Università degli studi di Padova
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria dell'Automazione

Distributed Localization of a Camera Network in SE(3)

Relatore
Prof. Angelo Cenedese

Correlatore
Prof. Simone Milani

Laureanda
Giulia Michieletto
matr. 1056309

Anno Accademico 2013/2014

Non c'è piacere nel successo,
se non lo dividi con qualcuno.
I fantastici 4

Contents

List of Figures

List of Tables

Abstract

In last decades, sensor networks have established themselves among the priority targets of research in the field of control theory. Indeed, several algorithms concerning these structures have been recently developed.

Camera systems are probably the most common type of sensor networks. They constitute a rich information sensing modality whose potential applications are numerous in civil, industrial and military context. However, their effectiveness greatly relies on the ability of each device to calibrate itself with respect to the other ones in the network (*relative pose*) and to a fixed absolute reference system (*absolute pose*). For this reason, the localization problem, i.e. the determination of the absolute rotation matrix and translation vector of each camera, still engenders great attention.

Main objective of this work is to tackle the aforementioned problem analyzing and improving the algorithm developed by Tron and Vidal [1]. The authors consider the estimation of the poses for a camera network and they converge to an optimal solution starting from noisy measurements of the relative poses. The method proposed is an iterative procedure based on the minimization of a suitable cost functional through a distributed strategy. The basic idea to reach the solution is to use the framework of consensus algorithms, but alternating a Riemannian gradient descent on the space of rotations and a Euclidean gradient descent on the space of translations.

The performance of the algorithm proposed in [1] is evaluated using synthetic and real data. More specifically, the analysis focuses on the examination of the results in terms of reliability and optimality of the solution found.

Abstract

Negli ultimi decenni, le reti di sensori si sono affermate tra i principali oggetti di ricerca nel campo della teoria del controllo. Diversi algoritmi riguardanti queste strutture sono stati recentemente sviluppati.

I sistemi di videocamere sono probabilmente il più comune tipo di reti di sensori. Essi costituiscono una ricca modalità di rilevamento dati le cui potenziali applicazioni sono numerose in ambito civile, industriale e militare. La loro efficacia, tuttavia, dipende in gran parte dalla capacità di ciascun dispositivo di calibrare se stesso rispetto agli altri elementi della rete (*posa relativa*) e ad un sistema di riferimento assoluto e fisso (*posa assoluta*). Per questo motivo, desta ancora molto interesse il problema della localizzazione, consistente nella determinazione della matrice di rotazione e del vettore di traslazione assoluti di ogni videocamera.

L'obiettivo principale di questo lavoro è quello di affrontare la suddetta questione analizzando e migliorando l'algoritmo sviluppato da Tron e Vidal [1]. Gli autori considerano il problema della stima delle pose per una rete di videocamere e convergono ad una soluzione ottimale a partire da misure rumorose delle pose relative. Il metodo proposto è una procedura iterativa basata sulla minimizzazione di un adeguato funzionale costo mediante una strategia distribuita. L'idea di base per raggiungere la soluzione è quella di utilizzare il framework degli algoritmi di consenso, ma alternando una discesa gradiente Riemanniana sullo spazio delle rotazioni e una discesa gradiente Euclidea sullo spazio delle traslazioni.

Le prestazioni dell'algoritmo proposto in [1] sono valutate utilizzando dei dati sintetici e reali. Nello specifico, l'analisi si concentra sull'esame dei risultati in termini di affidabilità e ottimalità della soluzione trovata.

1

Introduction

In 20th century sensor networks have established themselves among the priority targets of research in the field of control theory. These systems are usually composed by multiple interactive elements, named agents or nodes, placed throughout an environment of interest. Each of these devices can exchange information with its neighbors or with the whole network using various communication protocols (e.g. synchronous, broadcast, symmetric gossip, asymmetric gossip protocols in both their deterministic and randomized version) in order to achieve a common goal.

One of the principal aim of a multi-agents systems is to monitor phenomena of interest, to detect events and to control the environment according to the changes of information gained by every node. The strength of this type of systems resides in the interaction among agents which yields the emergence of complex collective behaviors enacted via simple local rules.

Therefore, sensor networks have recently emerged as a dominant technology in a wide range of application fields, from the civil and military context to the social and biological one. It is worth of notice that, even if the devices used in various applications can be quite different, all multi-agents systems share some common features, as scalability and cooperation. To model and design these properties, a distributed approach is required in place of a centralized one. Indeed, the presence of a central unit that is linked with all sensors in the network entails an easy design for the topology but a more rigid approach to tackle scalability issues, while the choice of a distributed paradigm can avoid problems concerning scalability and robustness to failures of both nodes and network. This allows the reduction of communication time favoring the goal achievement more quickly [2].

Contents

1.1 Camera systems

Camera systems are probably the most common type of sensor networks. They can be defined as large collections of well-coordinated cameras. Specifically, these systems might include devices of different types (RGB cameras, Time-of-Flight or structured light depth sensors) and different motion capabilities (fixed lens or pan-tilt-zoom /PTZ).

A fixed-lens camera is a low-cost camera with a fixed field of view. A PTZ device is capable of remotely controlling its own direction and zoom once installed, it can rotate, tilt and run enlargements or reductions of observed area or object. A RGB camera is the vision device traditionally used for the acquisition of color images. It acquires the three basic color components (red, green and blue) on three different wires, thereafter the signals are combined in order to assign a color to each pixel of the image.

Stereo vision systems have been widely refined in last years bringing satisfactory results, however they can not manage all situations; in particular, they are not able to provide a reliable depth information whenever visual cues can be scarcely found (flat untextured regions).

For this reason, a new type of systems has recently emerged in computer vision where high quality depth information is required. These are based on heterogeneous architectures, made of standard cameras and ToF sensors [3]. These last ones determine distances measuring the traveling time of an infrared impulse from the emitter of the ToF device to the object and back. The fusion of ToF and stereo data allows to obtain an improved 3D geometry as this solution combines the best features of both subsystems, such as high resolution, elevated accuracy and robustness with respect to scene peculiarities.

Technological progress in sensor design, computing and communications are leading to the development of new applications that will transform traditional vision systems into pervading smart camera networks. This technology has become ubiquitous in a variety of real-world applications including security and surveillance of industrial or civil areas, perimeter patrolling, intrusion detection and tracking, vehicle tracking, mapping and reconstruction of environments (e.g. search and rescue in critic scenarios, dynamic 3D reconstruction).

However, the critical issues related to the camera systems are several. Typical problems to solve are localization/calibration of the devices, data registration, alignment and association, high accuracy images acquisition, structure from motion (SfM). For many of them, interesting solutions have been proposed and tested obtaining satisfactory results; nevertheless the research of new strategies continues in order to improve the system performances.

This work addresses the localization problem, i.e. the determination of the position and orientation in the three-dimensional space of each device, as it represents the critical first step in any camera network deployment [1].

[1] Generally, the wording *localization problem* refers to the determination of the agent positions of a sensor network in the environment. Referring to camera system, the expression *calibration/registration problem* is more precise to indicate the issue of computing the rotation matrix and translation vector for each device of the network. Nevertheless, in this work the term *localization problem* is always used.

1.2 State of the art about localization problem

Manually measuring the pose of all cameras in the network and calibrating the whole system is a very tedious and time consuming task. Moreover, when scaling with the number of cameras or the complexity of the environment, this activity may become extremely difficult and practically unfeasible.

For these reasons a wide research work has been carried out on methods to solve calibration task automatically by exploiting information supplied by the agents. Various approaches have been adopted according to the assumptions made about the dimension of the ambient space (planar versus tridimensional), type of algorithm (centralized versus distributed), kind of measurements (e.g. distances, angles of arrival) and special assumptions about the environment and the deployment of the cameras.

In some cases, special nodes (called *beacons* or *anchors*), whose positions are known, are introduced in the network. Many algorithms tackle the localization problem exploiting the knowledge of the distances between all nodes of the networks and the anchors. In particular, Aspnes et al. [4] have provided a theoretical foundation of the issue indicating conditions for uniqueness in localization of networks with beacons and distance measurements.

As far as distance information exploitation is concerned, Mantzel et al. [5] have proposed an iterative technique for estimating the camera network positions and orientations based on the well-known concept of triangulation. Their Distributed Alternating Localization-Triangulation (DALT) algorithm is easily distributable imposing that each camera localizes itself using triangulated points and then collaborates with other nearby devices to triangulate other common points.

Nonetheless, other types of information can be used to solve the aforementioned problem. Inspired by the work of Rong and Sichitiu [6] and based on some key kinematic relationships, Bullo et al. [7] have proposed a localization and orientation approach based on angle-of-arrival sensing between neighboring nodes in the 2D case. Moreover, the authors have also explored necessary and sufficient conditions for a noiseless network to be calibrated in the 3D case. The main assumption of this work is that nodes do not have the ability to perform measurements expressed in a common reference frame.

Another interesting distributed solution to the localization problem has been proposed by Devarajan et al. [8]. Their work rests on a set of uncalibrated cameras in which communication occurs only among cameras that share on their image planes some of the scene points. Initially each camera independently forms a neighborhood cluster on which the local calibration takes place. Then the algorithm proceeds incrementally: calibrated nodes and scene points are incrementally merged into a common coordinate frame.

Finally, if a dynamic environment is considered, Funiak et al. [9] have demonstrated that a camera network can be automatically calibrated by tracking a moving object. Their main idea is to address the camera network calibration task by solving a simultaneous localization and tracking (SLAT) problem, where both the trajectory of the object and the poses of the cameras are estimated.

1.3 Contributions and structure of the thesis

In this thesis, a distributed method for automatically solving the calibration task is presented. Although traditional large scale systems are characterized by a centralized architecture, the problem is here addressed in a distributed manner by assuming that any camera in the network has its own reference frame and does not have any knowledge about the position of the other ones.

More specifically, the problem of reconstruction of the absolute poses is solved starting from the solution proposed by Tron and Vidal [1].

Their algorithm is based on the minimization of a suitable cost functional on $SE(3)^N$, i.e. the group of rigid body transformations of N agents, through a distributed strategy. It is an iterative procedure characterized by the alternation between Euclidean and Riemannian gradient descents.

In detail, Tron and Vidal represent the pose of each camera i as a couple $g_i = (R_i, T_i) \in SE(3)$ consisting in a rotation matrix and a translation vector, and the relative change of coordinates from device i to device j as $g_{ij} = g_i^{-1} \circ g_j = (R_{ij}, T_{ij}) \in SE(3)$, where $R_{ij} = R_i^T R_j$ and $T_{ij} = R_i^T(T_j - T_i)$. Hence the absolute pose of one camera can be obtained from the pose of another one applying the composition operation $g_j = g_i \circ g_{ij}$.

Starting from these definitions, a network is said to be *localized* if there is a set of relative transformations $\{(R_{ij}, T_{ij})\}$ such that, when the reference frame of the first node is fixed to (R_1, T_1), the other absolute poses (R_i, T_i) are uniquely determined. Therefore, the aim is to compute a set of relative transformations $\{g_{ij}\}$ that satisfy the proprieties stated and that, at the same time, are as close as possible to the relative noisy measurements $\tilde{g}_{ij}$ required as inputs by the algorithm.

The strategy proposed by Tron and Vidal is set up on least squares minimization criterion. Since the quantities considered are non-Euclidean, it exploits a measure in $SE(3)$. Given a network of N nodes on edge set $\mathcal{E}$, a functional cost is introduced as

$$\begin{aligned}\varphi(\{R_i\},\{T_i\},\{\lambda_{ij}\}) &= \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} d_g^2(g_i^{-1}g_j, \tilde{g}_{ij}) \\ &= \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} (d_{SO(3)}^2(R_i^T R_j, \tilde{R}_i j) + ||R_i^T(T_j - T_i) - \lambda_{ij}\tilde{t}_{ij}||^2) \\ &= \varphi_R(\{R_i\}) + \varphi_T(\{R_i\},\{T_i\},\{\lambda_{ij}\}).\end{aligned}$$

In words, the cost φ is the sum of two terms: φ_R involving only the rotational part R_i of the absolute pose g_i and φ_T involving all the variables, including the unknown scale factors λ_{ij} that are responsible for an undesired side-effect and have to be positive.

In summary, an optimal localization is found by solving the non-linear program

$$\begin{aligned}&\min_{\{R_i\},\{T_i\},\{\lambda_{ij}\}} \varphi(\{R_i\},\{T_i\},\{\lambda_{ij}\}) \\ &\text{subject to} \quad \lambda_{ij} \geq 1 \quad \forall (i,j) \in \mathcal{E}\end{aligned}$$

The basic idea to reach the solution is to use the framework of consensus algorithms, but alternating a Riemannian gradient descent on the space of rotations (φ_R minimization) and a Euclidean gradient descent (φ_T minimization).

The scenario is indeed particularly challenging when dealing with 2 or 3 degrees of freedom in the space of rotations (e.g. pan-tilt or full pan-tilt-zoom relative displacements) since the optimization functional presents multiple local minima; conversely, when reducing to a 1 degree of freedom problem (e.g. pan movement only) the distributed algorithm converge to the centralized solution that can be found analytically.

The aim of this work is to observe and improve the results achieved implementing the Tron-Vidal algorithm on an uncalibrated camera system. For this reason, new initialization strategies are proposed in order to overcome the disadvantages of the original procedure. Moreover, the performance of the algorithm is evaluated in particular situations considering the goodness of the solution reached in terms of errors with respect to the real poses. The analysis is conducted first considering a synthetic setups and then testing the results on a real camera network made up of PTZ and fixed devices. It is important to point out that much of the work carried out is an extension of what has been studied by M. Michelan, G. Baggio and S. Patron [10].

After this introductory chapter, the thesis is organized in six chapters. Chapter 2 is devoted to the introduction of some mathematical notions useful to better understand the proposed strategy. In Chapter 3, the localization problem is formulated in a formal manner, and the two possible approaches known in the literature to tackle it are illustrated. Chapter 4 is dedicated to the complete description of the algorithm whose performances are analyzed in the following chapters; particular attention is given to the possible initialization strategies. Chapter 5 constitutes an aside since it presents another algorithm. It illustrates the original calibration method proposed by Cortelazzo et al. [11] and based on the frequency approach. The decision to include this dissertation is justified by the fact that the initial idea was to mix the Tron-Vidal and Cortelazzo procedures in order to get results very close to the real poses, however it has not been possible to pursue this idea because of the problems about the code in frequency domain. In Chapter 6, a deeper analysis is made about the implementation of the Tron-Vidal algorithm. Finally, Chapter 7 summarizes the main conclusions reached at the end of all the work done proposing several ideas to develop in the future.

In conclusion, it is necessary to underline the presence of two appendices: one dedicated to the presentation of the pinhole camera model and one describing some formalisms available in the literature for the representation of rotations in three-dimensional space.

2

Mathematical preliminaries

This chapter is dedicated to the introduction of some useful notions to better understand the task to be solved, i.e. the localization problem. First, some definitions concerning graph theory are given as a necessary tool to model the issues related to sensor networks. Then, the attention is focused on the camera kinematics, recalling the main properties of matrix groups $SO(3)$ and $SE(3)$ which correspond to the spaces of rotations and rigid body transformations respectively. Finally, a few calibration algorithms are presented. These techniques are known in the literature for the determination of the extrinsic and intrinsic parameters of a camera, typically relying on the knowledge of the coordinates of some points of the scene before and after the projection on the image plane. In our setup these are taken as the first step to retrieve calibration/localization information from the image planes.

