, those nodes can use their coarse relative pose estimate, computed by composition on the walk between them in Γ_G, as a starting point for pose refinement. This is achieved by applying a fine registration algorithm (e.g. TrICP [30], see Section 6.3.1) to a large number of points initialized into coarse alignment.

As shown in Fig. 3, the actual point sets used for fine registration are selected, at each node, as those falling within the intersection of the two nodes' fields of view, as approximated by a cone of a certain angle and length extending along the positive z-axis of each node's coordinate system (via the coarse pose estimate). If there are fewer than a specified minimum number of points, which includes the case where there is no intersection at all, the nodes do not attempt pose refinement.

4.8. Indirect pose estimation

If a node B is reachable from a node A in Γ_C, P_{AB} may be estimated as the composition of the poses represented by the edges in the shortest walk from A to B.

A practical distributed implementation would involve each node having a copy of the current Γ_C. A node attempting to find a pose estimate would compute the shortest path to the destination node via Dijkstra's algorithm [31], yielding an approximation of the total composition of pose errors. It would then request the appropriate pairwise pose estimates from each node in the sequence, and compose them to obtain an indirect pose estimate, caching the result for future access.

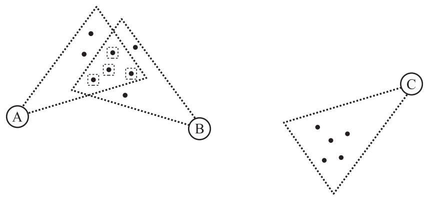


Fig. 3. Field of view cone approximation.

5. Calibration algorithm

Our calibration method is embodied by a completely distributed and asynchronous event-based algorithm. It is split into ten different processes at each node, six for coarse grouping and four for pairwise pose refinement. The processes execute within the context of their respective nodes' data spaces. Calibration is complete at a node when all ten processes at that node have terminated.

There are four necessary parameters to the algorithm itself: the point pack size f, the similarity threshold t_d, the match threshold t_m, and the consistency threshold t_c. Certain other parameters are also required depending on the implementation, notably the coarse and fine registration error thresholds t_{ec} and t_{ef}. The values of these parameters are, of course, common to all nodes.

The *point pack selection process* (Fig. 4) periodically – on an interval determined by practical limitations of the devices and network – selects a point pack of f points from the point set, computes its descriptor, and sends it to the destination node according to the binning function.

The *point pack matching process* (Fig. 5) maintains a database of received point packs. Each time a new point pack is received, it is compared to all previous point packs meeting the similarity condition via the coarse registration algorithm. For each matching point pack (where registration error $e < t_{ec}$), the match result is sent to one of the source nodes based on an unbiased, deterministic selection process.

The *match processing process* (Fig. 6) maintains an array of matches to each other node, to which incoming matches are added (provided that the other node is not already in this node's group). If

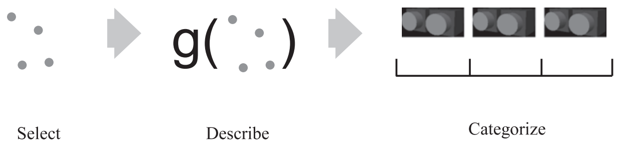


Fig. 4. Point pack selection process.

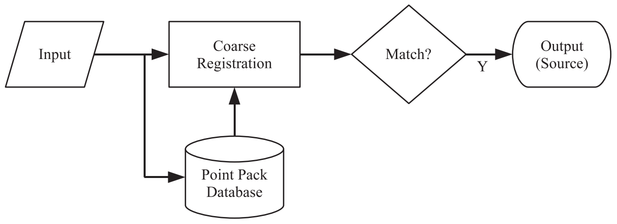


Fig. 5. Point pack matching process.

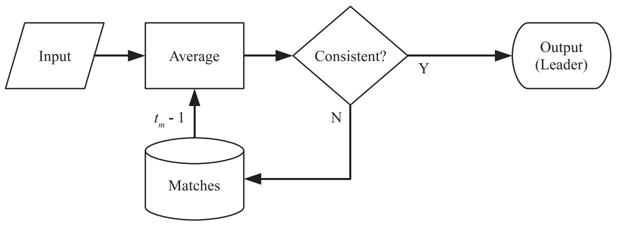


Fig. 6. Match processing process.

there are at least t_m matches, an average pose transformation is computed, and each match is checked for consistency by ensuring the Euclidean distance of a point projected through both poses is no more than t_c. If successful, a message is sent to the group merge initiator process of this node's group leader.

