Affinity-based Distributed Algorithm for 3D Reconstruction in Large Scale Visual Sensor Networks

Andrea Masiero[1] and Angelo Cenedese[2]

Abstract

In recent years, Visual Sensor Networks have emerged as an interesting category of distributed sensor-actor systems to retrieve data from the observed scene and produce information. Indeed, the request for accurate 3D scene reconstruction in several applications is leading to the development of very large systems and more specifically to large scale motion capture systems. When dealing with such huge amount of data from a large number of cameras it becomes very hard to make real time reconstruction on a single machine.

Within this context, a distributed approach for reconstruction on large scale camera networks is proposed. The approach is based on geometric triangulation performed in a distributed fashion on the computational grid formed by the camera network organized into a tree structure. Since the computational performance of the algorithm strongly depends on the order in which cameras are paired, to optimize the reconstruction a pairing strategy is designed that relies on an affinity score among cameras. This score is computed from a probabilistic perspective by studying the variance of the 3D target reconstruction error and resorting to a normalized cut graph partitioning.

The scaling laws and the results obtained in simulation suggest that the proposed strategy allows to obtain a significant reduction of the computational time.

I. INTRODUCTION

The recent technological advancements in the development of microelectronic sensors and the capillary diffusion of mobile devices are driving an increasing interest of the industrial and academic communities on the distributed systems and smart sensing topics. Networks of sensors are typically used to monitor the dynamics of specific phenomena or to measure the temporal evolution of some features or parameters of a system. In this context, the information provided by a single sensor is usually insufficient to obtain a complete and reliable description of the event of interest, but this gap can be bridged by resorting to a network of heterogeneous and often actuated devices that merge their information to obtain a more complete scene representation and knowledge.

The list of applications related to the sensor network framework is ever-growing and, in particular, Visual Sensor Networks (VSNs, i.e. camera networks), are emerging as an interesting class of distributed sensor-actor systems with peculiar detection capabilities and challenging issues related to the quantity and quality of the information flow they can retrieve from the observed scene [24]. Specifically, this paper concerns the study of a VSN aiming at the reconstruction of the 3D geometry of a scene. Among the different techniques proposed in the literature [22], [12] the focus here is on the case of a triangulation based reconstruction, and in particular on the use of smart cameras [13], [21], [2] and motion capture (MoCap) systems [29].

A number of works have been recently presented in the literature concerning how to properly merge the information over a network to make the sensors converge to a *consensus* [20], [31], [8], and how to possibly speed up such procedure exploiting the characteristics of the network itself [30], [4], [27]. Similarly to the consensus algorithm, our goal is to exploit both the information provided by all the cameras and the power of the network as a computational grid, and to do so each node of the network is provided of computational intelligence (smart camera network). However, differently from the original consensus problem, in the considered complex scene the system has to deal with measurements of different targets, and consequently with a data association problem. This specific characteristic and the requirement of real time reconstruction for example in large scale

[1]A.Masiero is with the Interdepartmental Research Center of Geomatics (CIRGEO), Università di Padova, via dell'Università 16, 35020 Legnaro (PD), Italy masiero@dei.unipd.it

[2]A.Cenedese is with the Dipartimento di Ingegneria dell'Informazione, Università di Padova, via Gradenigo 6/B, 35131 Padova, Italy angelo.cenedese@unipd.it

This report is a preprint version of the American Control Conference 2014 (ACC14) paper. If needed, please reference the ACC paper (same authors and title).

MoCap systems (e.g. hundreds of targets and dozens of cameras) urge to study a distributed reconstruction strategy specifically for these VSNs.

After a preliminary overview of the reconstruction approach given in Sec. II, Sec. III regards the formulation of a distributed algorithm to efficiently deal with the 3D reconstruction problem in large MoCap systems. The strategy proposed in this work is to distribute the computation over a binary tree, where each node of the tree corresponds to a camera. As shown in the simulations of Sec. V, this approach exploits the spatial clustering of typical real data [32], [1], [14], allowing a significant reduction of the reconstruction computational time. Different strategies to distribute the computation burden over a camera network have been considered in [3], [5], [6], [15].

In the second part of the paper, Sec. IV, a strategy to optimize the proposed binary tree reconstruction procedure is proposed. This optimization is based on the formulation of an *affinity measure* between cameras, related to the mean reconstruction error that can be obtained when employing the considered cameras. The affinity score is derived for a fixed target probability density and cameras configuration; however, the problem can be posed also from a dynamic point of view, and if the target probability density changes the affinity function can be updated consequently. The similarity measure is used to determine the coupling order for the network cameras in the reconstruction procedure, obtained by considering the affinity measure as weights of links on a graph and by using n-cut to compute proper partitions of the network [23].