Contents

2.1 Elements of graph theory

A multi-agents system can be represented with a network made up of N nodes organized into an undirected graph $\mathcal{G}$.

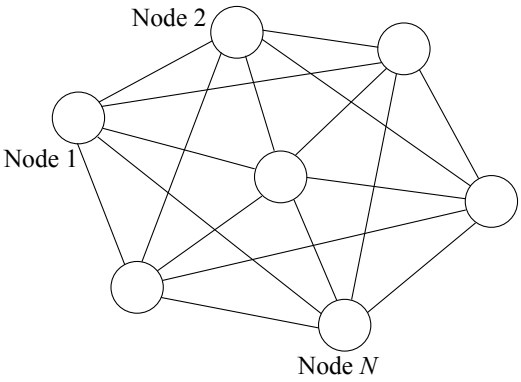


Figure 2.1.: Graph based network example

A *graph* $\mathcal{G}$ is an ordered pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. More specifically, the vertices of the graph correspond to the nodes of the network, while the pair (i, j) belongs to the set $\mathcal{E}$ if and only if the node i can communicate with the node j and viceversa. A graph which shows this symmetry property in the communication protocol among its nodes is referred as *undirected* or *non-oriented*: the pairs of vertices that represent the edges are not ordered. Otherwise, the graph is called *directed* or *oriented*. Moreover, a graph is named *complete* if each pair of vertices has an edge that connects them.

In mathematics and computer science, there exist different manners to represent a graph made up of N nodes. The most used method is its *adjacency matrix*: $A \in \mathbb{R}^{N \times N}$ such that

$$[A]_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that the adjacency matrix is symmetric in those cases where the associated graph is indirect. On the contrary if the graph is directed the matrix does not enjoy this property. Moreover, the elements on the main diagonal of A are equal to one only in the case where the graph presents *self loops*, i.e. there are edges that connect a vertex to itself.

Concerning nodes, it is important to define the *neighbors* of a node i as the set N_i of vertices which are connected with i in the graph:

$$N_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}. \tag{2.1}$$

In particular, N_i is a subset of $\mathcal{V}$ and its cardinality coincides with the *degree* of the node i, $\deg(i)$, i.e. the number of edges outgoing from node i.

Finally, other worthwhile elements of graph theory are the following ones:

- A *path* l from node i to node j in $\mathcal{G}$ is defined as a sequence of nodes starting with i and ending in j such that there exists an edge in $\mathcal{E}$ between consecutive

nodes. Formally, $l = \{v_1, \ldots, v_n\}$, $v_m \in \mathcal{V}$, $v_1 = i$, $v_n = j$, $(v_m, v_{m+1}) \in \mathcal{E}$, $m = \{1, \ldots, n-1\}$ [1].

A path is referred as *simple* if all the vertices of the path are distinct.

A graph is defined as *connected* if there is a path from any node to any other node in the graph.

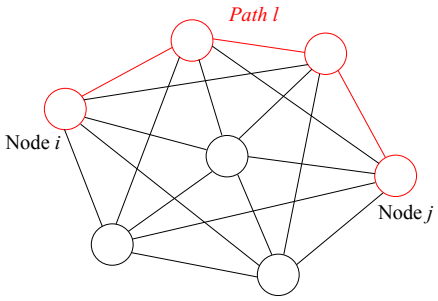


Figure 2.2.: Path example

- A *cycle* is a path from node i to itself without repeated nodes, except for the initial one and the final one [1].

  A cycle is referred as *simple* if all the nodes composing it are distinct (except for the initial one and the final one).

  An *acyclic* graph is a graph with no cycles.

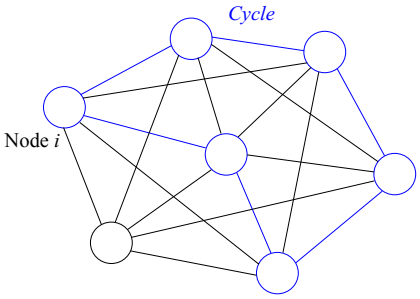


Figure 2.3.: Cycle example

- A *subgraph* of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph whose vertex set is a subset of that of $\mathcal{G}$, and whose adjacency relation is a subset of that of $\mathcal{G}$ restricted to this subset. In other words, it is a graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ such that $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$.

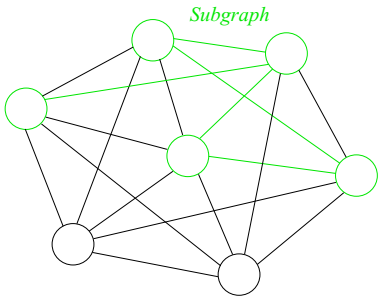


Figure 2.4.: Subgraph example

- A *tree* is a connected and acyclic graph.

  A *spanning tree* of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the subgraph such that it is a tree and $\mathcal{V}' = \mathcal{V}$, i.e. it includes all nodes of $\mathcal{G}$.

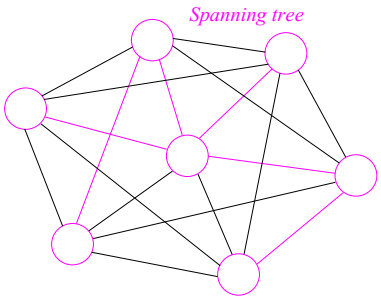


Figure 2.5.: Spanning tree example

2.2 Elements of camera kinematics

The position and orientation of each device in a camera system can be expressed with respect to others in the network (*relative pose*) and/or to a fixed absolute reference system (*absolute pose*).

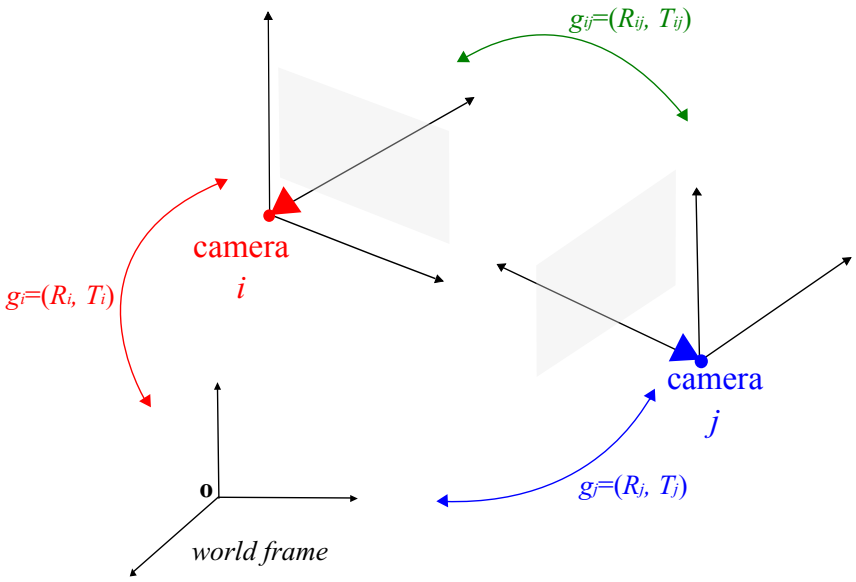


Figure 2.6.: Relationship between relative and absolute poses

The absolute pose is generally represented as the ordered couple $g = (R, T)$, where R is a matrix that indicates the rotation and T is a vector which refers to the translation. In particular, both R and T are expressed with reference to the same fixed world frame.

It is important to underline that the rotation matrix R belongs to the *special orthogonal group* which is defined as

$$SO(3) = \left\{ R \in \mathbb{R}^{3\times 3} : R^T R = I,\ det(R) = +1 \right\}. \tag{2.2}$$

More specifically, $SO(3)$ is a subgroup of *orthogonal group* $O(3)$ that is the group of matrices in which the inner product is preserved or, equivalently, it is the group of 3×3 orthogonal matrices.

In addiction, the translation vector T is simply set in $\mathbb{R}^3$.

Therefore, the camera pose g is an element of the group of rigid body transformations that is called the *special Euclidean group* and is defined as

$$SE(3) = \left\{ g = (R, T) : R \in SO(3),\ T \in \mathbb{R}^3 \right\}. \tag{2.3}$$

It is worth of notice that both $SE(3)$ and $SO(3)$ are *Lie groups*, i.e. algebraic groups characterized by a unique identity element and two group operations (multiplication and inversion) [10].

The absolute pose of a camera within a network can be related to the poses of the other network devices, using the relative poses expressions.

More formally, the absolute pose $g_i = (R_i, T_i)$ of camera i is linked to the absolute pose $g_j = (R_j, T_j)$ of camera j, its neighbor in the network, by the operation of composition $g_i \circ g_j = (R_i R_j, R_i T_j + T_i)$. As a consequence, it is possible to define the relative change of coordinates $g_{ij} = (R_{ij}, T_{ij}) = g_i^{-1} \circ g_j \in SE(3)$, where

$$R_{ij} = R_i^T R_j, \tag{2.4}$$
$$T_{ij} = R_i^T (T_j - T_i). \tag{2.5}$$

Hence, the absolute pose of camera j can be obtained from the pose of the i-th one as $g_j = g_i \circ g_{ij}$; more explicitly:

$$R_j = R_i R_{ij}, \tag{2.6}$$
$$T_j = R_i T_{ij} + T_i. \tag{2.7}$$

It is interesting to note that the relative transformations are invariant to the choice of a global reference frame.

2.2.1 Geometry of $SO(3)$

As just illustrated, a rotation in tridimensional space can be represented by a matrix which belongs to the *special orthogonal group* $SO(3)$, defined in Equation (2.2).

Given any $R \in SO(3)$, it is possible to identified the *tangent space at R*. It is expressed as

$$T_R(SO(3)) = \{ R\hat{\mathbf{v}} : \hat{\mathbf{v}} \in \mathfrak{so}(3) \}, \tag{2.8}$$

where $\mathfrak{so}(3)$ is the Lie algebra [1] of $SO(3)$, i.e. the space of skew-symmetric matrices

$$\mathfrak{so}(3) = \left\{ \hat{\mathbf{v}} \in \mathbb{R}^{3\times 3} : \hat{\mathbf{v}} = -\hat{\mathbf{v}}^T \right\}, \tag{2.9}$$

where $\hat{\mathbf{v}}$ denotes the matrix generating the cross product by $\mathbf{v} \in \mathbb{R}^3$, in other words $\hat{\mathbf{v}}\mathbf{u} = \mathbf{v} \times \mathbf{u}\ \forall \mathbf{u} \in \mathbb{R}^3$ [1].

[1] A *Lie algebra* is a vector space $\mathfrak{g}$ over a field F with an operation (Lie bracket) $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ which is bilinear and skew symmetric and satisfies the Jacobi identity.

A possible choice of metric in rotations group is given by the *Riemannian metric*

$$\langle R_i, R_j \rangle = \text{trace}\left\{R_i^T R_j\right\}, \quad R_i, R_j \in T_R(SO(3)). \tag{2.10}$$

The Riemannian metric is a continuous collection of inner products in the tangent space at each point [1] and it can be employed to measure the length of curves between two points of the group $SO(3)$. Curves with minimum length are referred as *geodesics*, while their lengths define the *Riemannian or geodesic distance* between the two points:

$$\begin{aligned} d_R^2(R_i, R_j) &= \frac{1}{2}\|\log(R_i^T R_j)\|_F^2 \\ &= -\frac{1}{2}\text{trace}\left\{\log(R_i^T R_j)^2\right\}, \quad R_i, R_j \in SO(3), \end{aligned} \tag{2.11}$$

where $\log(\cdot)$ is the matrix logarithm and $\|\cdot\|_F$ denotes the Frobenius norm [2].

Another possible choice of metric in $SO(3)$ is the *Frobenius or chordal metric*, through which the distance between two points is given by

$$d_F^2(R_i, R_j) = \|R_i - R_j\|_F^2, \quad R_i, R_j \in SO(3). \tag{2.12}$$

If the considered points are sufficiently close to each other then the chordal distance is similar to the geodesic one [10].

In an optimization procedure on $SO(3)$, two essential concepts are the *exponential map* and the *logarithm map* at a point in special orthogonal group.

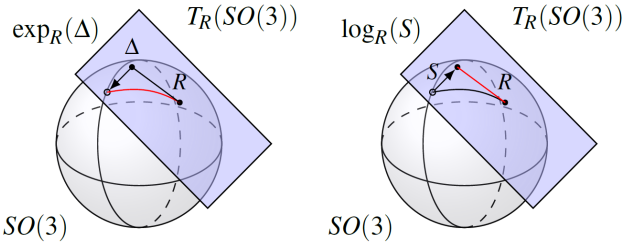


Figure 2.7.: Graphical representations of the action of exponential and logarithm maps on $SO(3)$

The exponential map at a point $R \in SO(3)$, $\exp_R(\Delta) : T_R(SO(3)) \to SO(3)$, is a diffeomorphism that associates to each point Δ in a neighborhood of the origin of the tangent space $T_R(SO(3))$ a point S on the (unique) geodesic passing through R in the direction Δ. The logarithm map $\log_R(S) : SO(3) \to T_R(SO(3))$ is the inverse of the exponential map. As consequence, the parallel transport method can be used to express these mappings at a generic point of rotations group:

[2] The *Frobenius norm* is the matrix norm of an $m \times n$ matrix A defined as the square root of the sum of the absolute squares of its elements, i.e. $\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$.

$$\exp_R(\Delta) = R\exp(R^T\Delta), \tag{2.13}$$

$$\log_R(S) = R\log(R^T S). \tag{2.14}$$

An intuitive idea of the actions of the exponential map and logarithmic map is illustrated in Figure 2.7.

The definition of the updates in an iterative optimization algorithm acting on $SO(3)$ is based on the calculation of the gradient following a gradient descent procedure.

Given a smooth function $f : SO(3) \to \mathbb{R}$, the *Riemannian gradient* $\text{grad}_R f$ is defined as

$$\text{grad}_R f = f_R - R f_R^T R, \tag{2.15}$$

where $f_R = \frac{\partial f}{\partial R}$ is the Euclidean derivative of the function f with respect to the matrix R. The Riemannian gradient can be rewritten considering the geodesic distance between two points $R_i, R_j \in SO(3)$:

$$\text{grad}_{R_i} d_R^2(R_i, R_j) = -R_i \log(R_i^T R_j) \ \in T_R(SO(3)). \tag{2.16}$$

2.2.2 Geometry of $SE(3)$

The pose of a camera includes not only the rotation matrix $R \in SO(3)$ but also the translation vector $T \in \mathbb{R}^3$; therefore it is defined in the group of rigid body transformations $SE(3)$ as in Equation (2.3).

The *special Euclidean group* $SE(3)$ is a subgroup of $E(3)$ that indicates the symmetry group of three-dimensional Euclidean space.

Unfortunately, unlike $SO(3)$, a natural (bi-invariant) Riemannian metric can not be defined on the special Euclidean group hence its structure is discarded considering $SE(3)$ as the Cartesian product $SO(3) \times \mathbb{R}^3$.

Using this strategy, the distance between the poses $g_i = (R_i, T_i)$ and $g_j = (R_j, T_j)$ of two different cameras can be computed as

$$d_{g,R}^2(g_i, g_j) = d_R^2(R_i, R_j) + \|T_i - T_j\|^2 \tag{2.17}$$

if the Riemannian metric is considered, or

$$d_{g,F}^2(g_i, g_j) = d_F^2(R_i, R_j) + \|T_i - T_j\|^2 \tag{2.18}$$

if the Frobenius one is preferred. In both cases, the gradient on $SO(3) \times \mathbb{R}^3$ is evaluated by considering separately the rotation and translation components.