The *group merge initiator process* (Fig. 7) initiates a merge with the other node in a merge message by messaging its group merge responder process. These processes employ a *merge lock* to prevent other asynchronous merges from occurring simultaneously, which would otherwise put the group in an indeterminate state. When it receives acknowledgment, it sends a group update message to all nodes in the group, updating them to the new group leader and composition as well as their new group coarse pose estimates.

The *group merge responder process* (Fig. 8) responds to a merge message with an acknowledgment. It then sends a group update message to all nodes in the group, updating only their group composition.

The *group update process* (Fig. 9) simply awaits group update messages and applies them to local group information in the proper order.

The *pose refinement initiator process* (Fig. 10) activates whenever the local group information is updated. For each new node in the group, it performs an unbiased, deterministic selection, and sends an initiation message containing its group coarse pose estimate to each selected node.

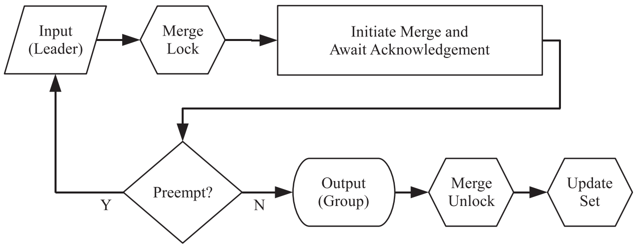


Fig. 7. Group merge initiator process.

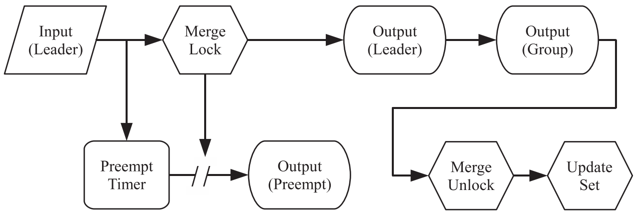


Fig. 8. Group merge responder process.

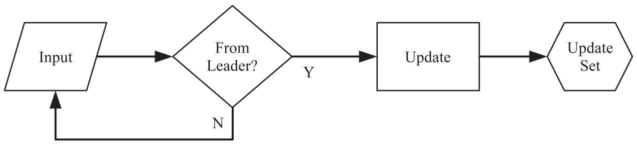


Fig. 9. Group update process.

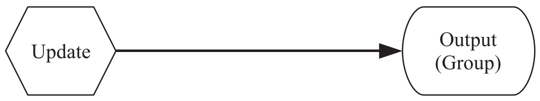


Fig. 10. Pose refinement initiator process.

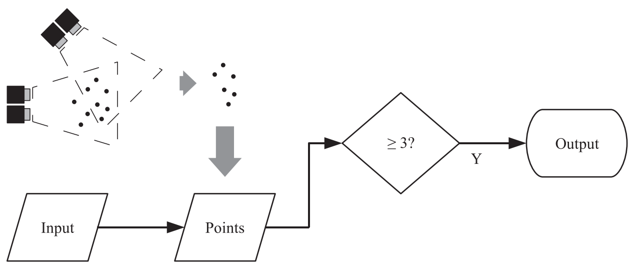


Fig. 11. Pose refinement responder process.

The *pose refinement responder process* (Fig. 11) uses the group coarse pose estimate of the initiating node to select a subset of candidate points from its point set, based on a cone approximation of the node's field of view. If there are at least 3 points in the subset, it sends them back to the initiating node for fine registration.

The *fine registration process* (Fig. 12) attempts to compute an accurate relative pose estimate with the responding node via fine registration of its point set with the responder's point subset. If the registration error $e < t_{ef}$, the result is stored and an update (containing the inverse of the pose) is sent to the responding node.

The *pose update process* (Fig. 13) simply awaits pose update messages and stores the results.