Even if the problem here is formulated in the MoCap framework, actually it is closely related also to other areas in VSN research such as the structure from motion problem (e.g. [19]), or the multi-view stereo context (see [9], [7]), where suitable "affinity" functions are used to properly select a set of "optimal" views. Differently from these approaches basically relying on already acquired experimental data, the affinity measure proposed in this paper relies on a probabilistic point of view (e.g. by exploiting target density distribution) and can be adapted to such multi-view stereo problems provided that the probabilistic approach is integrated with some deterministic information about features and cameras extracted from the considered images.

II. APPROACH OVERVIEW

The reconstruction of target positions from measurements taken from a set of cameras can be formulated as a geometric problem of triangulation [10], [11], [17]. Let a target i be placed at position ϕ_i in the 3D space and suppose that camera j_1 has a local measurement of i on its image plane at position q_{ij_1}. Then, using this measurement it is possible to say that i is (approximately) on a point along the line r_{ij_1} passing through q_{ij_1} and the optical center of camera j. Let also r_{ij_2} be the ray associated to another camera $j_2 \neq j_1$ and the same target i, then the rationale is that the target position $\bar{\phi}_i$ can be estimated by intersecting r_{ij_1} and r_{ij_2} (geometric triangulation). The described procedure can be adapted to deal with the multi-camera case and to take into account of the geometric errors on camera image planes [10], [11], [25].

If the system is composed by a limited number of cameras and targets, the reconstruction algorithm based on geometric triangulation can be implemented in a centralized fashion on a single machine to track the targets in real time. On the other hand, when considering the envisaged large system scenarios, it becomes difficult to simultaneously take into account the data provided by all the cameras. So, in this paper a different approach is proposed: first, only portions of the whole system are considered concurrently, and then the 3D reconstruction is achieved by progressively merging data from these different (and smaller) subsystems. Clearly in this framework, it is important to understand how the information is elaborated and merged by different cameras, meaning that some pairs of cameras will allow a better reconstruction[1] with respect to others.

With respect to a classical reconstruction scheme, where the matching of measurements is concurrently checked on all the cameras, here some modifications are introduced to make the problem manageable, as described in the following. Similarly to [9], [7] only small groups of cameras are used at the beginning (here we exploit camera pairs to compute candidate targets), and the obtained results are then assessed and refined using all the available data. Given m cameras and n targets, the strategy can be summarized as follows:

[1]The concept of reconstruction quality considered here wants to take into account several factors, among which the number of reconstructed targets, the reconstruction accuracy, and the required computational time.

• *matching*: for each pair of cameras (j_1, j_2), with $j_1 = 1, \ldots, m$ and $j_2 = j_1 + 1, \ldots, m$, the available 2D measurements from j_1 are compared with 2D measurements from j_2, searching for possible 3D real points through a geometric triangulation procedure;

• *back-projection*: when a pair of measurements from (j_1, j_2) is compatible, then the reconstructed point ϕ_i (potentially a real target position) is back-projected onto the other cameras image planes;

• *reconstruction*: let k be the number of measurements from different cameras that are compatible with ϕ_i. If k is larger than a chosen threshold $\bar{k}$, ϕ_i is recomputed by using all the k measurements to produce the estimate of the real target position $\hat{\phi}_i \approx \bar{\phi}_i$. The used measurements are deleted from the list of measurements available for new target search.

This algorithm, considered as a *centralized* procedure running on a single machine, shows a computational complexity increasing approximatively linearly with m and n, thus making this procedure not suitable for large scale scenarios. Conversely, if adapted to work in a *distributed* fashion on the computational grid formed by the camera network, a huge speed up of the whole reconstruction can be achieved. The rationale is that cameras that allow to reconstruct more targets have to be matched first, such that most of the 2D measurements are deleted quickly (in the first step of the reconstruction procedure).

The method proposed to distribute the computational load considers the cameras connected as in a binary tree: cameras correspond to the leaves of the tree, then at each level two groups of the previous level are merged. The computations at each node of the same level can be done in parallel, while different levels can be processed in a pipelined fashion. The camera connection is similar to that in [3], however the considered reconstruction procedures are different.

Then, the focus of the study is to find a strategy to pair cameras efficiently, namely to define and compute a suitable *affinity measure* between cameras: a large affinity score indicates that cameras allow for good reconstruction of a large number of targets, and should be paired first.

III. TREE DISTRIBUTED RECONSTRUCTION

The problem of reconstructing n targets from the measurements of m cameras is considered assuming each camera to be synchronized and provided of computational power. The aim of the study is to design a distributed reconstruction strategy to reduce the overall computational time and to this aim here a (strictly) binary tree algorithm is considered.