2.3 Calibration algorithms

The *calibration* is the process of estimating both the intrinsic and extrinsic parameters of a camera modeled as an ideal pinhole device. The principal elements of the pinhole model

are a plane (*image plane*) and a point (*optical center*): any point $\mathbf{Q}$ of the 3D space is projected ($\mathbf{q}$) on the image plane through the optical center, consequently passing from a three-dimensional space to a two-dimensional space, as illustrated in Appendix A. Therefore, the calibration is a necessary step in computer vision in order to extract correct information about 3D points from 2D images.

Various procedures are available in the literature, each of which has specific advantages and disadvantages. The following subsections describe two specific techniques, which can be adopted in the initialization of the algorithm implemented by Tron and Vidal that will be analyzed in the next chapters.

2.3.1 8-points algorithm

Given two cameras i and j with intersecting fields of view, one of the best known techniques to estimate the devices parameters is the 8-points algorithm. It is a method introduced by Christopher Longuet-Higgins in 1981 which exploits the correspondence between points belonging to different image planes, that are the projections of the same 3D point through the focal centers of the two cameras (*feature points*).

More specifically, let us consider a pair of cameras i and j, with relative unknown position R_{ij} and T_{ij} (rotation and translation respectively) and be $\mathbf{q_i}$ the point seen by i-th device and $\mathbf{q_j}$ the point seen by j-th device.

The 2D points $\mathbf{q_i}$ and $\mathbf{q_j}$ are related by

$$\mu_i \mathbf{q_i} = \mu_j R_{ij} \mathbf{q_j} + T_{ij}, \tag{2.19}$$

where μ_i and μ_j are the depths of the 3D point $\mathbf{Q}$ with respect to the i-th and j-th camera respectively.

Moreover, as known from computer vision, the corresponding points satisfy the epipolar constraint [3] expressed by the relation

$$\mathbf{q_i}^T T_{ij}^{\perp} R_{ij} \mathbf{q_j} = 0, \tag{2.20}$$

where $T_{ij}^{\perp}$ is an orthogonal matrix, i.e. $T_{ij}^{\perp} T_{ij} = 0$, and $T_{ij}^{\perp} R_{ij}$ is referred as *essential matrix* from which one can compute the normalized relative translation vector $t_{ij} = T_{ij}/\|T_{ij}\|$ and the relative rotation matrix R_{ij}.

More generally, the 8-points algorithm is a frequently cited strategy to compute the *fundamental matrix*, in place of the essential one, exploiting a set of eight or more point matches. The fundamental matrix contains information also on the intrinsic parameters of the camera as well as on the relative pose (see Appendix A).

The advantage of the 8-points algorithm is the simplicity of implementation [12], however it does not work properly when the feature points extracted from the images assume certain degenerate configurations, called critical surfaces [10].

[3] The *epipolar constraint* between two cameras refers to the fact that, given a point $\mathbf{q_1}$ on the first image plane, the corresponding one $\mathbf{q_2}$ on the second image plane must lie on the epipolar line. The *epipolar line* is the straight line of intersection between the image plane and the epipolar plane. The *epipolar plane* is the plane defined by the centers of the cameras and the 3D point $\mathbf{Q}$ of which $\mathbf{q_1}$ and $\mathbf{q_2}$ are the projections.

2.3.2 Bouguet's camera calibration toolbox

Another possible solution to estimate camera parameters is to exploit one of the toolbox available in Matlab. In [10] the problem is addressed employing the Bouguet's camera calibration toolbox, a specific tool designed for working with chessboard's images, widely used in computer vision.

In detail, using the functions provided by the toolbox, the computation of the cameras parameters is performed in two steps. First the *calibration matrix*, i.e. the matrix of intrinsic parameters, is computed for each device using a set of a dozen images of the same chessboard captured in different position by the same camera. Then, the extrinsic parameters of each camera are determined with respect to the chessboard reference frame.

The previously obtained results can be combined in order to recover some initial estimates of the relative rotation and translation for each pair of cameras able to communicate, as required as input in the algorithm developed by Tron and Vidal.

The advantages of this particular strategy are that it requires only a plane, the positions/orientations are not to be known, a good code is available online. On the other hand, the main disadvantage is the need of the definition of the correspondences between 3D and 2D spaces.

3

Image-based localization

In this chapter, the localization problem for a camera network is formally stated. To find a solution, two distinct approaches can be followed: a centralized or a distributed strategy. The first one involves the presence of a central unit capable of communicating with all nodes, i.e. cameras that constitute the network. In the second case, the problem is tackled by exploiting only the interactions between the agents, which are assumed to be able to exchange information between them.

Contents

3.1 Problem statement

Let us consider a camera network composed by N devices displaced in a certain limited environment. The network can be identified with an undirected graph $\mathcal{G}$ whose nodes correspond to the cameras that may differ by type. This graph has an edge from node i to node j only if the fields of view of the camera i and j are partially overlapped. Using this convention, the adjacency matrix gives information about which elements of the network share part of the scene view.

The solution of the localization problem for a camera network is to recover the absolute pose of each device, namely its rotation matrix R and translation vector T, i.e. the extrinsic parameters.

As far as translation is concerned, it is necessary to identify the three components of the vector $T \in \mathbb{R}^3$. While, in geometry various formalisms exist to express a rotation in three dimensions, as described in Appendix B. Nevertheless, whatever the formalism adopted, the parameters that describe a rotation in 3D space are always three. Therefore the problem of localization for a camera network coincides with the determination of the six parameters in total (problem with six degrees of freedom) for each device.

At this point, since absolute poses are linked to relative ones, it is possible to give a formal definition of *localized network*:

Definition 1. *A network of N agents is said to be localized (localized network) if there is a set of relative transformations $\{g_{ij} = (R_{ij}, T_{ij})\}$ between node i and node j such that, when the reference frame of the first node is fixed to $g_1 = (R_1, T_1)$, all the other absolute poses $g_i = (R_i, T_i)$, $i = 2 \dots N$ are uniquely determined.*

This definition translates in graph theory language: for any path l from node 1 to node i, there is $g_i = g_l \circ g_1$, regardless of the chosen path [1].

The set of relative transformations $\{g_{ij}\}$ that satisfy the consistency constraints given in Definition 1 can be found following one of the approaches described in the next paragraphs.

3.2 Centralized approach

Traditional large scale systems are characterized by a centralized architecture [2]. In this scenario, all nodes in the network send the information gathered in the environment to a central unit, which elaborates it in order to produce the required results.

Solving the camera localization problem via centralized approach implies that the various devices dispatch the images acquired to the central unit so that it can process them and estimate the pose of each agent, generally through the optimization of a suitable cost function.

In literature, solutions exist that are classified into linear and non-linear methods. The main advantage of the linear strategies is their computational efficiency, however these methods suffer from lack of accuracy and robustness. On the contrary, the results achieved through non-linear approaches are more accurate and robust but the computational burden is noteworthy and initial estimates are required [13].

Pursuant to the classical approach, the pose estimation issue can be formulated as a nonlinear least-squares problem whose goal is to iteratively find the pose of each device in the network such that the sum of the square distances between the solutions and the initial noisy measurements is the smallest. In other words, it is necessary to minimize a certain error metric. The issue can be solved exploiting nonlinear optimization algorithms.

The most popular techniques that are generally employed are the Gauss-Newton method and the Levenberg-Marquardt method. The former operates by iteratively linearizing the error equation around the current solution approximated by first-order Taylor series expansion (in the beginning, the current solution coincides with the noisy measurement) and then solving the linearized system for the next approximate solution. The latter can be considered as a compromise between the steepest descent [1] and the Gauss-Newton method as it behaves differently depending on the gap between the current solution and the correct one [14].

It is important to underline that, in both cases, there is no guarantee that the algorithm converges or it will converge to a correct solution, since the chosen cost function may not be convex and therefore it can present more local minima. Specifically, the Gauss-Newton method should quickly converge to the correct solution but only if the initial approximate one is close enough to it. On the contrary, when the current solution is not good (it is far from the correct one) and/or the linear system is illconditioned, the algorithm may converge slowly or even fail [14].

Finally, it is worth noting that the localization issue for a camera network is often considered within a large framework: the *bundle adjustment*, i.e. the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameters (camera pose and/or calibration) estimates [15]. Classically, bundle adjustment can also be formulated as a nonlinear least squares problem.

3.3 Distributed approach

The distributed algorithms do not require the presence of a central unit to which all the devices in the network are linked. Indeed this approach differs from the centralized one since the nodes are able to elaborate and share information with their local neighbors. This property can be exploited to estimate a certain quantity as it happens in the consensus procedures.

In particular, a generalization of the classical consensus algorithm is the essential idea of the method proposed by Tron and Vidal in [1] to estimate the absolute poses for a camera network. In the next chapter the algorithm is described in detail, while here the attention is focused on the consensus strategy.

[1] The *steepest descent method*, also known as *gradient descent method*, is a first-order optimization algorithm: to find a local minimum of a function, the procedure involves that one iteratively takes steps proportional to the opposite of the gradient (or of the approximate gradient) of the function at the current point.

3.3.1 Average consensus algorithm

Let us consider a network made up of N agents, each of which is capable of communicating only with its neighbors. A scalar state $x_i(0) \in \mathbb{R}$ is initially associated to each node $i \in \{1, 2, \ldots, N\}$ in the network.

A distributed consensus algorithm relies on an iterative protocol followed by each agent in the network to update its state until all nodes are in a consensus configuration, i.e. $x_i(T) = \alpha \; \forall i \in \{1, 2, \ldots, N\}$, where α is referred as the *collective decision* or *consensus value* of the network reached after T iterations [1].

In particular case of average consensus, the collective decision α coincides with the arithmetic mean of the initial states, that is

$$\alpha = \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i(0). \tag{3.1}$$

The protocol for iteratively reaching the average consensus value is given by

$$x_i(t+1) = x_i(t) + \varepsilon \sum_{j \in N_i} (x_j(t) - x_i(t)), \tag{3.2}$$

where $x_i(t)$ denotes the state associated to i-th node during the t-th iteration, N_i represents the set of nodes which are connected with i in the network (see Equation (2.1)) and ε is a design parameter called *step-size*.

As far as this last parameter is concerned, it can be demonstrated that if the graph associated with the network is connected and ε is less than the inverse of the maximum degree of the graph, $\Delta_{\mathcal{G}} = \max_i \deg(i)$, then the nodes converge to the mean $\bar{x}$, i.e. $\lim_{t \to \infty} x_i(t) = \bar{x}$ [16].

It can be shown that Equation (3.2) is a gradient descent step that minimizes the cost function

$$\varphi(x_1, x_2, \ldots, x_N) = \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} (x_i - x_j)^2, \tag{3.3}$$

where $i(i, j) \in \mathcal{E}$ implies that the sum is taken only on the terms $i, j \in \mathcal{V}$ (set of nodes/ agents in the network) such that $(i, j) \in \mathcal{E}$ (set of edges/ links between the nodes). Furthermore, the global minimum of the convex function (3.3) is attained when all states are in consensus configuration.

3.3.2 Riemannian consensus algorithm

The average consensus algorithm can be naturally extended to the case of Riemannian manifolds as $SO(3)$. This procedure is referred as Riemannian consensus [17].

Let $(\mathcal{M}, \langle \cdot, \cdot \rangle)$ be a Riemannian manifold with metric $\langle \cdot, \cdot \rangle$ and let $\varphi : \mathcal{M} \to \mathbb{R}$ be a smooth function defined on $\mathcal{M}$. Considering an initial point $x_0 \in \mathcal{M}$, it is possible to define a (steepest) gradient descent procedure along the geodesic, such that

$$x(t+1) = \exp_{x(t)}(-\varepsilon \mathrm{grad}_x \varphi(x(t))), \qquad x(0) = x_0. \tag{3.4}$$

Practically, at each iteration t, a new estimate $x(t+1)$ is calculated starting from the

current one, $x(t)$, and moving in the opposite direction of the gradient with a step-size ε arbitrarily chosen (small enough).

These concepts can be applied considering $\mathcal{M} = SO(3)$. In this case, the function cost (3.3) takes the form

$$\varphi(x_1, x_2, \ldots, x_N) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} d_R^2(x_i - x_j), \tag{3.5}$$

where the Riemannian distance substitutes the Euclidean one. Moreover, the consensus value is obtained updating all nodes states via (3.4).

4

Proposed algorithm

In this chapter, a solution to the localization problem is given. The algorithm proposed is derived from the work of Tron and Vidal [1]. The authors have solved the problem of finding a consistent localization for a N cameras network by defining a suitable cost function and showing how it can be minimized on the space of rigid body transformations in a distributed manner. Their strategy is based on the generalization of the existing consensus procedure starting from noisy and inconsistent measurements, and takes advantage of the intrinsic non-Euclidean structure of the space of camera poses. Exploiting the structure of the algorithm described in [1], the interest here is focused mainly on the initialization phase: new approaches are proposed with the purpose of improving the whole procedure performances.

Contents

4.1 Inputs and Outputs

The inputs required by the algorithm presented in [1] are some initial noisy measurements $\tilde{g}_{ij} = (\tilde{R}_{ij}, \tilde{t}_{ij})$ of the relative poses $g_{ij} = (R_{ij}, T_{ij})$ between each pair of cameras of the network with intersecting fields of view. The strategies introduced in Chapter 2 (calibration algorithms, Section 2.3) are two possible ways to recover some initial estimates of the relative rotation $\tilde{R}_{ij}$ and relative translation $\tilde{t}_{ij} = \tilde{T}_{ij}/\|\tilde{T}_{ij}\|$, up to a normalizing constant, between two cameras of the network with partially overlapping fields of view.

The outputs of the algorithm are, instead, the absolute poses $g_i = (R_i, T_i)$ of all the devices that form the network, referred to the same world frame system.

As a consequence, according to the definition of localized network given in Chapter 3, the goal of the algorithm proposed by Tron and Vidal is to find a set of relative transformations $\{g_{ij}\}$ that satisfy the consistency constraints cited in the Definition 1 and that, at the same time, are as close as possible to the initial estimates $\tilde{g}_{ij}$.

To tackle the problem, a natural criterion suggested by the authors of [1] is a least squares approach minimizing the sum of square distances. However, since the quantities of interest are non-Euclidean, the distance to be considered is that defined by (2.17). It is important to remark that the distance $d_{g,R}(\cdot,\cdot)$ on $SE(3)$ has been determined by decomposing the space of rigid body transformations in the Cartesian product $SO(3) \times \mathbb{R}^3$, in which $SO(3)$ is endowed with Riemannian metric. For this reason, considering the least squares criterion and being $g_{ij} = (R_{ij}, T_{ij})$ and $\tilde{g}_{ij} = (\tilde{R}_{ij}, \tilde{T}_{ij})$, a natural choice for the cost function to minimize is the following one:

$$\begin{aligned} \varphi &= \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} d^2_{g,R}(g_{ij}, \tilde{g}_{ij}) \\ &= \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} (d^2_R(R_{ij}, \tilde{R}_{ij}) + \|T_{ij} - \tilde{T}_{ij}\|^2). \end{aligned} \tag{4.1}$$

The purpose of the algorithm is to minimize φ but the consistency constraints given in the definition of localized network (Definition 1) have to be satisfied. It is worth of notice that these constraints are not distributed although they involve the entire network.