6. Implementation

6.1. Apparatus

The physical stereo smart camera "nodes" in our experimental implementation are pairs of 1.4-megapixel CCD cameras, mounted in a short-baseline stereo configuration and connected to a PC via a IEEE-1394 (Firewire) interface. These cameras are calibrated for stereo vision using Bouguet's method [33] through its implementation in the OpenCV computer vision library. The PC to which any given pair is attached performs local smart camera functionality as well as its processes in the distributed calibration algorithm. The network is a pre-arranged set of TCP/IP hosts representing the PCs.

6.2. Local point detection

Detection of 3D interest points at each node, while not actually part of the calibration algorithm itself, is a very important compo-

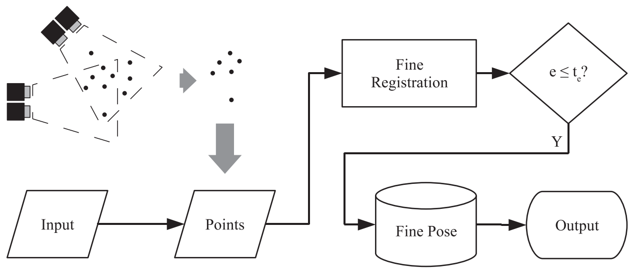


Fig. 12. Fine registration process.

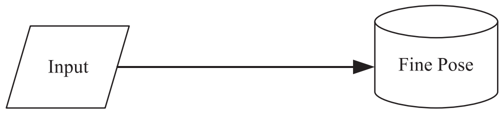


Fig. 13. Pose update process.

nent of the overall process. In order to achieve the goal of calibration in a completely automatic fashion, we have implemented software which detects and triangulates a set of 3D interest points from a stereo pair of images, to be run at each node. Section 6.2.1 expands on the actual process and its impact on our calibration method.

The software itself is implemented in C, using the open-source OpenCV computer vision library.

6.2.1. Interest point detection

The algorithm described in this work relies heavily on stable (repeatable) interest point detection invariant to 3D Euclidean transformations in the viewpoint, as the percent overlap (Eq. (1)) between nodes' point sets must be significant to achieve good results. Most current methods are scale- and affine-invariant [23] and thus do not achieve this directly.

Our current implementation detects Harris corners [24] (the Shi and Tomasi [25] variant; in previous experiments we used FAST [26,27]) in each image of the stereo pair, then uses epipolar-constrained ZNCC (zero-normalized cross-correlation) to match them, and finally employs epipolar geometry to triangulate their 3D positions [22]. We employ some thresholding on the 2D interest points as well as the matching function to limit the results to a relatively small number (on the order of 30–80) of 3D interest points.

This scheme unfortunately does not yield the necessary repeatability for calibration against an arbitrary scene. We therefore currently use one or more calibration targets, which are simply objects with strong, easily detected interest points placed within the visible scene.

6.3. PyDSSCC

We provide a full implementation of the algorithm itself as a Python module, called PyDSSCC (Python Distributed Smart Stereo Camera Calibrator). It implements each of the ten processes described in Section 5 as a thread, and uses thread-safe message queues for input and output messages.

PyDSSCC is free software licensed under the GNU General Public License and may be obtained online.

6.3.1. Registration

PyDSSCC also includes implementations of the necessary registration algorithms. For coarse registration, a fully-contained DARCES [29] implementation is employed, as the point packs in this stage consist of small numbers of points and must match completely. For fine registration, an implementation of the Trimmed Iterative Closest Point algorithm [30] is used, which works well for the varying degrees of point overlap one expects to obtain (as discussed in Section 6.2.1).

7. Experimental results

7.1. Performance metrics

We have selected three general performance metrics which may quantify the usefulness of any distributed smart camera calibration method. These are fairly straightforward when used for design, as our interpretation of each is directly related to some relevant characteristic of the calibrated 3D DSC network. Quantitative comparisons with other algorithms prove more difficult, as the interpretations of the metrics vary somewhat depending on the fundamental nature of the algorithm.

7.1.1. Convergence

Convergence is the measure of the algorithm's ability to bring nodes into a common reference frame and its time performance in doing so. For our purposes, there are actually two levels to consider:

1. The ability of coarse grouping to merge into a minimum number of groups.
2. The ability of pairwise pose refinement to establish a maximum number of pairwise estimates.

Calibration is considered successful in terms of convergence when Γ_G and Γ_C are connected.