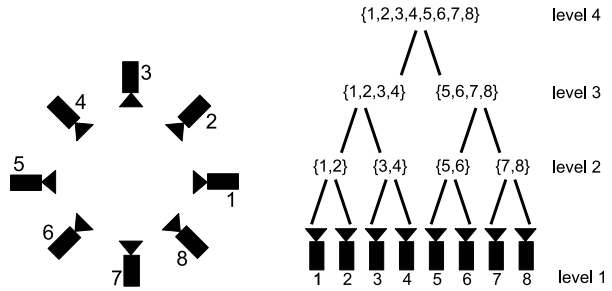
Be the cameras numbered from 1 to m (see Fig. 1(a)) and, without loss of generality, assume $m = 2^L$. Furthermore, let the groups $\{\mathcal{G}_{l,h}\}$ be defined as follows:

$$\begin{aligned}
\mathcal{G}_{1,h} &= \{h\}, \quad h = 1, \ldots, m, \\
\mathcal{G}_{2,h} &= \{2h-1, 2h\}, \quad h = 1, \ldots, m/2, \\
&\vdots \\
\mathcal{G}_{l,h} &= \{2^{l-1}(h-1)+1, \ldots, 2^{l-1}h\}, h = 1, \ldots, m/2^{l-1},
\end{aligned}$$

with $l \leq L$. A convenient way to represent the groups $\{\mathcal{G}_{l,h}\}$, is to use a tree graph as in Fig. 1(b), where each node of the tree corresponds to a group $\{\mathcal{G}_{l,h}\}$, and l is the level of such group. The algorithm starts elaborating the leaves of the tree (level 1) as described in the following, and then it iteratively moves up to the root.

First, the algorithm merges the information of single cameras forming $m/2$ groups of camera pairs: the h-th pair $\mathcal{G}_{2,h}$ tries to match pairs of measurements (the first taken from $\mathcal{G}_{1,2h-1}$, the second from $\mathcal{G}_{1,2h}$): when two measurements are compatible (i.e. their projection error is small [11]), they are assumed to be measurements of the same target and thus they form a new *crossing point* of order 2 (measurements from a single camera are considered crossing points of order 1). Similarly, when two crossing points of order k_1 and k_2 are compatible they form a new crossing point of order $k_1 + k_2$.

At level l the algorithm merges the information of $\mathcal{G}_{l,2h-1}$ and $\mathcal{G}_{l,2h}$, for $h = 1, \ldots, m/2^l$. When a crossing point of order $k \geq \bar{k}$ (where $\bar{k}$ is a design parameter) it is considered as a detected target. When a target is



(a) (b)

Fig. 1. (a) Example of camera scenario. (b) Binary tree scheme.

detected all its measurements (and the crossing points that involve them) are no longer considered in the algorithm next steps[2].

Computations at each node of the tree are assumed to be done by one of the cameras in the group associated to such node. While level l uses information elaborated by level $l-1$, the nodes of level l can make their computations in parallel. Assuming that nodes in a level are synchronized, then the time τ_l to compute level l is the maximum computational time among all the nodes in level l. So the overall computational time is given by the sum of the maximum loads for each level: $\sum_{l=1}^{L} \tau_l$ (sequential approach).

Alternatively, the algorithm can be pipelined: the network nodes are divided into L groups, each associated to a level of the tree (starting from level 2 to the root). In particular, each node of the tree corresponds to a camera (i.e. to a processing device), and each level of nodes computes a level of the algorithm corresponding to a different time instant. Then, the output of a level is passed as input to the following level of nodes. In this way the computational time can be significantly reduced: let $\tau_{\max}$ be the maximum computational complexity over all the levels, $\tau_{\max} = \max(\tau_1, \ldots, \tau_L)$, then the first output is available after $L\tau_{\max} \geq \sum_{l=1}^{L} \tau_l$, however the outputs at each of the following time instants are available with a period of $\tau_{\max}$, where for large networks $\tau_{\max}$ is typically much lower than $\sum_{l=1}^{L} \tau_l$.

IV. ALGORITHM OPTIMIZATION: AFFINITY SCORE AND N-CUT PARTITION

Then, to optimize the tree reconstruction scheme, a reconstruction order that minimizes the computational time need to be devised. However, if the system dimensions (m, n) are large, the determination of the best reconstruction order is impracticable and in order to determine a suboptimal optimization strategy, the following observations are due:

- experimentally, most of the computational time of the algorithm is spent comparing pairs of camera measurements (matches or crossing points of order 1);
- the computational power available in the tree decreases by a factor 2 at each unitary increment of l;
- the bottleneck of the computational complexity of the algorithm is typically close to the root (for $l \approx L$);
- the presence of unmatched measurements (i.e. not associated to any detected target) have a significative impact on the overall computational burden and it has to be limited.