Therefore, in [1] the authors suggest to reparametrize each relative transformation g_{ij} with the absolute ones g_i and g_j: in doing so, the consistency constraints are automatically satisfied even though it requires that communication is only possible between neighboring nodes. This approach is endorsed by the equivalence expressed by the following proposition [10]:

Proposition 1. *The following are equivalent:*

1. *The network is localized.*
2. *For any cycle l, the transformation along the cycle is $g_l = (I, 0)$.*
3. *There exists a set of absolute poses $g_i = (R_i, T_i)$ such that $g_{ij} = g_i^{-1} \circ g_j$.*

The proof of the proposition is not given here but it is available in [18]. Nonetheless, as stated in [10], it is immediate to realize that, given the absolute poses g_i, one can compute

consistent relative transformations g_{ij}; conversely, if the relative transformations are given, there exists a set of g_i such that each g_{ij} factorizes as $g_i^{-1} \circ g_j$.

Adopting the proposed formulation of the problem, a new challenge emerges: the translation vectors can be estimated only within a scale factor. Thereby, it is necessary to add others unknown quantities as new variables, i.e. the scale factors λ_{ij} such that $\tilde{T}_{ij} = \lambda_{ij}\tilde{t}_{ij}$, where the vectors $\tilde{t}_{ij}$ represent the normalized translations.

As a consequence, the cost function (4.1) has to be rewritten as

$$\begin{aligned}\varphi(\{R_i\},\{T_i\},\{\lambda_{ij}\}) &= \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} d^2_{g,R}(g_i^{-1}g_j, \tilde{g}_{ij}) \\ &= \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} (d^2_R(R_i^T R_j, \tilde{R}_{ij}) + \|R_i^T(T_j - T_i) - \lambda_{ij}\tilde{t}_{ij}\|^2) \\ &= \varphi_R(\{R_i\}) + \varphi_T(\{R_i\},\{T_i\},\{\lambda_{ij}\}). \end{aligned} \tag{4.2}$$

The cost φ is the sum of two terms: φ_R involving only the rotational part R_i of the absolute pose g_i and φ_T involving all the variables.

As far as the unknown scale factor $\{\lambda_{ij}\}$ is concerned, it has to be positive and it is responsible for an undesired side-effect. In fact, if the trivial solution $T_i = T_j$, $\forall\, i,j \in \mathcal{V}$ and $\lambda_{ij} = 0$, $\forall\ (i,j) \in \mathcal{E}$ is substituted in (4.2), then the global minimum $\varphi_T = 0$ is achieved regardless of the value of the rotations. Hence, if any constraints on T_i or λ_{ij} are imposed, then the minimization of the cost function (4.2) leads to solutions that are not meaningful (e.g., all the translations collapsed to the same point) [10].

To settle both the questions, in [1] Tron and Vidal suggest to constraint the minimum scale, i.e., $\lambda_{ij} \geq 1\ \forall (i,j) \in \mathcal{E}$. The authors underline that this constraint is global because it involves all the nodes in the network; however, each node can enforce it separately, hence making it distributed.

In summary, the problem to tackle can be rewritten as the non-linear program

$$\begin{aligned} &\min_{\{R_i\},\{T_i\},\{\lambda_{ij}\}} \varphi(\{R_i\},\{T_i\},\{\lambda_{ij}\}) \\ &\text{subject to} \quad \lambda_{ij} \geq 1 \quad \forall (i,j) \in \mathcal{E} \end{aligned} \tag{4.3}$$

In order to find a distributed and optimal solution for (4.3), two simpler subproblems can be considered. First, an initial set of rotations has to be determined by optimizing φ_R only and ignoring the translational component. Next, assuming the rotations are fixed, an initial set of translations and scales has to be computed by optimizing φ_T only. The final step (that can be neglected) involves the minimization of φ over all variables.

There are many reasons to pursue a multi-step solution [1]:

- the simpler subproblems can actually correspond to real localization problems;
- by breaking the analysis into two parts, one can gain a better understanding of what the issues are in the solutions;
- minimizing the functions φ_R and φ_T is easier than minimizing φ, above all because the complete cost function has multiple local minima so it is difficult to achieve the correct one starting from a random configuration.

In the next paragraphs the iterative procedure implemented to find the solution of the problem (4.3) is described in detail: four steps are considered, i.e. initialization, estimation of rotations, estimation of translations and complete estimation/refinement. In particular, since in this work new methods to initialize the procedure are presented, the follow section is entirely dedicated to this issue.

4.2 Initialization

The first step of the procedure proposed by Tron and Vidal consists, as usual, in the initialization. In this specific case, the optimization algorithm works on absolute transformations while the initial noisy measurements are relative poses: it is necessary to determine the rotation matrix $R_i(0)$ and the translation vector $T_i(0)$ at zero-th iteration for each camera i which composes the network.

Since at the beginning the localization of the network is totally unknown, the initial set of values is arbitrary, however the basic idea is to determine it close enough to the optimal solution. In effect, the component of the cost function relative to the rotations φ_R is minimized through a Riemannian gradient descent: it is a non-linear optimization strategy that leads to one of the possible local minima of the function, which may not correspond to a correct. localization, even when the initial measures are without noise (this can be easily verified empirically as suggested in [1]).

4.2.1 Frobenius norm method

In [1] a completely distributed approach, supported by simulation results, is presented.

Regarding the initial translations, the authors observe that they can prove the existence of the solution but not its uniqueness. To solve the translational ambiguity connected with the choice of a global world frame, they suggest to impose the initial average of the translations to be zero, e.g. by simply setting $T_i(0) = 0 \; \forall i \in \mathcal{V}$.

As far as rotations are concerned, it is suggested to minimize a modified cost function φ'_R. Using the Frobenius metric on $SO(3)$ to define the rotational component of the cost function, φ_R can be replaced by

$$\varphi'_R = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} d_F^2(R_i^T R_j, \tilde{R}_{ij}) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \|R_j - R_i \tilde{R}_{ij}\|_F^2. \tag{4.4}$$

Thanks to this substitution, the calculation of the update equation for the rotations of a certain node k is simpler because the Euclidean formula can be used in the computation of the gradient:

$$\frac{\partial \varphi'_R}{\partial R_k} = \sum_{(k,i)\in\mathcal{E}} (R_k - R_i \tilde{R}_{ki}^T) + \sum_{(i,k)\in\mathcal{E}} (R_k - R_i \tilde{R}_{ik}^T). \tag{4.5}$$

In summary, in [1] the authors argue that

- it is experimentally proved that the function φ'_R is easier to minimize than φ_R;
- the results found in this way are extremely close to the ground truth;

- the function (4.4) is convex;

and, consequently, they conjecture that the function φ'_R does not have local minima but only one global minimum that is reached at the end of its minimization procedure.

However, this conjecture seems to be contradicted. Although it is not possible to prove it analytically, there are empirically demonstrated cases in which the particular symmetry in the displacement of cameras in the network and/or the communication protocol adopted and/or the distribution of errors do not allow to converge to a solution.

Therefore, in this work the attention is focused on alternative initialization approaches.

4.2.2 Spanning tree method

An initialization strategy briefly cited in [1] but widely discussed in [10] is the spanning three method. In this procedure, there are three steps to follow:

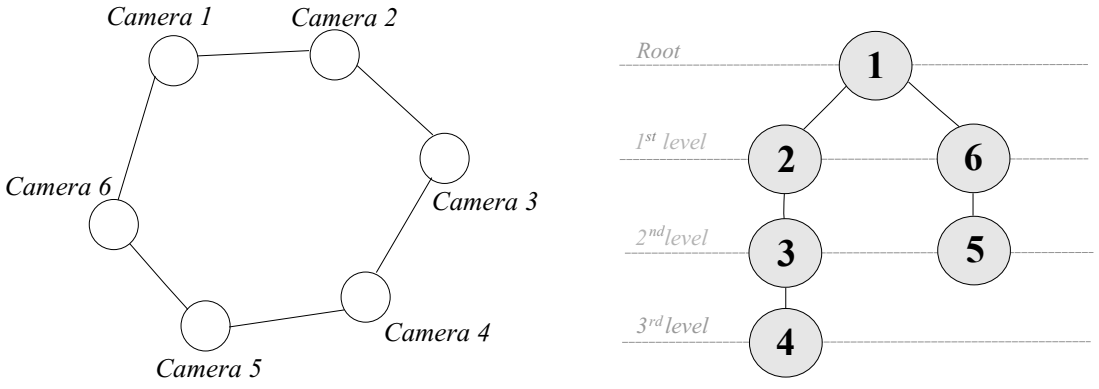
1. Choose any node as a reference (e.g. node 1) and find a spanning tree that provides the paths l_i from node 1 to any other node i in the network.

2. Set $R_1(0) = I$ and set $R_i(0) = R_j(0)\tilde{R}_{ji}$ for all $i \in \mathcal{V}$, where j is the parent node of i in the designed spanning tree.

3. Set $T_1(0) = 0$ and set $T_i(0) = R_j(0)\tilde{T}_{ji} + T_j(0)$ for all $i \in \mathcal{V}$, where j is the parent node of i in the designed spanning tree.

As underlined in [10], it is important to observe that, intuitively, the relative transformation errors are composed and the imprecision increases by growing the distance of the i-th node from the reference node (root); thus it is advisable to create a spanning tree as balanced as possible.

To better understand the method proposed, an example is given. Let us consider a network made up of six cameras, represented by the graph reported in Figure 4.1 (a). It is a particular topological case since each device can communicate only with its neighbors: the graph associated to the network is circulant [1].



(a) Graph based network (b) Spanning tree

Figure 4.1.: Example of single spanning tree initialization method

[1] A *circulant graph* is an undirected graph that has a cyclic group of symmetries, i.e the i-th vertex is adjacent to the $(i + j)$-th and $(i - j)$-th vertices for each j in a list l.

The Figure 4.1 (b) shows a possible construction of the spanning tree for the graph analyzed: it has the node 1 as root (so this node is chosen as reference) and two branches constituted by three and two nodes respectively. Evaluating this configuration, the initial values for rotations and translations are:

- $R_1(0) = I$ and $T_1(0) = 0$;
- $R_2(0) = R_1(0)\tilde{R}_{12}$ and $T_2(0) = R_1(0)\tilde{T}_{12} + T_1(0)$;
- $R_3(0) = R_2(0)\tilde{R}_{23}$ and $T_3(0) = R_2(0)\tilde{T}_{23} + T_2(0)$;
- $R_4(0) = R_3(0)\tilde{R}_{34}$ and $T_4(0) = R_3(0)\tilde{T}_{34} + T_3(0)$;
- $R_5(0) = R_6(0)\tilde{R}_{65}$ and $T_5(0) = R_6(0)\tilde{T}_{65} + T_6(0)$;
- $R_6(0) = R_1(0)\tilde{R}_{16}$ and $T_6(0) = R_1(0)\tilde{T}_{16} + T_1(0)$.

The initialization method based on the construction of a spanning tree has the advantage of being robust and fast. Nevertheless it has also some disadvantages.

First of all, it could appear partially distributed; in fact the reference node must be chosen manually and this operation can fail. Secondly, the initial set of translations is computed without considering the scale factors λ_{ij}. However these variables should be set in a reasonable manner and on the basis of the information a priori known about the environment in which the camera network is located. For example, in a perimeter surveillance framework, it is reasonable to assume $\lambda_{ij} = 1\ \forall (i,j) \in \mathcal{E}$ given that the cameras are probably almost equidistant, in this way it is possible to obtain the whole estimation except only for a scale factor λ [10].

4.2.3 Multi spanning trees method

An alternative solution for the initialization step is to use a strategy based on the construction of multiple spanning trees.

This approach relies on two important assumptions. The main one concerns the noisy measurements of the relative poses of the cameras in the network. Indeed, in general $\tilde{g}_{ij}$ is different from $\tilde{g}_{ji}^{-1}$ for all $(i,j) \in \mathcal{E}$. The second one is the need of knowing the absolute pose of one device (e.g. camera 1) in the network in order to set the reference frame.

Under these hypothesis, the first step of the procedure is the determination of all the possible different paths that connect the reference node to any other node in the graph associated with the network.

Analyzing a certain path P_k that links reference node 1 to the generic one i, it can be shown that it is possible to identify 2^{n_k} initial absolute poses for the camera i, where n_k is the number of edges that separate 1 from i in the chosen path. In fact, for each edge of the path, the pose of the previous node, p, can be exploited to compute the pose of the next one, s, in two ways: either using g_{ps} or using g_{sp}^{-1}. As consequence, applying the same reasoning for all the m paths from node 1 to node i, it is feasible to individuate $\sum_{k=1}^{m} 2^{n_k}$ initial absolute poses for the camera i.

To clarify the various steps of the procedure described, it is useful to give an example.

Let us consider the network of four cameras, whose associated graph is displayed in Figure 4.2. Also in this case, a circulant graph is evaluated.

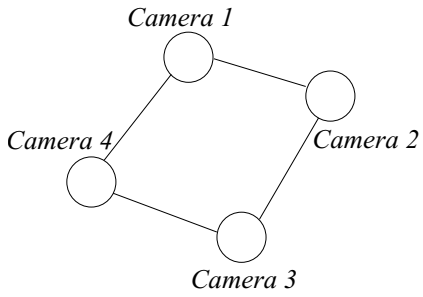
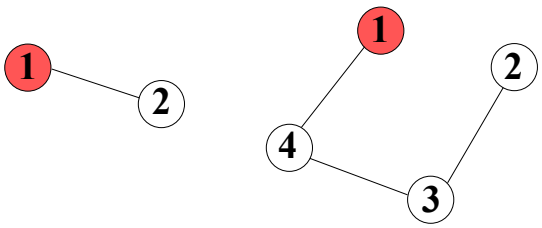


Figure 4.2.: Graph based network

All possible paths that link node 1 to any other node of the network are reported in Figures 4.3 (a), (b) and (c). Observing them, there are

- $2^1 + 2^3 = 10$ initial poses for the camera 2,
- $2^2 + 2^2 = 8$ initial poses for the camera 3,
- $2^3 + 2^1 = 10$ initial poses for the camera 4.



(a) Paths from 1 to 2

(b) Paths from 1 to 3

(c) Paths from 1 to 4

Figure 4.3.: Paths from node 1 to any other node of the network

The last step of the initialization procedure is to compute the average of the initial poses so obtained.

The multi spanning trees method has the great advantage of consistently reducing the uncertainty of the initial poses with respect to the strategy based on the construction of a single spanning tree. However, the scale factor issue is not solved and once again one of the cameras of the network, whose pose has to be known, maintains a privileged role being chosen as the reference node at the beginning of the procedure. Moreover, the multi spanning trees method is not fast as that illustrated in 4.2.2.

The slowness of the procedure is essentially caused by the fact that the determination of multiple spanning trees requires a certain computational burden that increases proportionally to the number of possible configurations, which is in turn proportional to the number of devices in the network. Furthermore, the execution speed is penalized by the fact that the average of the rotations is not arithmetically calculated as for the translations. Since the rotational matrices belong to the group $SO(3)$, the result of the average operation is constrained to be again an orthogonal matrix.