7.1.2. Accuracy

Accuracy is the measure of the error in the algorithm's resulting pose estimates. With a direct pairwise pose estimate, the accuracy is simply the inverse of the error in the estimate. For any pair of nodes mutually reachable in Γ_C, then, the accuracy is approximately the inverse of the total error (weight) along the shortest path.

The global average of error accumulated between two arbitrary nodes will generally increase with the size of the network, and is also subject to various other topological influences. A more useful metric is the local accuracy of 3D reconstructions up to a certain distance in terms of field of view (i.e., walk length in Γ_V). We define the n-hop error at a node A as follows:

$$E_n(A) = \sum_X \frac{E(P_A, P_X)}{|\mathbb{N}_n(A)|}, \quad \forall X \in \mathbb{N}_n(A) \tag{10}$$

where $\mathbb{N}_n(A)$ is the set of all nodes within a walk of length n or shorter of node A in a qualitative estimation of Γ_V. This is computed for all nodes in $\mathbb{N}$ and averaged to obtain the overall error.

The mean error in a pose estimate can be determined by averaging the Euclidean distance between a number of points with ground-truth correspondence, detected and triangulated at the nodes separately from those used for calibration.

7.1.3. Scalability

Scalability is the measure of the effect on the algorithm's performance of the number of nodes in the network. The three primary resources to consider are node-local computing resources (i.e., CPU and memory), node-local data storage, and network bandwidth.

In order to properly evaluate scalability, it is necessary to examine individual factors arising from the algorithm itself. The most significant of these can be summarized in terms of the number of nodes in the network $|\mathbb{N}|$ as follows:

- Point pack dissemination requires bandwidth resources in $|\mathbb{N}|$ *per node*.
- Point pack matching requires computing and storage resources in $|\mathbb{N}|$.

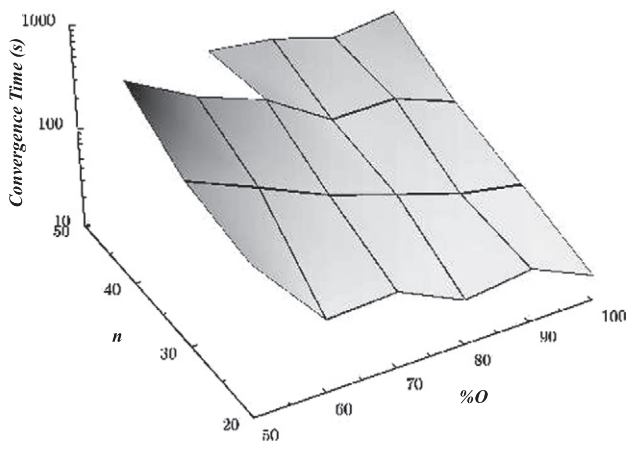
This assumes that each node maintains a more or less constant number of pairwise edges in Γ_V regardless of $|\mathbb{N}|$, as would be the case with most applications. In cases where this assumption does not hold, it is necessary to add a third factor:

- Pairwise pose refinement computation requires computing resources in $|\mathbb{N}|$.

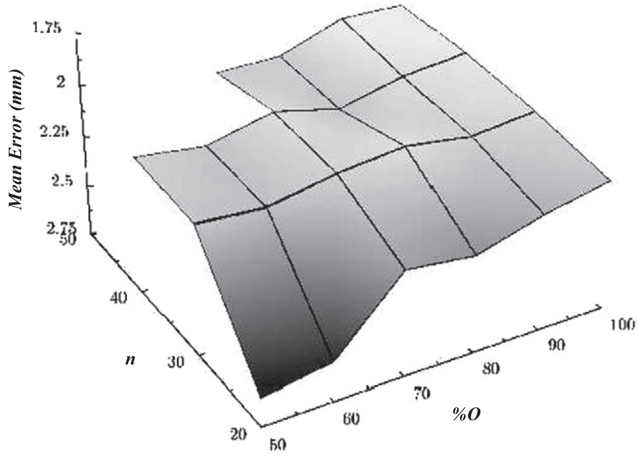
Scalability in all three resources can be quantized experimentally in terms of the above factors.

7.2. Manual point set

In order to test the capabilities of the calibration algorithm and tune its parameters under controlled conditions, the first



(a) Convergence Time Trends in n and $\%O$



(b) Accuracy Trends in n and $\%O$

Fig. 14. Manual experiment results.

experiment series is designed to operate on manually selected points with full correspondences across all four nodes. The primary purpose of this experiment type, once suitable parameters are found, is to test the effects of different point set sizes and overlap characteristics on convergence and accuracy.