While an increment of the data load to be processed at the last levels of the tree (close to the root) changes very much their computational times (because of the small number of devices used), such an increment in the first levels (close to the leaves) of the tree typically does not affect very much their computational time (because more computational power is available). Therefore: *To reduce the computational complexity of the algorithm (both in the sequential and the pipelined version), elaborate data so as to detect as many targets as possible at the first steps of the algorithm.*

[2]Computational time reduction with respect to the classical reconstruction scheme (where the centralized matching of measurements is concurrently made on all cameras) is mostly obtained thanks to the distribution of the computational burden over the net and to the removal of measurements of already detected targets from the reconstruction procedure.

Notice that $\mathcal{G}_{L,1}$ is formed by all the cameras, and is partitioned in two groups $\mathcal{G}_{L-1,1}$ and $\mathcal{G}_{L-1,2}$. Similarly, each group $\mathcal{G}_{l,h}$ at level l, $\forall l > 1$, is partitioned in two groups $\mathcal{G}_{l-1,2h-1}$, $\mathcal{G}_{l-1,2h}$. Also, an affinity score $\sigma^2_{r,\mathcal{G}_{l,h}}$ related to a generic set of cameras $\mathcal{G}_{l,h}$ is introduced, based on the mean reconstruction error variance of the targets in the visible part of the scene (note that this is a partial reconstruction) [16].

With no a priori information about the targets and their configuration, they are considered as (approximately) independent and identically distributed. Let $\hat{\phi}_i = (\hat{x}_i, \hat{y}_i, \hat{z}_i)^\top$ be the reconstructed position of target i, positioned in $\bar{\phi}_i = (\bar{x}_i, \bar{y}_i, \bar{z}_i)^\top$, where $i = 1, \ldots, n$. Then, assuming a probability density distribution of a target $p_\phi(\phi_i)$ in the scene volume V, the mean reconstruction variance σ^2_r is:

$$\begin{aligned}\sigma^2_r &\approx \int_V \mathbf{E}[(\hat{\phi}_i - \phi_i)^\top(\hat{\phi}_i - \phi_i) \mid \phi_i] p_\phi(\phi_i)\, d\phi_i \\ &\approx \frac{1}{n'}\sum_{i=1}^{n'} \mathbf{E}[(\hat{\phi}_i - \phi_i)^\top(\hat{\phi}_i - \phi_i) \mid \phi_i]\ ,\end{aligned}$$

where it is introduced a discrete approximation on voxels (with an abuse of notation indicated with $\{\phi_1, \ldots, \phi_{n'}\}$) whose positions are sampled from $p_\phi(\phi_i)$).

To take into account of the presence of occlusions, let the variable θ_{ij} be defined as follows: $\theta_{ij} = 1$ if target i is visible by camera j, and 0 otherwise. Let Θ_i be a random vector representing the values of the variables $\{\theta_{i1}, \ldots, \theta_{im}\}$. Then:

$$\sigma^2_r \approx \frac{1}{n'}\sum_{i=1}^{n'}\sum_{\Theta_i} \mathbf{E}[(\hat{\phi}_i - \phi_i)^\top(\hat{\phi}_i - \phi_i) \mid \Theta_i] p(\Theta_i|\phi_i)\ ,$$

where the second sum is varying the possible values of Θ_i.

In the case of *self-occlusions*, which are considered in this work, assume the object much larger than a marker and approximatively convex and let its orientation at point ϕ_i be identified by the unit vector orthogonal to the tangent plane in ϕ_i and going outwards the object. The target is mostly visible by cameras positioned approximatively on the direction pointed by the object orientation. Then, the curvature of the object can be well approximated by its tangent plane and thus a marker in ϕ_i is occluded by the object to cameras at more than $\pi/2$ angle with respect to its orientation, while it is visible by the others.