To solve the aforementioned issue, it is advisable to reason as in [19]. In this article, the search of the mean of a certain number of rotational matrices is tackled as a minimization problem: given N rotations R_i, it is necessary to find the rotation R in correspondence to which a specific cost function takes the minimum value. It is reasonable to choose as the function to minimize as the sum of the squares of the distances between the different values R_i and R, i.e. to operate once again according to the least squares criterion.

Exploiting the geodesic metric (2.11), the cost function can be expressed by

$$C(R) = \sum_{i=1}^{N} d_R^2(R, R_i). \tag{4.6}$$

The minimization of (4.6) is known in literature as the *Karcher mean of the rotations*. In [20] a convergent algorithm to find this mean is presented: the main idea is computing the average in the tangent space and then projecting back onto the manifold $SO(3)$ via the exponential map.

Algorithm 1 Computing the rotation mean

1: Set $R := R_1$. Choose a tolerance $\epsilon > 0$.
2: **loop**
3: Compute $\mathbf{r} = \frac{1}{N} \sum_{i=1}^{N} \log(R^T R_i)$
4: **if** $\|\mathbf{r}\| < \epsilon$ **then**
5: **return** R
6: **end if**
7: Update $R = \exp(\mathbf{r})$
8: **end loop**

Implementing the Algorithm 1 for each device i of the network, an initial set of rotations $\{R_i(0)\}$ can be obtained. On the other hand, as far as translations are concerned, the simple arithmetical mean has to be computed for every camera in order to draw the initial values $T_i(0)$, $\forall i \in \mathcal{V}$.

4.2.4 Multi spanning trees method with virtual camera

To overcome the problems related to the choice of the reference node, it is useful to proceed by opting for a multi spanning trees method that differs from the strategy described in the previous paragraph as it includes an additional artificial node in the network, referred as *virtual camera*.

The peculiarity of this node is that it is able to communicate with all other devices in the network. In other words the knowledge the relative poses between the virtual

camera and all the other ones present in the network is available. In graph theory, this assumption translates into the inclusion of a node having an edge that links it to every other node in the graph, i.e. the virtual camera is represented by a vertex adjacent to every other vertex in the graph. This request is key in the procedure considered here since it is based on the construction of different spanning trees, all with the same root, namely the virtual camera.

In practice, the decision of maintaining the same root results in fixing a certain reference system against which the poses are absolute certain, which is why the position of the artificial node is not entirely irrelevant because it determines the location of the observer in the scene.

An example is useful to clarify this. Let us reconsider the graph in Figure 4.1 (a): the first thing to do is to insert the artificial node in the network, as shown in Figure 4.4.

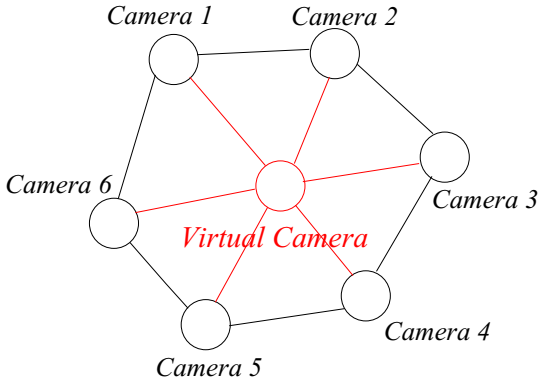


Figure 4.4.: Graph based network with artificial node

Then, it is necessary to build various spanning trees: it is important setting always the virtual camera as the root. Figure 4.5 shows a few possible solutions. It is worth noticing that the considered trees are perfectly balanced: this choice is motivated by the previous observation about the propagation of the errors.

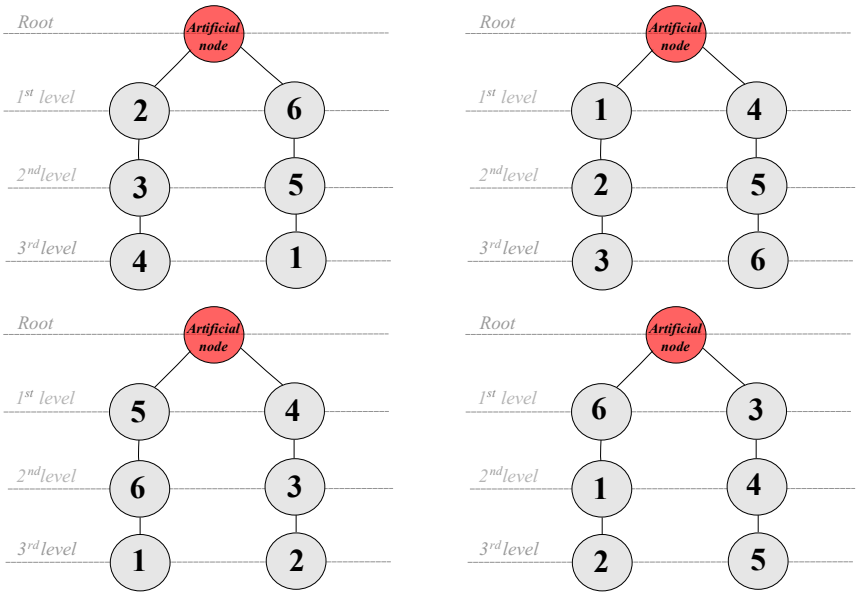


Figure 4.5.: Spanning trees

The initialization is concluded by computing the average of the values of the rotations and translations obtained by building the various spanning trees. To calculate the means, it is necessary to proceed as illustrated in Section 4.2.3.

The multi spanning trees method which involves the introduction of a virtual camera is not much faster than the strategy based on the construction of multiple spanning trees without the addition of the artificial node. The motivation is the fact that the greater computational load is represented by the the construction of the trees and by the calculation of the averages. Nevertheless, the method presented in this paragraph has the advantage of being fully distributed: no camera is chosen as the reference, as the point of view of the observer is a priori imposed.

4.3 Steps

The main steps of the algorithm proposed by Tron and Vidal are the separate estimations of the absolute rotation and translation for each device in the network. The procedure also includes a final step in which the cost function φ is optimized with respect to all the variables of the localization problem.

The algorithm thus consists in three distinct cycles *for*, whose number of iterations may be different. In particular, the last step is generally performed a smaller number of times, if not completely ignored since it often leads to worsening the estimates obtained previously.

4.3.1 Estimation of rotations

The first subproblem to be tackled is the estimation of the rotational part R_i of the absolute pose g_i. In [1] the authors propose a distributed and iterative procedure that uses the framework of consensus algorithm as the key idea. The principal innovation is the need to replace the Euclidean gradient descent with the Riemannian gradient descent, relative to the space of rotations.

The proposed algorithm proceeds iteratively in two steps until convergence.

Given the functional

$$\varphi_R(\{R_i\}) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} d_R^2(R_i^T R_j, \tilde{R}_{ij}), \tag{4.7}$$

1. Each node k of the network computes the Riemannian gradient of φ_R with respect to its rotation R_k. This operation can be carried out using (2.16) as

$$\text{grad}_{R_k}\varphi_R = -R_k \sum_{(k,i)\in\mathcal{E}} [\log(R_k^T R_i \tilde{R}_{ki}^T) + \log(R_k^T R_i \tilde{R}_{ik}^T)]. \tag{4.8}$$

   In [1] the authors emphasize the fact that the computation of $\text{grad}_{R_k}\varphi_R$ involves a sum over nodes i, which are neighbors of node k only. Therefore a distributed approach can be enacted. Moreover, it is worth of notice that the result of (4.8) is a vector that belongs to the tangent space of R_k.

2. Each node k of the network needs to update its rotation moving along the geodesic defined by (4.8). Specifically, if $R_k(t)$ denotes the estimate of R_k at the t-th iteration, then $R_k(t+1)$ is determined by moving along the geodesic in the direction $-\text{grad}_{R_k(t)}$ with a properly chosen step-size ε.

   The update equation can be written as

$$R_k(t+1) = \exp_{R_k(t)}(-\varepsilon \, \text{grad}_{R_k(t)} \varphi_R). \tag{4.9}$$

The number of iterations to execute must be fixed a priori. It needs to be not too high to avoid slowing down the algorithm but it also needs to be not too low otherwise the final estimates are not enough precise (the cost function does not reach a value enough close to the minimum).

4.3.2 Estimation of translations

The second subproblem to solve is the estimation of the translational part T_i of the absolute pose g_i. The issue can be tackled by using an approach similar to that presented for the rotations, i.e. minimizing the cost function φ_T using projected gradient descent.

The component φ_T of the original cost function involves all the variables of the localization problem:

$$\varphi_T(\{R_i\}, \{T_i\}, \{\lambda_{ij}\}) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \|R_i^T(T_j - T_i) - \lambda_{ij}\tilde{t}_{ij}\|^2. \tag{4.10}$$

For this reason, the rotations $\{R_i\}$ are assumed fixed in order to reduce the complexity of the problem. The still unknown variables are $\{T_i\}$ and $\{\lambda_{ij}\}$.

First, the gradient of φ_T is computed with respect to T_k for each node k of the network as

$$\frac{\partial \varphi_T}{\partial T_k} = \sum_{(k,i)\in\mathcal{E}} [2(T_k - T_i) + \lambda_{ki} R_k \tilde{t}_{ki} - \lambda_{ik} R_i \tilde{t}_{ik}]. \tag{4.11}$$

Then, similarly, it is determined with respect to λ_{lk} for each edge (l,k) of the network as

$$\frac{\partial \varphi_T}{\partial \lambda_{lk}} = \lambda_{lk} - (T_l - T_k)^T R_k \tilde{t}_{kl}. \tag{4.12}$$

Finally, the descent gradient is exploited in the calculation of the update equations of both translation and scale factors. Concerning the latter ones, it is important to pay attention to the constraint imposed on the scale factors: $\lambda_{ij} \geq 1 \; \forall (i,j) \in \mathcal{E}$.

As a consequence, the updating equations are

$$T_k(t+1) = T_k(t) - \varepsilon \frac{\partial \varphi_T}{\partial T_k(t)}, \tag{4.13}$$

$$\lambda_{lk}(t+1) = \max\left\{1, \lambda_{lk}(t) - \varepsilon \frac{\partial \varphi_T}{\partial \lambda_{lk}(t)}\right\}. \tag{4.14}$$

The parameter ε indicates the step-size that can be chosen in an appropriate manner to guarantee the convergence of the procedure to the global minimum. Analogously, it

is necessary to set the number of the iterations following the same reasoning carried out for the rotation estimates.

4.3.3 Complete estimation

The final step of the algorithm proposed by Tron and Vidal provides for the minimization of the original cost function φ.

The procedure to follow is a combination of the methods illustrated in the previous paragraphs. In fact, it is necessary to:

1. Find a consistent set of rotations by minimizing φ_R.
2. Find a consistent set of translations and scale factors though the optimization of φ_T, while the rotations are kept fixed.
3. Minimize φ in order to refine all the previous estimates.

It is fundamental to observe that if the function to minimize is φ then the gradient with respect to R_k is different from the one in (4.8). In fact, it contains an additional term coming from φ_T, i.e.

$$\begin{aligned}\operatorname{grad}_{R_k}\varphi = \operatorname{grad}_{R_k}\varphi_R + \\ + \sum_{(k,i)\in\mathcal{E}} \lambda_{ki}[(T_i - T_k)\tilde{t}_{ki}^T - R_k\tilde{t}_{ki}(T_i - T_k)^T R_k].\end{aligned} \tag{4.15}$$

The updates then follow the gradient as before.

To summarize, the Figure 4.6 shows the entire structure of the proposed algorithm remarking the inputs and the outputs of the various optimization steps.

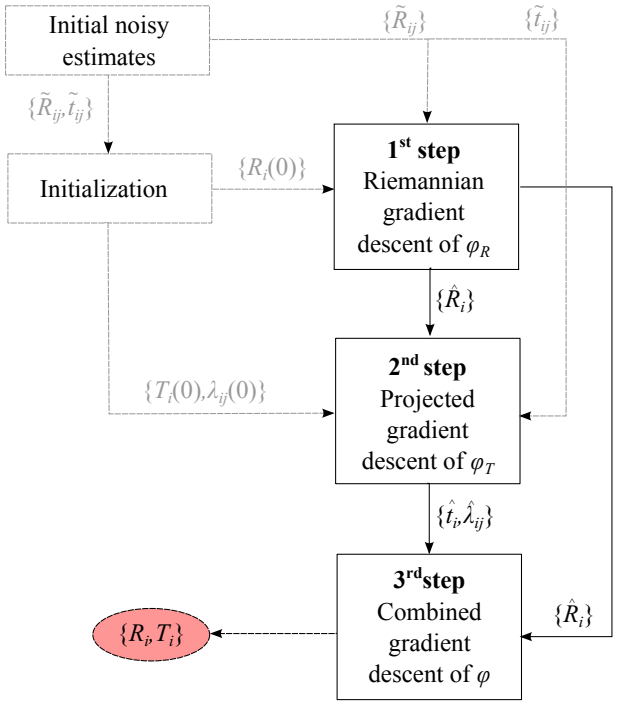


Figure 4.6.: Schematic representation of algorithm proposed

5

Frequency domain technique

This chapter is devoted to the presentation of an original approach for reconstructing the relative poses in a camera network. This technique operates in frequency domain exploiting the Fourier transform properties [21]. The algorithm has been proposed by Cortelazzo et al. in 2002 [11]: the camera system localization problem is tackled by decoupling the estimate of the rotation parameters from the estimate of the translation ones. This operation is made possible by exploiting the frequency domain, as demonstrated by the first part of the chapter in which the main properties of the Fourier transform are introduced. In the second part, the solution to the localization problem is minutely described.

The decision to dedicate an entire chapter to this method is justified by the fact that the initial idea of this thesis work was to merge the algorithm proposed by Cortelazzo et al. with the one conceived by Tron and Vidal. More specifically, the strategy operating in frequency provides as output a set of relative poses. These can be used as inputs in the iterative procedure described in Chapter 4 instead of results arising from the calibration algorithms. The belief is that the algorithm in [11] produces more reliable estimates improving also the accuracy of the Tron-Vidal method and reducing the number of iterations to make the execution more faster. Truthfully, many problems have appeared concerning the code in frequency domain, therefore it has not be possible to validate the theoretical results through the simulations on real or synthetic camera network. For this reason this combined solution is at the forefront of future work.

Contents

5.1 Fourier transform

A continuous time signal $x(t),\ t \in \mathbb{R}$ can be identified through its Fourier transform (FT), which is itself a complex function of a real variable, defined as

$$\mathcal{X} : \mathbb{R} \mapsto \mathbb{C}, \quad \mathcal{X}(k) = \mathcal{F}[x(t)] = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi kt} dt, \tag{5.1}$$

where the variable denoted by k has the meaning of frequency, the inverse of time.

The Fourier transformation can be inverted allowing to unambiguously determine the original signal x from its transform $\mathcal{X}$ as they are tied by the following relation (*inverse Fourier transform equation*)

$$x(t) = \int_{-\infty}^{+\infty} \mathcal{X}(k) e^{j2\pi kt} dk. \tag{5.2}$$

The knowledge of the Fourier transformation (5.1) of a signal is equivalent to the knowledge of the signal itself so it can be seen as an alternative manner to represent the signal preserving its characteristics [22].

The Fourier transform has many properties that provide with a significant amount of insight into the transform and into the relationship between the time-domain and the frequency-domain description of the signal [23].