7.2.1. Procedure

A total of 22 point subsets are extracted from the data, and each is tested using the distributed calibration software, with all four nodes running locally on the same workstation. This procedure is repeated twice for each subset, and the average results for convergence time and mean error are calculated and recorded.

7.2.2. Results

Fig. 14 shows the trends in the results from the manual point set experiments, within the ranges where calibration yielded reasonable results. As n increases, the accuracy increases as there are more points for fine registration. However, convergence time also increases because more point packs must be tested for matching. As $\%\ O$ decreases, convergence time increases and accuracy decreases.

7.3. Automatic point set

Having established some criteria for reasonably timely convergence in the manual point set experiments, the next step is to test

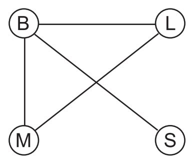


Fig. 15. Calibration graph for automatic experiment.

real automatic calibration of the network. The purpose of these experiments is to test the convergence and accuracy performance of the algorithm in real conditions.

7.3.1. Procedure

Four instances of the software stack are run on the PCs to which four physical stereo camera rigs are connected. Convergence time and the final Γ_C are recorded. A ground truth point set is manually selected for each camera rig, and the mean error is calculated and recorded.

7.3.2. Results

The mean error and convergence time of a typical experiment from this series is shown below. Fig. 15 shows the final Γ_C, Fig. 16 shows the physical deployment of the nodes for the experiment, and Fig. 17 shows the visualization of the resultant pose estimates (which can be seen to match the configuration in Fig. 16).

- *Mean error:* 2.7666 mm
- *Convergence time:* 159 s

We note that actual convergence time depends heavily on a number of system-specific details, so these results are presented here to demonstrate feasibility and for internal comparison.

7.4. Virtual point set

Since only four physical camera rigs are available, testing scalability to larger networks is impossible in an automatic experiment and difficult to control using the manual methods. Instead, controlled virtual point sets are supplied to the same calibration algorithm implementation to test the scalability metric.

7.4.1. Procedure

Point sets are generated for 5, 10, 15, 20, and 25 nodes. The total outgoing bandwidth in kilobytes, final size of the matching database in point packs, and total number of coarse and fine registration executions are recorded.

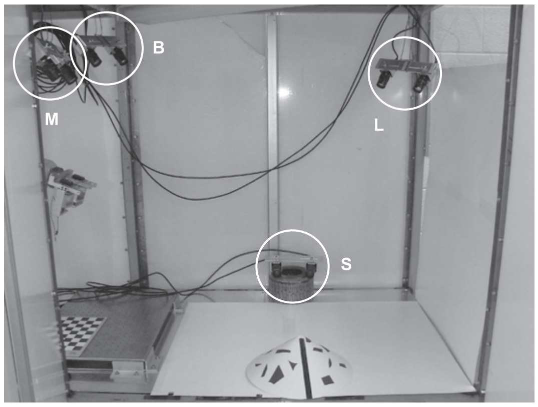


Fig. 16. Camera deployment for automatic experiment.

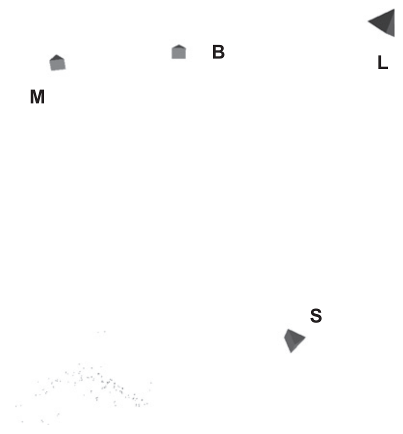


Fig. 17. Pose visualization for automatic experiment.

7.4.2. Results

As expected, total bandwidth usage per node increases approximately linearly in relation to the number of nodes in the network (Fig. 18). This affects different networks in different ways. In a network where the physical medium is shared by all nodes – the worst-case scenario – the total network bandwidth usage is the relevant factor. In that case, the bandwidth usage increases non-linearly, potentially at up to $|\mathbb{N}|^3$. However, many routing methods used in sensor networks are much more efficient and therefore mitigate this effect.