According to these assumptions, the values of $\{\theta_{ij}\}_j$ can be computed from the object orientation in ϕ_i (assuming cameras' orientations known). Let φ, ϑ (where $0 \leq \varphi \leq 2\pi$, $-\pi/2 \leq \vartheta \leq \pi/2$) be two angles identifying the target orientation. Then,

$$\sigma^2_r \approx \frac{1}{n'}\sum_{i=1}^{n'}\int_0^{2\pi}\int_{-\pi/2}^{\pi/2}\sum_{\Theta'_i} \mathbf{E}[(\hat{\phi}_i - \phi_i)^\top(\hat{\phi}_i - \phi_i) \mid \Theta_i] \cdot p(\Theta_i|\varphi,\vartheta)p(\varphi)p(\vartheta)d\varphi d\vartheta\ ,$$

where $p(\varphi) = \frac{1}{2\pi}$, $p(\vartheta) = \frac{1}{\pi}$; summing over a finite set of values of φ and ϑ the approximation holds:

$$\sigma^2_r \approx \sum_{i=1}^{n'}\sum_{h=1}^{H}\sum_{k=1}^{K} \frac{\mathbf{E}[(\hat{\phi}_i - \phi_i)^\top(\hat{\phi}_i - \phi_i) \mid \varphi_h, \vartheta_k]}{n'HK}, \tag{1}$$

where H and K are the number of samples $\{\varphi_h\}$ and $\{\vartheta_k\}$ used for φ and ϑ, respectively, and $\mathbf{E}[(\hat{\phi}_i - \phi_i)^\top(\hat{\phi}_i - \phi_i) \mid \varphi_h, \vartheta_k]$ refers to the reconstruction error variance [18].

When (1) is used to compute the mean reconstruction variance restricted to the region visible by the cameras in $\mathcal{G}_{l,h}$, namely $V_{\mathcal{G}_{l,h}}$, it is indicated as $\bar{\sigma}^2_{r,\mathcal{G}_{l,h}}$. Thus, being $\bar{\sigma}^2_{r,\mathcal{G}_{l,h}}$ the mean reconstruction error of the cameras in $\mathcal{G}_{l,h}$, it is related to the *affinity* between the cameras in $\mathcal{G}_{l,h}$: the smaller value of the mean reconstruction error $\bar{\sigma}^2_{r,\mathcal{G}_{l,h}}$, the higher is the affinity between the cameras in $\mathcal{G}_{l,h}$.

Hence, to maximize the number of targets detected at the first levels (with respect to the visible ones) we compute the partitioning of each group $\mathcal{G}_{l+1,h/2}$ as follows:

$$(\hat{\mathcal{G}}_{1,1}, \ldots, \hat{\mathcal{G}}_{L,1}) = \arg\min_{\mathcal{G}_{1,1},\ldots,\mathcal{G}_{L,1}} \sum_{l=1}^{L} \max_h \left(\bar{\sigma}^2_{r,\mathcal{G}_{l,h}}\right) \tag{2}$$

where $(\mathcal{G}_{1,1},\ldots,\mathcal{G}_{L,1})$ vary among all the admissible partitions, and $(\hat{\mathcal{G}}_{1,1},\ldots,\hat{\mathcal{G}}_{L,1})$ are the suboptimal ones. The rationale here is *to exploit the localization of data to make the reconstruction procedure faster*.

Unfortunately, though, the number of partitions to be checked in (2) increases exponentially with m, thus the direct minimization in (2) becomes quickly impracticable. Therefore, instead of minimizing the functional in (2), we formulate a similar problem that is computationally more advantageous and whose solution can be computed by distributing the computation over the network.

If both the mean reconstruction error between cameras j_1 and j_2, $\bar{\sigma}^2_{r,j_1j_2}$, is low (i.e. good reconstruction), and that between j_2 and j_3, $\bar{\sigma}^2_{r,j_2j_3}$, is low, then often also $\bar{\sigma}^2_{r,j_1j_3}$ and $\bar{\sigma}^2_{r,j_1j_2j_3}$, are low as well (although, there are singular camera-target configurations that may violate this consideration). Then, the mean reconstruction errors $\{\bar{\sigma}^2_{r,j_1j_2}\}_{j_1=1,\ldots,m,j_2=1,\ldots,m}$ are sufficient to provide a rough description of the reconstruction capabilities over the network.

In this context, the undirected graph is considered, where each node corresponds to a smart camera of the VSN and the weight of the edge between node (i.e. camera) j_1 and j_2 is $w(j_1,j_2)=\bar{\sigma}^2_{r,j_1j_2}$. Then, the graph is partitioned using a normalized cuts graph partitioning [23]: a cut in a graph has the effect of partitioning it in two parts A and B, and the value of the cut is given by the sum of the weights of the edges starting from A and ending at B. Differently from a standard cut, a normalized cut considers the value of the cut as a fraction of the total edge connections to all nodes in the graph. More specifically, the value of a normalized cut that separates the graph in two partitions A and B is defined as follows:

$$N_{cut}(A,B)=\frac{\sum_{j_1\in A,j_2\in B}w(j_1,j_2)}{\sum_{j_1\in A,\forall j_2}w(j_1,j_2)}+\frac{\sum_{j_1\in A,j_2\in B}w(j_1,j_2)}{\sum_{j_2\in B,\forall j_1}w(j_2,j_1)}\ .$$

With the aim of partitioning the graph in parts with low mutual reconstruction ability (i.e. high values of $\bar{\sigma}^2$ between two nodes, one taken from the first partition and one from the other), the goal here is that of finding the maximum normalized cut.