The following list itemizes only the principal ones, useful to better understand the algorithm presented in the next section. In particular, let us denote the signals as $x(t)$ and $y(t)$, while their FT are referred as $\mathcal{X}(k)$ and $\mathcal{Y}(k)$ respectively.

- (*Linearity*) Given a signal $ax(t) + by(t)$ with $a, b \in \mathbb{R}$, its Fourier transform is expressed as

  $$a\mathcal{X}(k) + b\mathcal{Y}(k), \tag{5.3}$$

  as it can be derived by applying the definition (5.1). The linearity property is easily extended to a linear combination of an arbitrary number of signals [23].

- (*Time Shifting*) If the signal $x(t)$ is time shifted by a quantity $t_0 \in \mathbb{R}$, i.e. $x(t - t_0)$, then its FT becomes

  $$\mathcal{X}(k) e^{-j2\pi kt_0}. \tag{5.4}$$

  This fact can be proved replacing t by $t - t_0$ in the equation (5.1) and considering the properties of the integral calculus. One consequence of the time-shift property is that the magnitude of the Fourier transform for a time-shifted signal is not altered [23].

- (*Scaling*) Let us consider the signal $x(at)$, where a is a not null real constant. Its Fourier transform is

  $$\frac{1}{|a|} \mathcal{X}\left(\frac{k}{a}\right). \tag{5.5}$$

  This property follows directly from the definition (5.1).

- (*Convolution*) The convolution operation between two signals is expressed as $(x * y)(t) = \int_{-\infty}^{+\infty} x(\tau) y(t - \tau) d\tau = \int_{-\infty}^{+\infty} x(t - \tau) y(\tau) d\tau$. Passing to frequency domain,

it results

$$\mathcal{X}(k)\mathcal{Y}(k), \tag{5.6}$$

i.e. the Fourier transform maps the convolution of two signals into the product of their FTs.

- (*Product*) Analyzing the previous property, it is easy to verify that the Fourier transform of the product of two signal, $x(t)y(t)$, is the convolution result

$$(\mathcal{X} * \mathcal{Y})(k). \tag{5.7}$$

Because of duality between the time and frequency domain, the convolution in time domain corresponds to a multiplication in frequency domain and viceversa.

The Fourier transform is a very powerful tool to analyze and process images, converting the data from the spatial to the frequency domain.

In general, the mathematical representation of an image is a function of two spatial variables, i.e. $f(x, y)$. Consequently its Fourier transform is not unidimensional but the frequency variable k is replaced by the frequency vector $\mathbf{k}$.

More specifically, the value of the function f at a given point is the image intensity at that point (spatial domain), whereas its transform may be considered a representation of the signal associated with the rate of change of luminance in the image itself (frequency domain). As the value of the spatial frequency increases, the Fourier transform denotes the level of visual details contained in the image. As an example, the transform of a blurred image is mainly localized at low frequencies, while the transform of an image with minute details presents significant values at high frequencies [24].

In summary, it is important to emphasize that the Fourier transform of an image contains the same information of the image itself: the two domains differ only in the way in which information is represented.

5.2 Localization problem

The properties of the Fourier transform listed in previous paragraph are the key elements of the algorithm proposed by Cortelazzo et al. [11] in order to solve the problem of reconstruction of the relative poses for a camera network. The authors adopt a frequency approach that allows decoupling the estimate of the rotation parameters from the estimate of the translation parameters. In [11] the efficiency of the algorithm is clearly confirmed by the results of extensive testing.

The procedure operating in frequency domain is based on three steps. The first two steps are dedicated to the estimation of the rotation parameters exploiting a convenient representation and the projection of the magnitude of the Fourier transform. The last step is necessary for the recovery of the translational part of the pose that occurs using a standard phase correlation technique after the compensation for rotation in one of two views.

As it was underlined in the introduction, the initial idea of this work was to take advantages of the results reached by this original method using them as inputs for the

Tron-Vidal algorithm with the purpose of achieving a more accurate estimate of the camera absolute poses. The Cortelazzo procedure, in fact, offers many advantages.

First of all, it is a featureless algorithm which exploits the information about the geometric regularity captured by Fourier transform without the need of knowing the correspondence between two views. Then, it allows to split a 6-parameters problem into two separate 3-parameters problems and to converge to a global solution. Finally, it can operate automatically requiring the common region between adjacent views as inputs: this information can be obtained simply following a preassigned taking procedure.

5.2.1 Problem statement

Let $s_1(\mathbf{x})$ and $s_2(\mathbf{x})$, $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T \in \mathbb{R}^3$, be the two maximally overlapping portions of range data sets relative to the free-form surface of a 3D rigid object. The two surfaces, supposed to be known a priori from the adopted 3D scanning procedure, are related through a rigid rototranslational motion as

$$s_2(\mathbf{x}) = s_1(R^{-1}\mathbf{x} - T), \tag{5.8}$$

i.e. $s_1(\mathbf{x})$ is rotated according to the matrix $R \in SO(3)$ and then the result of this operation is translated by the vector $T \in \mathbb{R}^3$ obtaining $s_2(\mathbf{x})$.

In order to avoid the mathematical difficulties about the Fourier domain, the two common regions $s_1(\mathbf{x})$ and $s_2(\mathbf{x})$ are convolved with a suitable Gaussian kernel obtaining the 3D companion solids $l_1(\mathbf{x})$ and $l_2(\mathbf{x})$ that ideally maintain the same spatial information as the corresponding 3D surfaces. For example, defining the kernel $h(\mathbf{x}) = e^{-\sigma\|\mathbf{x}\|^2}$, $\sigma \in \mathbb{R}$, a small solid wighted region is created around surface data so that the weight decreases proportionally to the distance from the surface itself.

Passing to the frequency domain, the Fourier transformation of the relationship between $l_1(\mathbf{x})$ and $l_2(\mathbf{x}) = l_1(R^{-1}\mathbf{x} - T)$ yields

$$\mathcal{L}_2(\mathbf{k}) = \mathcal{L}_1(R^{-1}\mathbf{k})e^{-\jmath 2\pi \mathbf{k}^T RT} \tag{5.9}$$

$$\text{where} \quad |\mathcal{L}_2(\mathbf{k})| = |\mathcal{L}_1(R^{-1}\mathbf{k})|. \tag{5.10}$$

It worth of notice that the estimation of the rotational part can be decoupled from the estimation of the translation part because it conditions only the magnitude of the Fourier transformation according to the shift property. In this way , it is possible to estimate R and then T, simplifying the original problem involving six parameters into two 3-parameters problems.

5.2.2 Estimation of rotation matrix

The first two steps of the procedure suggested in [11] are employed to estimate to rotation parameters. The formalism adopted to express $R \in SO(3)$ is the *equivalent axis representation*: the 3D rotation is described as a single rotation of amplitude ψ around a single axis having the direction individuated by the unit vector $\omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T \in \mathbb{R}^3$, $\|\omega\| = 1$. More details about this representation are available in Appendix B, however

it imposes $R = e^{\Omega\psi},\ \Omega = \Omega(\omega)$.

The first step of algorithm in [11] coincides with the estimation of the rotation axis exploiting the relationship between the Fourier transform magnitudes. In fact, it is easy to prove that the direction given by the vector ω belongs to the locus $\Delta(\mathbf{k}) = 0$, where

$$\Delta(\mathbf{k}) = \left| \frac{|\mathcal{L}_1(\mathbf{k})|}{\mathcal{L}_1(\mathbf{0})} - \frac{|\mathcal{L}_2(\mathbf{k})|}{\mathcal{L}_2(\mathbf{0})} \right| = \left| \frac{|\mathcal{L}_1(\mathbf{k})|}{\mathcal{L}_1(\mathbf{0})} - \frac{|\mathcal{L}_1(R^{-1}\mathbf{k})|}{\mathcal{L}_1(\mathbf{0})} \right|. \tag{5.11}$$

As consequence, to find the rotation axis it is necessary to

1. express the difference function $\Delta(k_x, k_y, k_z)$ in spherical coordinate system as $\Delta_s(k_\rho, k_\varphi, k_\theta)$;

2. determine the radial projection of $\Delta_s(k_\rho, k_\varphi, k_\theta)$, i.e. identifying the function $\mathcal{P}(k_\varphi, k_\theta) = \int_0^\infty \Delta_s(k_\rho, k_\varphi, k_\theta) dk_\rho$;

3. minimize the quantity $\mathcal{P}(k_\varphi, k_\theta)$ (e.g. through a simulated annealing method) in order to obtain the angular coordinates estimate $(\hat{\varphi}, \hat{\theta})$;

4. compute the estimate of the vector ω as $\hat{\omega} = \begin{bmatrix} \cos\hat{\varphi}\sin\hat{\theta} & \sin\hat{\varphi}\sin\hat{\theta} & \cos\hat{\theta} \end{bmatrix}^T$.

It is worth noticing that the use of the radial projection simplifies the search in $\mathbb{R}^3$ for a line of the locus $\Delta(\mathbf{k}) = 0$ into the minimization over $\mathbb{R}^2$ of a surface [11].

The second step of the procedure is the estimation of the rotation angle.

To start, it is convenient to define a new coordinate system (u, v, w) with an axis in direction of $\hat{\omega}$, i.e.

$$\mathbf{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = C^T \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix}, \quad \text{where} \quad C = \begin{bmatrix} -\sin\hat{\varphi} & -\cos\hat{\theta}\cos\hat{\varphi} & \sin\hat{\theta}\cos\varphi \\ \cos\hat{\varphi} & -\cos\hat{\theta}\sin\hat{\varphi} & \sin\hat{\theta}\sin\hat{\varphi} \\ 0 & \sin\hat{\theta} & \cos\hat{\theta} \end{bmatrix}. \tag{5.12}$$

Using the notation $|\tilde{\mathcal{L}}_n(\mathbf{u})| = |\mathcal{L}_n(C\mathbf{u})|,\ n = 1, 2$, the relationship (5.10) about the Fourier transform magnitudes can be rewritten as

$$|\tilde{\mathcal{L}}_2(\mathbf{u})| = |\tilde{\mathcal{L}}_1(C^{-1}R^{-1}C\mathbf{u})| = |\tilde{\mathcal{L}}_1(R_w(\psi)\mathbf{u})|, \tag{5.13}$$

or more explicitly, as

$$\left| \tilde{\mathcal{L}}_2 \left(\begin{bmatrix} u \\ v \\ w \end{bmatrix} \right) \right| = \left| \tilde{\mathcal{L}}_1 \left(\begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \right) \right| = \left| \tilde{\mathcal{L}}_1 \left(\begin{bmatrix} \mathbf{r}(\psi) \\ 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \right) \right|. \tag{5.14}$$

It is important to observe that the matrix $R_w(\psi)$ belongs to the group $SO(3)$, as it is a rotation matrix. Analogously, the submatrix $\mathbf{r}(\psi) \in SO(2)$.

To determine the rotation angle ψ, it possible to proceed similarly to what previously done, projecting the Fourier transform magnitudes along the w-axis

$$p_n(u, v) = \int_{-\infty}^{+\infty} |\tilde{\mathcal{L}}_2(u, v, w)| dw, \quad n = 1, 2. \tag{5.15}$$

This operation allows to reduce the problem to a planar rotation estimation. Furthermore, the projection can be used again to cut down the dimensionality of the issue. In fact, defining the polar reference system (r, ν) related to the system (u, v), it is possible to express the quantity $p_n(u, v)$ as $\tilde{p}_n(r, \nu) = p_n(r\cos\nu, r\sin\nu)$, $n = 1, 2$ and then to compute the 1D radial projections

$$f_n(\nu) = \int_0^\infty \tilde{p}_n(r, \nu)dr, \quad n = 1, 2 \tag{5.16}$$

$$\text{such that} \quad f_2(\nu) = f_1(\nu - \psi). \tag{5.17}$$

The original 2D problem of estimating the rotation angle ψ is turned into the problem of estimating a 1D translation shift (of value ψ) and this issue can be solved by a 1D phase correlation technique [11]. This method is based on the Fourier transform, therefore, because of its Hermitian symmetry, the estimates $\hat{\psi}$ and $\hat{\psi} + \pi$ are both valid solutions.

The value $\hat{\psi}$ is computed analyzing the phase correlation function $q(\nu)$. This last one is derived considering the relationship (5.17) in the frequency domain, $F_2(k_\nu) = F_1(k_\nu)e^{-j2\pi k_\nu \psi}$, and then computing the inverse Fourier transform of the normalized product, i.e.

$$\begin{aligned} q(\nu) &= \mathcal{F}^{-1}\left[\frac{F_1^*(k_\nu)F_2(k_\nu)}{|F_1(k_\nu)F_2(k_\nu)|}\right] \\ &= \mathcal{F}^{-1}[e^{-j2\pi k_\nu \psi}] \\ &= \delta(\nu - \psi). \end{aligned} \tag{5.18}$$

The ambiguity on the correct rotation matrix is removed in the last step of Cortelazzo algorithm.

5.2.3 Estimation of translation vector

Before proceeding with the estimate of the translation vector, it is necessary to understand which of the possible candidates $\hat{R}_1 = e^{\hat{\Omega}\hat{\psi}}$ and $\hat{R}_2 = e^{\hat{\Omega}(\hat{\psi}+\pi)}$ for the rotation estimate is the correct one.

Let us first assume that the estimate of rotation axis and angle are not biased by any errors ($\hat{\Omega} = \Omega$ and $\hat{\psi} = \psi$) so that $\hat{R}_1 = R$ and consider the two new signals

$$d_1(\mathbf{x}) = l_2(\hat{R}_1\mathbf{x}) = l_1(R^{-1}R\mathbf{x} - T) = l_1(\mathbf{x} - T), \tag{5.19}$$

$$d_2(\mathbf{x}) = l_2(\hat{R}_2\mathbf{x}) = l_1(R^{-1}\hat{R}_2\mathbf{x} - T) \tag{5.20}$$

$$= l_1(e^{-\Omega\psi}e^{\Omega(\psi+\pi)}\mathbf{x} - T) = l_1(e^{\Omega\pi}\mathbf{x} - T). \tag{5.21}$$

It can be observed that the first signal coincides with $l_1(\mathbf{x})$ translated by T, while the second one is a version of $l_1(\mathbf{x})$ modified through the translation T but also through a reflection, i.e. a rotation of π.

The authors of [11] suggest to use a phase correlation type algorithm for estimating the shift T as well as for solving the disambiguation problem.

Similarly to the second step, it is necessary to define the normalized ratios

$$\Phi_1(\mathbf{k}) = \frac{\mathcal{L}_1^*(\mathbf{k})D_1(\mathbf{k})}{|\mathcal{L}_1(\mathbf{k})D_1(\mathbf{k})|} = e^{-j2\pi\mathbf{k}^T T} \qquad \text{and} \qquad \Phi_2(\mathbf{k}) = \frac{\mathcal{L}_1^*(\mathbf{k})D_2(\mathbf{k})}{|\mathcal{L}_1(\mathbf{k})D_2(\mathbf{k})|} \tag{5.22}$$

where $D_n(\mathbf{k}) = \mathcal{F}[d_n(\mathbf{x})],\ n = 1, 2$, and then to evaluate their inverse Fourier transforms.