The number of point packs stored at each node increases approximately linearly in relation to the number of nodes (Fig. 19). Point packs are very small data (a series of f 3-tuples, an identifier, and a geometric descriptor value), but when scaling to extremely large networks it must be ensured that adequate storage is provided at each node for these point packs.

The number of coarse registration operations performed at each node increases approximately linearly in relation to the number of nodes (Fig. 20); as expected, this is proportional to the number of point packs stored. If processing throughput is the limiting factor, this increase will cause the convergence time to increase linearly with the number of nodes.

Since, in this network, the number of vision graph edges per node does not generally increase as its total number of nodes increases, the number of fine registrations per node is approximately constant.

8. Conclusions

A feature-based calibration method for distributed smart stereo camera networks has been developed which converges well, provides accurate pairwise orientation, and scales well with network size within the scope of our investigation. This provides a base upon which to build a full 3D visual sensor network providing primitive data-centric queries, upon which in turn a variety of high-level applications can be developed. We have provided both a general theoretical development (Section 4), and a description of an asynchronous distributed algorithm (Section 5) with implementation (Section 6).

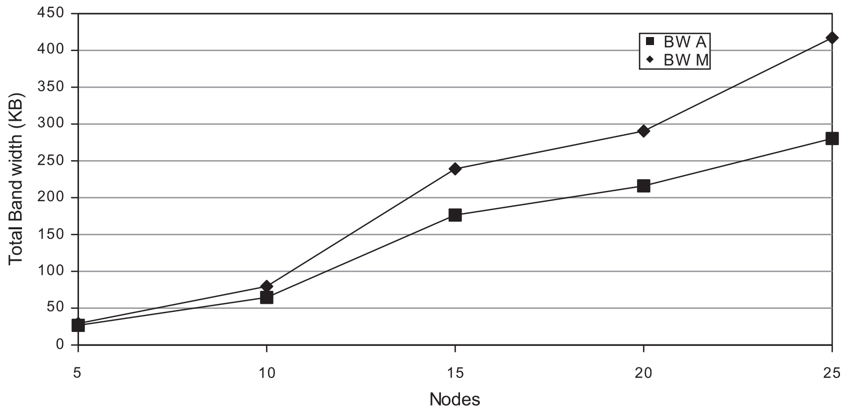


Fig. 18. Bandwidth usage in $|\mathbb{N}|$ (average and maximum).

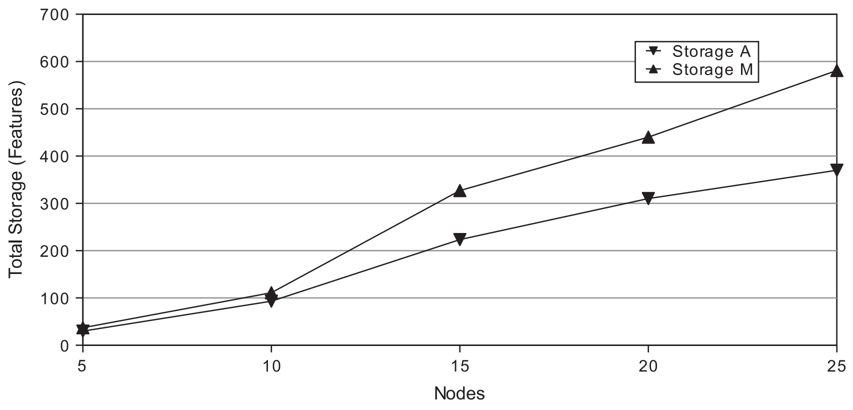


Fig. 19. Node-local storage in $|\mathbb{N}|$ (average and maximum).