From [23], it immediately follows that the problem can be solved by computing the eigenvector x of $D^{-1/2}(D-W)D^{-1/2}$ corresponding to the maximum eigenvalue, where D is a diagonal matrix whose i-th element on the diagonal is $\sum_{j_2}w(j_1,j_2)$, while W is a symmetric matrix such that the element on the j_1-th row and j_2-th column is $w(j_1,j_2)$. Each entry of x corresponds to a node of the graph. Then the partition of the graph in two sets can be determined by checking if $x(j_1)$ is greater than a proper threshold $\bar{x}$, for each node j_1. Then, the normalized cut partition is repeated for each subset at each level of the tree until the tree is completely formed.

Interestingly, the eigenvector of interest can also be computed by means of the power iteration method [28], whose computational burden can be easily and conveniently distributed over the network (other linear algebra algorithms that can be formulated in a distributed way are considered in [26]). However, even considering the complete eigenvalue decomposition, its computational complexity is polynomial (approximately $O(m^3)$).

V. SIMULATIONS

In this Section the computational complexities of the centralized, of the non-optimized tree distributed, and of the optimized tree distributed algorithms are compared in some examples. In order to make the comparison indipendent on the specific used devices, the results are reported in terms of number of elementary mathematical and communication operations. For simplicity, the cost of each elementary mathematical and communication operation is assumed to be unitary, however, similar results can be obtained also for different values of such costs. In all the considered comparisons the sample computational complexities are computed as the mean of 100 independent reconstructions.

First, in Fig. 2 we compare the computational complexity of the proposed (non-optimized and non-pipelined) tree distributed algorithm with that of the centralized one. The comparison is done in a scenario characterized by clustered targets: the total number of targets n ranges from 64 to 1024 (arranged in 8 clusters), whereas the number of cameras is set to 128.

Fig. 3(a) compares the number of detected targets (percent) along the execution time of the tree reconstruction algorithm when using a random reconstruction order or the optimized one based on the affinity score. In this example $m=16$, $n=256$. Remarkably, a further beneficial effect of adopting the affinity based pairing order

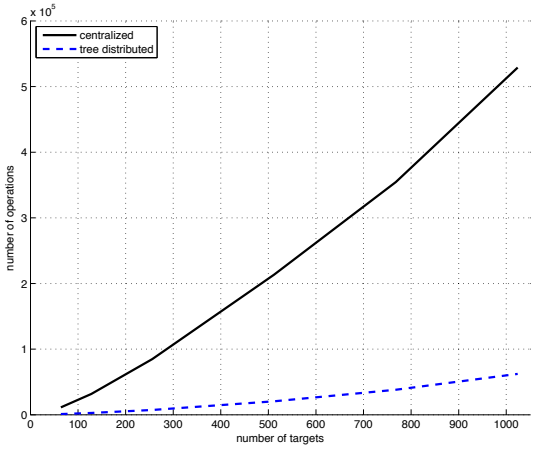
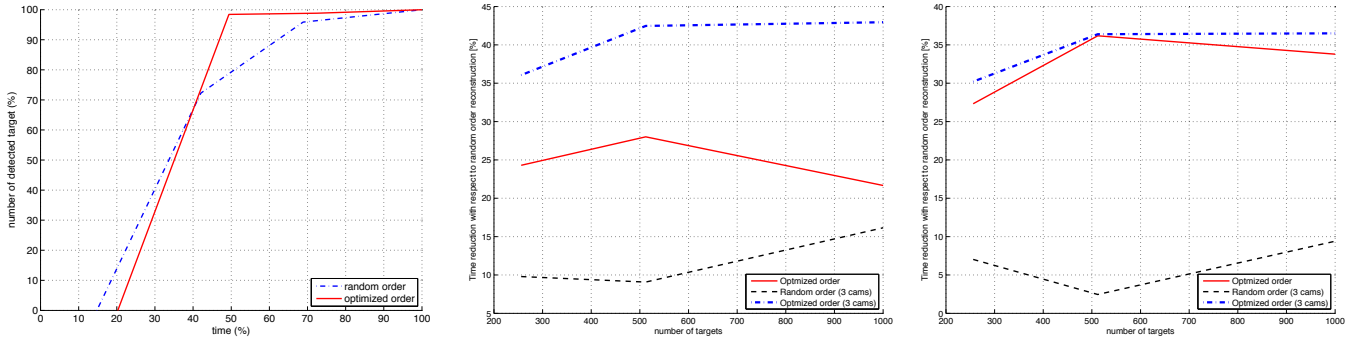