It is important to note that

$$\phi_1(\mathbf{x}) = \mathcal{F}^{-1}[\Phi_1(\mathbf{k})] = \delta(\mathbf{x} - T), \tag{5.23}$$

while $\phi_2(\mathbf{x}) = \mathcal{F}^{-1}[\Phi_2(\mathbf{k})]$ is not an impulsive function. As a consequence, the location of the peak of $\phi_1(\mathbf{x})$ suggests the estimate of the translation vector T, moreover the comparison between $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ clarifies which is the correct rotation matrix.

In summary, it is necessary to underline that this last step is based on the hypothesis of absence of noise in order to simplify the computation. In real situations the estimation of rotation parameters are inevitably affected by uncertainty, furthermore if the errors are not too large then the discrimination criterion is still valid. In any way, it is possible to conclude that the most critical part of the algorithm is the estimate of the rotation matrix and the error on R is proportional to the error on translation vector subsequently estimated [11].

6

Validation

This chapter is devoted to the validation of the algorithm previously proposed. First of all, some analytic results are derived concerning the optimization of the part of the cost function related to rotations. It is important to put in evidence that the reasoning is conducted in a 2D scenario because it is not possible to find an analytic formulation of the problem in the three-dimensional case. Considering a determined case study, the results analytically obtained are compared with the solution reached through the implementation of the algorithm. This approach allows to gain better insights in the iterative procedure and to asses its validity and correctness in a simplified scenario. In the second part of the chapter, the performances of the whole algorithm described in Chapter 4 are tested simulating different setups in Matlab environment. Particular attention is devoted to the analysis of the differences that arise from the use of various initialization strategies. Finally, in the last part of the chapter a real setup is evaluated.

Contents

6.1 Analytic results

The analysis of the algorithm proposed in Chapter 4 is particularly interesting when the degree of freedom of the localization problem is reduced to 1, i.e. a 2D scenario is considered. In this case, the optimization of the rotational part of the cost function has a nice analytic formulation. It is also worth noticing that in the two-dimensional case the function φ_R converges to the same solution obtained with a centralized approach; meanwhile, when dealing with 2 or 3 degrees of freedom in the space of rotations, the optimization functional presents multiple local minima: for these reasons, only in 2D space the minimum can be found analytically.

Using the notations introduced in Chapter 2 about the Riemannian metric, the function φ_R for a N cameras network can be rewritten in a more explicit form:

$$\begin{aligned}\varphi_R &= \frac{1}{2}\sum_{i=1}^{N}\sum_{(i,j)\in\mathcal{E}} d_R^2(R_i^T R_j, \tilde{R}_{ij}) \\ &= -\frac{1}{4}\sum_{i=1}^{N}\sum_{(i,j)\in\mathcal{E}} \text{trace}\left\{\log^2(R_j^T R_i \tilde{R}_{ij})\right\}. \end{aligned} \tag{6.1}$$

It is a well-known fact that $SO(2) = \left\{R \in \mathbb{R}^{2\times 2} : R^T R = I,\ det(R) = +1\right\}$, the Lie group of rotations in 2D space, is isomorphic to the circle group $\mathbb{T} = \{z \in \mathbb{C} : |z| = 1\}$, i.e. $SO(2) \simeq \mathbb{T}$ [10]. Indeed, denoting with $\theta \in [-\pi, \pi)$ the angle of rotation, there exists the following biunivocal correspondence

$$\mathbb{T} \ni z = e^{j\theta} \quad \leftrightarrow \quad \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = R_\theta \in SO(2). \tag{6.2}$$

Exploiting this isomorphism existing in 2D case, it is immediate to realize that the geodesic distance between $R_{\theta_1}, R_{\theta_2} \in SO(2)$ coincides with the length of the arc that links the points $z_1 = e^{j\theta_1}, z_2 = e^{j\theta_2} \in \mathbb{T}$ set on the unit circle. Whereas, the Frobenius distance is represented by their chordal distance. A graphical interpretation of these relations is presented in Figure 6.1.

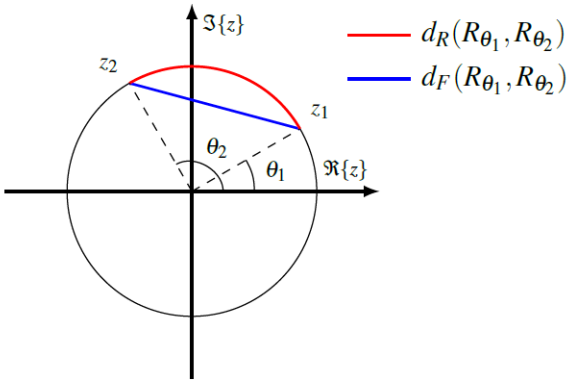


Figure 6.1.: Graphical representations of Riemannian and Frobenius distances on $SO(2) \simeq \mathbb{T}$

After these premises, it is natural to introduce θ_i as the angle of rotation given by the unknown matrix $R_i \in SO(2)$ and $\tilde{\theta}_{ij}$ as the angle of rotation of the noisy measurement given by $\tilde{R}_{ij}$. Hence, a single term of the cost functional (6.1) can be written as

$$\begin{aligned} f(\theta_i, \theta_j) &= -\frac{1}{2}\text{trace}\left\{\log^2(R_j^T R_i \tilde{R}_{ij})\right\} \\ &= -\frac{1}{2}\text{trace}\left\{\log^2\left(\begin{bmatrix} \cos(\theta_i - \theta_j) & \sin(\theta_i - \theta_j) \\ -\sin(\theta_i - \theta_j) & \cos(\theta_i - \theta_j) \end{bmatrix}\begin{bmatrix} \cos\tilde{\theta}_{ij} & \sin\tilde{\theta}_{ij} \\ -\sin\tilde{\theta}_{ij} & \cos\tilde{\theta}_{ij} \end{bmatrix}\right)\right\} \\ &= -\frac{1}{2}\text{trace}\left\{\log^2\left(\begin{bmatrix} \cos(\theta_i - \theta_j + \tilde{\theta}_{ij}) & \sin(\theta_i - \theta_j + \tilde{\theta}_{ij}) \\ -\sin(\theta_i - \theta_j + \tilde{\theta}_{ij}) & \cos(\theta_i - \theta_j + \tilde{\theta}_{ij}) \end{bmatrix}\right)\right\} \\ &= -\frac{1}{2}\text{trace}\left\{\begin{bmatrix} 0 & (\theta_i - \theta_j + \tilde{\theta}_{ij}) \\ -(\theta_i - \theta_j + \tilde{\theta}_{ij}) & 0 \end{bmatrix}^2\right\} = (\theta_i - \theta_j + \tilde{\theta}_{ij})^2. \end{aligned} \tag{6.3}$$

The very simple form assumed by $f(\theta_i, \theta_j)$ is obtained observing that the principal logarithm of a matrix $R_\theta \in SO(2)$ has a close-form expression, i.e.

$$\log R_\theta = \begin{bmatrix} 0 & \theta \\ -\theta & 0 \end{bmatrix}, \quad \theta \in [-\pi, \pi). \tag{6.4}$$

As a consequence, the summation (6.1) can be expressed as a function of the angles of rotation

$$\varphi_R = \frac{1}{2}\sum_{i=1}^{N}\sum_{(i,j)\in\mathcal{E}} f(\theta_i, \theta_j) = \frac{1}{2}\sum_{i=1}^{N}\sum_{(i,j)\in\mathcal{E}} (\theta_i - \theta_j + \tilde{\theta}_{ij})^2, \tag{6.5}$$

and the problem to tackle turns into minimizing the convex function (6.5) with the only constraint $\theta_i \in [-\pi, \pi)$, i. e.

$$\min_{\theta_i \in [-\pi,\pi)} \frac{1}{2}\sum_{i=1}^{N}\sum_{(i,j)\in\mathcal{E}} f(\theta_i, \theta_j). \tag{6.6}$$

To compute the global minimum (that corresponds to the local one), it is necessary to calculate the *gradient* and the *Hessian* of φ_R:

$$\nabla_i \varphi_R = \frac{\partial \varphi_R}{\partial \theta_i} = \sum_{(i,j)\in\mathcal{E}} (\theta_i - \theta_j + \tilde{\theta}_{ij}) - \sum_{(i,j)\in\mathcal{E}} (\theta_j - \theta_i + \tilde{\theta}_{ji}), \tag{6.7}$$

$$[H_{\varphi_R}]_{ij} = \frac{\partial \varphi_R}{\partial \theta_i \theta_j} = \begin{cases} 2\text{deg}(i) & \text{if} \quad i = j \\ -2 & \text{if} \quad i \neq j \text{ and } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{6.8}$$

It results that $H_{\varphi_R} = 2L$, i.e. the Hessian is twice the *Laplacian matrix* of the graph $\mathcal{G}$ associated to the network. The matrix L is one of the possible representation of $\mathcal{G}$ and it is positive semi-definite, as its eigenvalues are real and non-negative.

Therefore, the minimum can be found by imposing the gradient $\nabla_i \varphi_R$ equal to zero for all $i \in \mathcal{V}$, i.e. by solving the system of linear equations

$$\nabla_i \varphi_R = 0, \ \forall i \in \mathcal{V} \quad \Longrightarrow \quad 2L\boldsymbol{\theta} = \tilde{\boldsymbol{\theta}}, \tag{6.9}$$

where $\boldsymbol{\theta} \in \mathbb{R}^N$ is the column vector of the angles θ_i, while $\tilde{\boldsymbol{\theta}} \in \mathbb{R}^N$ is the one in which the quantities $\sum_{(i,j)\in\mathcal{E}} \tilde{\theta}_{ji} - \sum_{(i,j)\in\mathcal{E}} \tilde{\theta}_{ij}$ are stacked.

From linear equations theory, it is possible to demonstrate that if $\tilde{\boldsymbol{\theta}} \subseteq Im(L)$ then the Equation (6.9) always admits solutions. In this case, there are infinitely ones parametrized by

$$\boldsymbol{\theta}^\star = \frac{1}{2} L^\dagger \tilde{\boldsymbol{\theta}} + \frac{1}{2}(I - L^\dagger L)\boldsymbol{\chi}, \tag{6.10}$$

where $L^\dagger$ denotes the pseudoinverse of the matrix L, $\boldsymbol{\chi} \in \mathbb{R}^N$ and $(I - L^\dagger L)\boldsymbol{\chi} \in ker(L)$.

To conclude, it is essential to underline that $\boldsymbol{\theta}^\star = \frac{1}{2} L^\dagger \tilde{\boldsymbol{\theta}}$ represents the minimum norm solution but it is not always the best solution as it can be verified from the analysis of the following case study. The same considerations apply when considering the full $R \in SO(3)$ matrix but with only one degree of freedom θ.

6.1.1 Case study

Let us consider a 2D scenario with four cameras. Figure 6.2 displays the configuration of the network: the green triangles represent the cameras, while the blue lines identify the communication links between the devices.

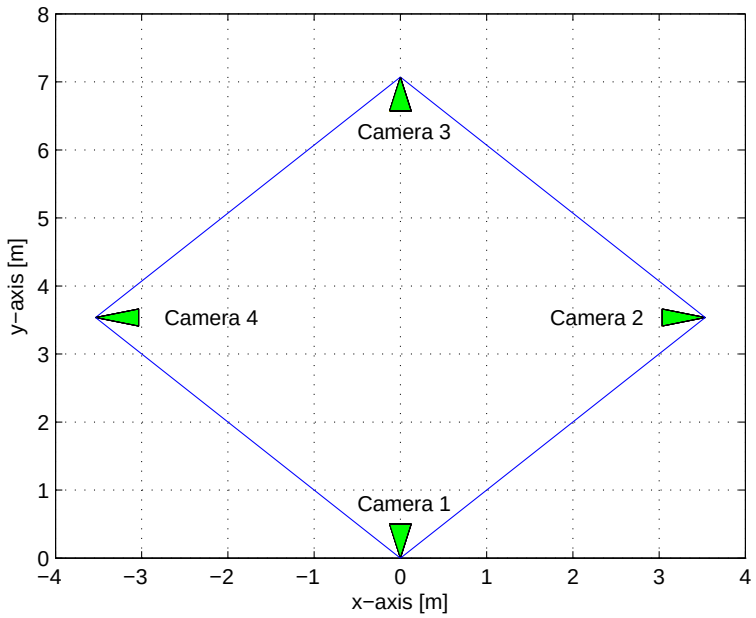


Figure 6.2.: 2D network

It can be observed that each element of the network is capable of exchanging information only with its neighbors, so the system is associated to a circulant graph. Moreover, the devices are placed with a particular symmetry as they are at the corners of a rhombus. Specifically, each camera is oriented in order to form angles of 90 degrees with its neighbors: the rotations have been defined to occur only on the xy plane and they can be identified by a unique angle, as suggested in the previous paragraph.

More specifically,

$$R_1 = \begin{bmatrix} \cos(0°) & \sin(0°) \\ -\sin(0°) & \cos(0°) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.11}$$

$$R_2 = \begin{bmatrix} \cos(-90°) & \sin(-90°) \\ -\sin(-90°) & \cos(-90°) \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{6.12}$$

$$R_3 = \begin{bmatrix} \cos(-180°) & \sin(-180°) \\ -\sin(-180°) & \cos(-180°) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{6.13}$$

$$R_4 = \begin{bmatrix} \cos(-270°) & \sin(-270°) \\ -\sin(-270°) & \cos(-270°) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{6.14}$$

Exploiting the relationships defined in Chapter 2 concerning the kinematics of the cameras, it is possible to determine the relative poses of the devices from the absolute ones that are manually imposed. In addition, thanks to the Matlab functionalities, the calculated relative poses can be made noisy by adding Gaussian noise with a certain variance to the relative angles. In this way, the relative noisy poses required as inputs in the algorithm described by Tron and Vidal are available.

It is therefore possible to implement the algorithm described in Chapter 4. The initialization is fulfilled by building a single spanning tree having node 1 as the root and two unbalanced branches made up of nodes 2-3 and node 4 respectively. The choice of using this method, based on the construction of a single spanning tree, is justified by the fact that it is the fastest strategy, moreover the purpose of the analysis is independent of the strategy used for the calculation of the initial absolute poses.

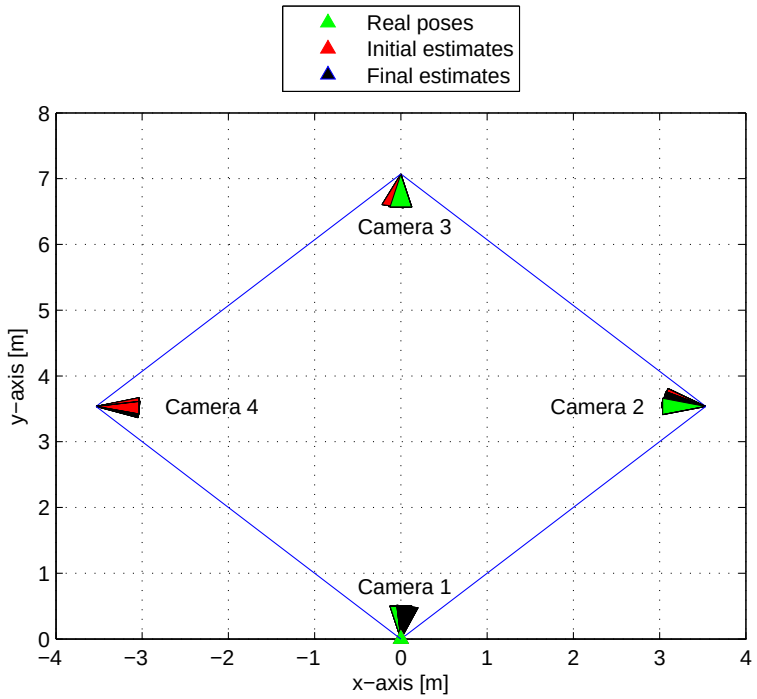


Figure 6.3.: Implementation of Tron-Vidal algorithm on 2D network

Figure 6.3 shows the result achieved by implementing the algorithm discussed earlier for the part related to rotation estimates, thus ignoring the translations and the scale factors which remain unaffected throughout the simulation. In order to get an accurate solution, the number of iterations to be performed is set to 1200, while the step-size ε is fixed to 0.01.