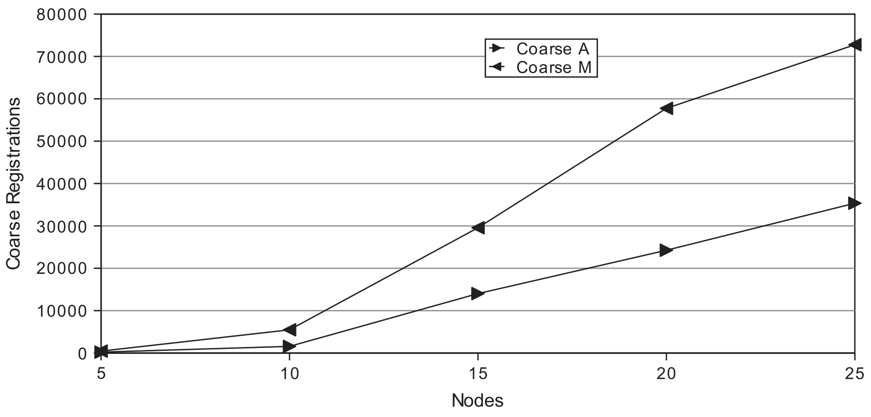


Fig. 20. Coarse registration processing in $|\mathbb{N}|$ (average and maximum).

Due to our implementation's reliance on the repeatability of local interest point detection, which has been shown to be poor over large general Euclidean transformations of the viewpoint [23], it is currently necessary to control the scene somewhat (e.g. using calibration targets) to ensure convergence. However, as this is a limitation of interest point detection and is not inherent to our algorithm, direct substitution of feature detection methods may easily improve this in future implementations.

Appendix A. Derivation of point pack matching probability (Eq. (5))

First, the individual probabilities that F_A and F_B will be within the set of shared points $S_A \cap S_B$ are found. Let $P(Q) = P(F_A \subseteq S_A \cap S_B)$ and $P(R) = P(F_B \subseteq S_A \cap S_B)$, and let $C(n,k)$ represent the binomial coefficient, indicating the number of possible non-repeating combinations of size k chosen from a set of size n.

$$P(Q) = \frac{C(|S_A \cap S_B|, f)}{C(|S_A|, f)} \tag{11}$$

$$P(R) = \frac{C(|S_A \cap S_B|, f)}{C(|S_B|, f)} \tag{12}$$

$$P(Q \cap R) = P(Q)P(R) = \frac{C(|S_A \cap S_B|, f)^2}{C(|S_A|, f)C(|S_B|, f)} \tag{13}$$

Assuming that conditions Q and R are satisfied (that is, all points in packs F_A and F_B are shared in $S_A \cap S_B$), for a given F_A, only one combination F_B will match it.

$$P(F_A \approx F_B | Q \cap R) = \frac{1}{C(|S_A \cap S_B|, f)} \tag{14}$$

From Eqs. 13 and 14, $P(F_A \approx F_B)$ can be calculated as

$$\begin{aligned} P(F_A \approx F_B) &= P(F_A \approx F_B | Q \cap R)P(Q \cap R) \\ &= \left[\frac{1}{C(|S_A \cap S_B|, f)}\right]\left[\frac{C(|S_A \cap S_B|, f)^2}{C(|S_A|, f)C(|S_B|, f)}\right] \\ &= \frac{C(|S_A \cap S_B|, f)}{C(|S_A|, f)C(|S_B|, f)} = \frac{|S_A \cap S_B|!f!(|S_A| - f)!(|S_B| - f)!}{|S_A|!|S_B|!(|S_A \cap S_B| - f)!} \end{aligned} \tag{15}$$