Fig. 2. Number of operations for reconstructing n targets using m cameras, where $m = 128$, while n varies from 64 to 1024. Comparison of the computational complexities of the centralized (black line), and the distributed algorithm (blue line dashed).

beyond the gain in the full reconstruction time appears from this plot: the reconstruction strategy based on the camera affinity allows to obtain a higher number of reconstructed targets earlier with respect to the random reconstruction order. In other words, the speed of *learning the scene* is strongly improved, in that it is possible to obtain an almost complete (rough) reconstruction at an early stage of the whole procedure.



(a) (b) (c)

Fig. 3. (a) Number of detected targets vs the computational time (normalized with respect to that of the optimized order) obtained using the (dashed-dotted blue line) random, and (red solid line) optimized reconstruction order. (b-c) Computational time reduction with respect to random reconstruction order with pairs at leaf level: sequential (b) and pipelined (c) distributed algorithm, respectively referring to $\sum_{l=1}^{L} \tau_l$ and $L\tau_{\max}$. Time reduction obtained by using: optimized reconstruction as in Sec. IV with pairs at leaf level (red solid line), random order with triplets at leaf level (black dashed line), optimized reconstruction as in Sec. IV with triplets at leaf level (blue dash-dotted line).

In the scenario of Fig. 3(b)-(c), we consider $m = 32$ cameras and a number of targets n ranging from 256 to 1024, placed in a room whose size are approximately $20 \times 20 \times 3$ m^3. In this case, a further modification to the tree-based reconstruction algorithm is considered: according to [19], we notice that considering groups of 3 cameras at the first step of the procedure allows to significantly reduce the number of artifact targets, and consequently, it typically allows to reduce the computational time. Hence, also a modified version of the tree structure of Sec. III is considered, where at leaf level camera triplets are used instead of pairs. Fig. 3 shows the (percent) computational time reduction obtained by employing the reconstruction algorithm with the following reconstruction orders[3]:

- random order employing triplets at the leaf level;

[3]Since cameras are positioned without a specific topological order, the computational complexity is evaluated with respect to a random reconstruction order with pairs at leaf level (that changes for each reconstruction).

- order determined by the optimization of Sec. IV;
- order determined by the optimization of Sec. IV employing triplets at the leaf level.

As shown in Fig. 3, the optimization of the reconstruction order leads to a significant computational time reduction with respect to the use of a random reconstruction order. However, it is worth to notice that, in agreement with the approach in [19], the best results are obtained by applying the optimized order to the modified reconstruction scheme where the leaf level is given by camera triplets, with respect to the binary tree of Sec. III. Nonetheless, the performance difference between the binary tree scheme and the triplet based one is less clear in the pipelined case.

Finally, in accordance with the improved use of computational power, our study shows that the proposed tree distributed algorithm prove to be much faster than the centralized one. Let the gain γ be the factor between the computational time of the centralized algorithm versus the time of the (non-pipelined and non-optimized) distributed algorithm. Then, γ increases approximately linearly as m becomes larger (more computational devices are used in the network), however it is always strictly lower than m (which is an upper bound to the value of the gain factor).

VI. CONCLUSIONS

This paper considers the problem of distributed 3D reconstruction in a large VSNs, such as those constituted by large scale MoCap systems. In particular, a tree distributed reconstruction scheme is proposed, which results to be much faster with respect to the non-distributed algorithm.

Furthermore, a reconstruction order optimization method that can be efficiently computed has been proposed: this is based on the computation of an affinity score between groups of cameras that allows targets to be reconstructed first. Differently from other data-driven procedures, this affinity functional is based on a probabilistic approach.

The use of the optimized reconstruction order can significantly reduce the computational time of the reconstruction algorithm with respect to the use of a random reconstruction order. Finally, a further computational time reduction can be obtained by using a tree reconstruction scheme where the leaf level is composed by groups of three cameras (instead of a fully binary tree with leaf level of pairs).

Future works include a better characterization of the reconstruction errors and the performance quantification, also with respect to experimental setups and data and in comparison with other techniques.

REFERENCES

[1] A.A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30:2826–2841, 2007.

[2] H. Aghajan and A. Cavallaro. *Multi-Camera Networks, Principles and Applications*. Academic Press, 2009.

[3] E. Borovikov and L. Davis. A distributed system for real-time volume reconstruction. In *Proc. of the 5th IEEE Int. Workshop on Computer Architectures for Machine Perception*, pages 183–189, In CAMP, 2000.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, June 2006.

[5] J. Falcou, J. Sérot, T. Chateau, and F. Jurie. A parallel implementation of a 3D reconstruction algorithm for real-time vision. In *Parallel Computing: Current & Future Issues of High-End Computing, Proc. of the Int. Conf. ParCo 2005*, pages 663–670, In ParCo, 2005.

[6] J.S. Franco, C. Ménier, E. Boyer, and B. Raffin. A distributed approach for real-time 3d modeling. In *Proc. of the 2004 Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW04)*, 2004.

[7] Y. Furukawa, B. Curless, S.M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Proc. of the 23rd IEEE Conf. on Computer Vision and Pattern Recognition (CVPR10)*, pages 1434–1441, 2010.

[8] F. Garin and L. Schenato. A survey on distributed estimation and control applications using linear consensus algorithms. *Networked Control Systems, Springer Lecture Notes in Control and Information Sciences*, 406:75–107, 2011.

[9] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S.M. Seitz. Multi-view stereo for community photo collections. In *Proc. of the 11th IEEE Int. Conf. on Computer Vision (ICCV07)*, 2007.

[10] R.I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[11] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[12] Young Min Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun. Multi-view image and ToF sensor fusion for dense 3D reconstruction. In *IEEE 12th Int. Conf. on Computer Vision Workshops (ICCV09 Workshops)*, pages 1542–1549, 2009.

[13] A. Klausner, A. Tengg, and B. Rinner. Distributed multi-level data fusion for networked embedded systems. *J. on Selected Topics in Signal Processing*, 2(3):538–555, 2008.

[14] W. Li and H. Dai. Cluster-based distributed consensus. *IEEE Trans. on Wireless Communications*, 8(1):28–31, 2009.
[15] S. Liu, K. Kang, J.-P. Tarel, and D.B. Cooper. Distributed volumetric scene geometry reconstruction with a network of distributed smart cameras. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR09)*, pages 2334–2341, 2009.
[16] A. Masiero and A. Cenedese. A Kalman filter approach for the synchronization of motion capture systems. In *Proc. of the 51st IEEE Conf. on Decision and Control (CDC12)*, pages 2028–2033, Maui, Hawaii, USA, December 2012.
[17] A. Masiero and A. Cenedese. On triangulation algorithms in large scale camera network systems. In *Proc. of the 2012 American Control Conf. (ACC12)*, pages 4096–4101, Montréal, Canada, June 2012.
[18] A. Masiero and A. Cenedese. Reconstruction error in a motion capture system. In *arXiv*, 2012.
[19] D. Nister. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *Proc. of the European Conf. on Computer Vision (ECCV00)*, pages 649–663, 2000.
[20] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. on Automatic Control*, 49:1520–1533, 2004.
[21] B. Rinner and W. Wolf. An introduction to distributed smart cameras. *Proc. of the IEEE*, 96(10):1565–1575, 2008.
[22] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of the 22nd IEEE Conf. on Computer Vision and Pattern Recognition (CVPR06)*, 2006.
[23] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.
[24] S. Soro and W. Heinzelman. A survey of visual sensor networks. *Advances in Multimedia*, 2009:1–22, 2009.
[25] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proc. of the Int. Workshop on Vision Algorithms: Theory and Practice, Springer Lecture Notes on Computer Science*, pages 298–372, 1999.
[26] R. Tron and R. Vidal. Distributed computer vision algorithms through distributed averaging. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR11)*, pages 57–63, 2011.
[27] K.I. Tsianos and M.G. Rabbat. Fast decentralized averaging via multi-scale gossip. In *Proc. of the 6th IEEE Int. Conf. on Distributed Computing in Sensor Systems*, DCOSS'10, pages 320–333. Springer-Verlag, 2010.
[28] R. von Mises and H. Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM - Zeitschrift fr Angewandte Mathematik und Mechanik*, 9:152–164, 1929.
[29] P.-B. Wieber, F. Billet, L. Boissieux, and R. Pissard-Gibollet. The HuMAnS toolbox, a homogenous framework for motion capture, analysis and simulation. In *Int. Symposium on the 3D Analysis of Human Movement*, Valenciennes, France, 2006.
[30] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. In *Proc. 42nd IEEE Conf. on Decision and Control (CDC03)*, volume 5, pages 4997–5002, 2003.
[31] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proc. of the 4th Int. Conf. on Information Processing in Sensor Networks*, pages 63–70, 2005.
[32] O. Younis, M. Krunz, and S. Ramasubramanian. Node clustering in wireless sensor networks: recent developments and deployment challenges. *Network, IEEE*, 20(3):20–25, 2006.