References

[1] A. Mavrinac, X. Chen, K. Tepe, Feature-based calibration of distributed smart stereo camera networks, in: Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras, 2008.
[2] S. Soro, W. Heinzelman, A survey of visual sensor networks, in: Advances in Multimedia, 2009.
[3] M. Akdere, U. Cetintemel, D. Crispell, J. Jannotti, J. Mao, G. Taubin, Data-centric visual sensor networks for 3D sensing, in: Proceedings of the 2nd International Conference on Geosensor Networks, 2006.
[4] K. Heath, L. Guibas, Multi-person tracking from sparse 3D trajectories in a camera sensor network, in: Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras, 2008.
[5] G. Englebienne, T. van Oosterhout, B. Kröse, Tracking in sparse multi-camera setups using stereo vision, in: Proceedings of the 3rd ACM/IEEE International Conference on Distributed Smart Cameras, 2009.
[6] R. Malik, P. Bajcsy, Automated placement of multiple stereo cameras, in: Proceedings of the 8th ECCV Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras, 2008.
[7] J. Jannotti, J. Mao, Distributed calibration of smart cameras, in: Proceedings of the International Workshop on Distributed Smart Cameras, 2006, pp. 55–61.
[8] D. Devarajan, R.J. Radke, Distributed metric calibration of large camera networks, in: Proceedings of the 1st Workshop on Broadband Advanced Sensor Networks, 2004.
[9] W.E. Mantzel, H. Choi, R.G. Baraniuk, Distributed camera network localization, in: Proceedings of the 38th Asilomar Conference on Signals, Systems and Computers, 2004.
[10] S. Funiak, C. Guestrin, M. Paskin, R. Sukthankar, Distributed localization of networked cameras, in: Proceedings of the 5th International Conference on Information Processing in Sensor Networks, 2006, pp. 34–42.
[11] C. Beall, H. Qi, Distributed self-deployment in visual sensor networks, in: Proceedings of the International Conference on Control, Automation, Robotics and Vision, 2006, pp. 1–6.
[12] C.J. Taylor, B. Shirmohammadi, Self localizing smart camera networks and their applications to 3D modeling, in: Proceedings of the International Workshop on Distributed Smart Cameras, 2006, pp. 46–50.
[13] A. Barton-Sweeney, D. Lymberopoulos, A. Savvides, Sensor localization and camera calibration in distributed camera sensor networks, in: Proceedings of the 3rd International Conference on Broadband Communications, Networks and Systems, 2006, pp. 1–10.
[14] G. Kurillo, Z. Li, R. Bajcsy, Wide-area external multi-camera calibration using vision graphs and virtual calibration object, in: Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras, 2008.
[15] H. Medeiros, H. Iwaki, J. Park, Online distributed calibration of a large network of wireless cameras using dynamic clustering, in: Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras, 2008.
[16] C. Yu, G. Sharma, Q-SIFT: efficient feature descriptors for distributed camera calibration, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2009, pp. 1849–1852.
[17] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker, GHT: a geographic hash table for data-centric storage, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, 2002, pp. 78–87.
[18] Z. Cheng, D. Devarajan, R. Radke, Determining vision graphs for distributed camera networks using feature digests, EURASIP Journal on Advances in Signal Processing 2007 (2007) 11, Article ID 57034.
[19] E. Lobaton, P. Ahammad, S.S. Sastry, Building an algebraic topological model of wireless camera networks, in: H. Aghajan, A. Cavallaro (Eds.), Multi-Camera Networks, Elsevier, 2009, pp. 95–115.
[20] D.F. Huber, Automatic 3D modeling using range images obtained from unknown viewpoints, in: Proceedings of the 3rd International Conference on 3D Digital Imaging and Modeling, 2001, pp. 153–160.
[21] I. Stamos, M. Leordeanu, Automated feature-based range registration of urban scenes of large scale, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 555–561, 2003.
[22] H.C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, Nature 293 (1981) 133–135.
[23] A. Baumberg, Reliable feature matching across widely separated views, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1774–1781, 2000.

[24] C. Harris, M. Stephens, A combined corner and edge detector, in: Proceedings of the 4th Alvey Vision Conference, pp. 147–151, 1988.
[25] J. Shi, C. Tomasi, Good features to track, in: Proceedings of the 9th IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600, 1994.
[26] E. Rosten, T. Drummond, Fusing points and lines for high performance tracking, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 1508–1511, 2005.
[27] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: Proceedings of the 9th European Conference on Computer Vision, pp. 430–443, 2006.
[28] J. Salvi, C. Matabosch, D. Fofi, J. Forest, A review of recent range image registration methods with accuracy evaluation, Image and Vision Computing 25 (5) (2007) 578–596.
[29] C.-S. Chen, Y.-P. Hung, J.-B. Cheng, RANSAC-based DARCES: a new approach to fast automatic registration of partially overlapping range images, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (11) (1999) 1229–1234.
[30] D. Chetverikov, D. Svirko, D. Stepanov, P. Krsek, The trimmed iterative closest point algorithm, in: Proceedings of the International Conference on Pattern Recognition, pp. 545–548, 2002.
[31] E.W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.
[32] M. Raynal, Distributed Algorithms and Protocols, John Wiley & Sons, 1988.
[33] J. Bouguet, Camera Calibration Toolbox for Matlab. <http://www.vision.caltech.edu/bouguetj/calib_doc> (current January 3, 2008).