Indeed, the visual information provided by Figure 6.3 not is particularly meaningful: the initial, final and real poses slightly deviate as the generation of noisy poses has been obtained by considering a variance of only five degrees on the real relative angles.

In order to provide a measure of the effectiveness of the algorithm is useful to introduce the *mean error on rotations*, $\bar{e}_R$. This performance index is defined as

$$\bar{e}_R = \frac{1}{N} \sum_{i=1}^{N} \|R_i - \hat{R}_i\|_F, \tag{6.15}$$

where N is the number of cameras in the network, R_i denotes the real rotation of the i-th camera and $\hat{R}_i$ indicates the final estimated rotation of the i-th camera.

Similarly, the error can be evaluated individually for each camera rotation considering the Frobenius norm of the difference between the real measurement and the final estimate, i.e.

$$e_{R_i} = \|R_i - \hat{R}_i\|_F. \tag{6.16}$$

The performances of the algorithm in terms of errors are illustrated in Table 6.1: it can be highlighted a small improvement on $\bar{e}_R$ at the end of the execution.

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}
Initial estimates	0.185	0	0.440	0.275	0.025
Final estimates	0.173	0.213	0.295	0.054	0.128

Table 6.1.: Errors on rotations obtained implementing the Tron-Vidal algorithm on 2D network

The most significant result of ongoing analysis is the value assumed by the cost function at the end of the iterative procedure: φ_R is monotonically decreasing towards the convergence value 0.085 as shown in Figure 6.5.

It can also be followed the analytical reasoning exposed in the previous paragraph: to do so, it is necessary to define the values of the Laplacian matrix L and the vector $\tilde{\boldsymbol{\theta}}$. Explicitly,

$$L = \begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{\theta}} = \begin{bmatrix} \tilde{\theta}_{41} + \tilde{\theta}_{21} - \tilde{\theta}_{12} - \tilde{\theta}_{14} \\ \tilde{\theta}_{12} + \tilde{\theta}_{32} - \tilde{\theta}_{21} - \tilde{\theta}_{23} \\ \tilde{\theta}_{23} + \tilde{\theta}_{43} - \tilde{\theta}_{32} - \tilde{\theta}_{34} \\ \tilde{\theta}_{14} + \tilde{\theta}_{34} - \tilde{\theta}_{41} - \tilde{\theta}_{43} \end{bmatrix}.$$

It is useful to note that all the noisy relative angles $\tilde{\theta}_{ij}$ have a value of approximately ninety degrees, the angles $\tilde{\theta}_{41}$ and $\tilde{\theta}_{14}$ except since it is necessary to add $+360°$ and $-360°$ respectively because going along the network a full circle clockwise or counterclockwise is executed.

The optimal estimation is obtained by applying the formula (6.10), after converting the measures of the angles from degrees to radiants. It results

$$\boldsymbol{\theta}^{\star} = \begin{bmatrix} 2.386 \\ 0.871 \\ -0.871 \\ -2.387 \end{bmatrix} + \begin{bmatrix} -2.386 \\ -2.386 \\ -2.386 \\ -2.386 \end{bmatrix} = \begin{bmatrix} 0 \\ -1.515 \\ -3.257 \\ -4.773 \end{bmatrix} \text{ rad} \quad \Longrightarrow \quad \boldsymbol{\theta}^{\star} = \begin{bmatrix} 0 \\ -86.810 \\ -186.605 \\ -273.467 \end{bmatrix} \text{ deg} \qquad (6.17)$$

The vector $\boldsymbol{\theta}^{\star}$ is the sum of two components: the first one is the minimum norm solution, while the second term is a vector that belongs to the kernel of L and is chosen to reset the error on the rotation of the first camera.

Figure 6.4 reports the graphical comparison between the poses analytically computed and those estimated through the application of Tron-Vidal algorithm. Also in this case, the information given is not so significant as the results gained with two methods are very close.

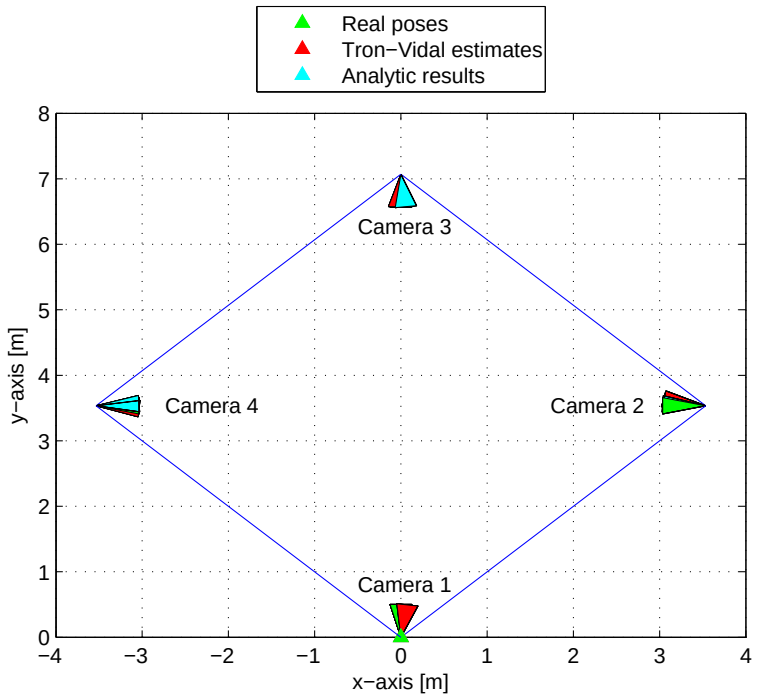


Figure 6.4.: Comparison between the results obtained by the analytic computation and by the implementation of Tron-Vidal algorithm on 2D network

Evaluating the cost function in correspondence to the solution (6.17) found, it is easy to verify that the analytical minimum value assumed by φ_R is equal to the convergence value achieved by applying the algorithm illustrated in Chapter 4, i.e. φ_R =0.085 (Figure 6.5). Approximately after 900 iterations of the Tron-Vidal algorithm the value of the cost function falls to a minimum, which coincides with the analytical value.

Nevertheless the errors $\bar{e}_R$ are different in the two cases, in the sense that the analytical calculation allows to obtain a final mean error that is smaller with respect to the algorithm as illustrated by Figure 6.6. Truthfully, this behavior is justified by the

fact that the errors on rotations are evaluated considering the Frobenius norm and the minimizing cost function heeds a measure of error based on the Riemannian metric.

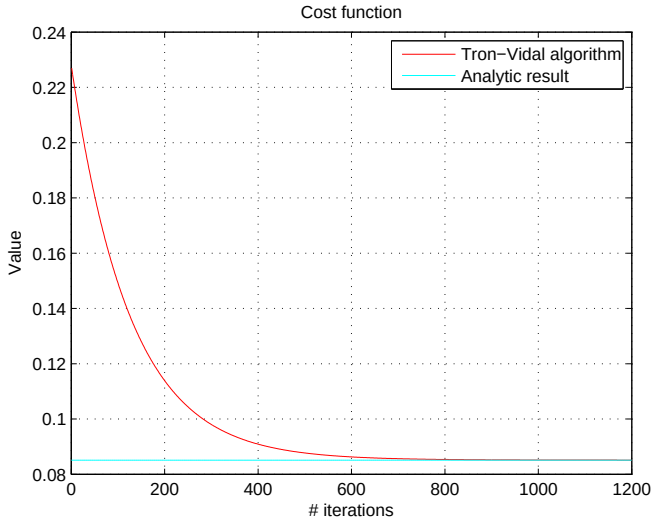


Figure 6.5.: Comparison of the trends of the cost function in the analytic computation and during the implementation of Tron-Vidal algorithm on 2D network

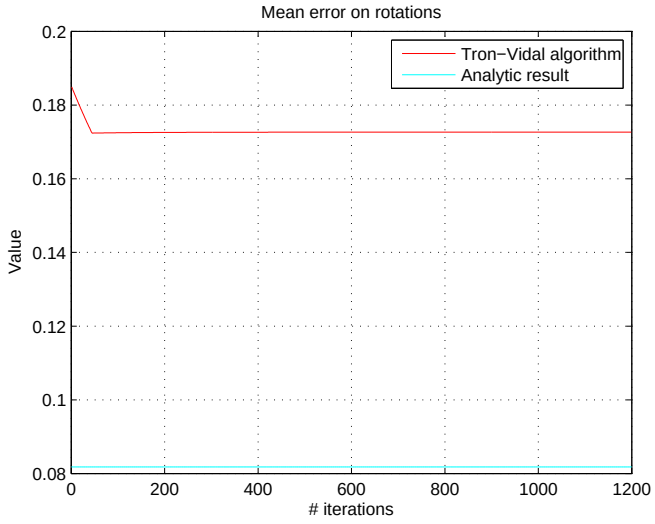
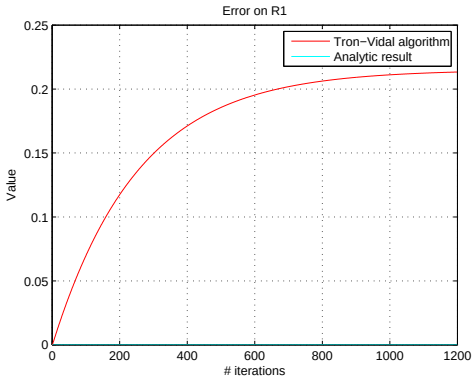


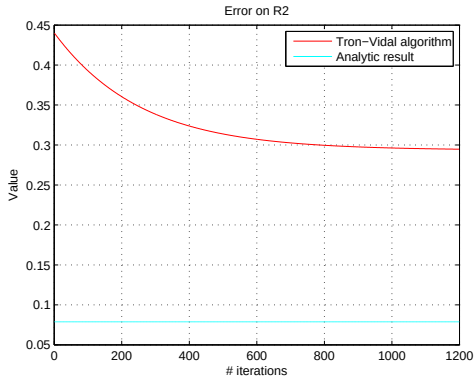
Figure 6.6.: Comparison of the trends of the mean error on rotations in the analytic computation and during the implementation of Tron-Vidal algorithm on 2D network

However, evaluating the error on the rotation of each camera the situation is different as indicated by Figures 6.7. It is above all interesting to observe the behavior of e_{R_1}. In the analytical case it is always null due to the choice of the additional term in the minimum norm solution; unfortunately, in the iterative solution the error grows from zero to 0.213 because of the initialization strategy. Moreover the error on R_4 follows an unusual behavior and is probably due to the corrective terms added on the relative angles $\tilde{\theta}_{41}$ and $\tilde{\theta}_{14}$.

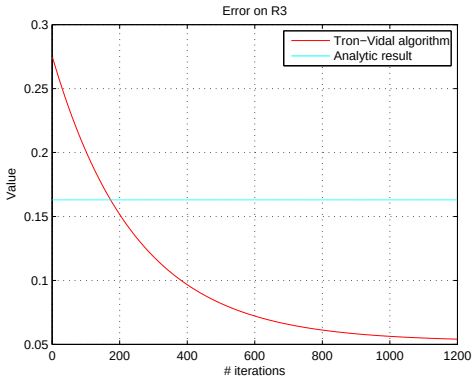
All the numerical values of the errors are illustrated in Table 6.2: for the algorithm implementation, only the final ones are reported.



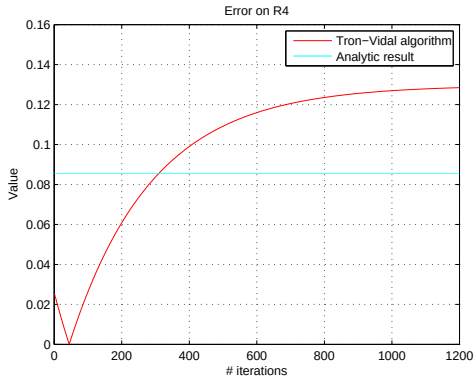
(a) Error on R_1



(b) Error on R_2



(c) Error on R_3



(d) Error on R_4

Figure 6.7.: Comparison of the trends of the errors on rotations in the analytic computation and during the implementation of Tron-Vidal algorithm on 2D network

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}
Tron-Vidal algorithm	0.173	0.213	0.295	0.054	0.128
Analytic results	0.082	0	0.079	0.163	0.086

Table 6.2.: Errors on rotations obtained though the analytical computation and implementing the Tron-Vidal algorithm on 2D network

6.2 Simulations

In this section, some simulations are carried out on different synthetic setups created in Matlab environment. The goal is to evaluate the performances of the algorithm introduced in Chapter 4. More specifically, the analysis focuses on the optimality of the found results.

For this reason, two performance metrics are considered: the *mean error* $\bar{e}_R$ *on rotations* and the *mean error* $\bar{e}_T$ *on translations*. The first one has been defined in the previous paragraph, the second one is expressed in analogous manner:

$$\bar{e}_T = \frac{1}{N} \sum_{i=1}^{N} \|T_i - \hat{T}_i\|. \tag{6.18}$$

In addition to the mean errors, it is also interesting to analyze the errors on the various poses singularly considered. For rotations, these indices have already been determined as the Frobenius norm of the difference between the real measurements and the final estimates. Concerning the translations, the error on $T_i,\ i \in \mathcal{V}$ is stated as

$$e_{T_i} = \|T_i - \hat{T}_i\|. \tag{6.19}$$

As far as the scale factors $\{\lambda_{ij}\}$ are concerned, paper [1] does not give any useful information about how to treat these parameters. Following the approach suggested in [10], in all the simulations the quantities λ_{ij} are initialized to 1 for all $(i,j) \in \mathcal{E}$ and then all the results are scaled by a factor λ=5 to be plotted. This choice enables to overtake the problem of scale ambiguity by assuming that all cameras are equidistant from the observed scene.

The analysis begins with an assessment of the performances of the different initialization methods. More specifically, a comparison is performed between

1. single spanning tree method (SST) and multi spanning trees method (MST) (6.2.1);
2. single spanning tree method (SST) and multi spanning trees method with virtual camera (MSTVC) (6.2.2);
3. multi spanning trees method (MST) and multi spanning trees method with virtual camera (MSTVC) (6.2.3).

Then, the simulations are carried out to evaluate the performance of the entire iterative procedure presented in Chapter 4. The aim is to observe the algorithm robustness with respect to different conditions:

- the algorithm is initialized with a single spanning tree method or with one of the multi spanning trees strategies (6.2.4);
- the initial noisy relative poses have very different levels of uncertainty; (e.g. the relative pose between camera 1 and 2 is very noisy, while the others are less noisy)(6.2.5);
- the network has a topology more complex than those considered so far, i.e. the associated graph is not circulant because the devices can communicate also with nodes that are not their neighbors (6.2.6);
- the value of step-size ε varies in a certain range (6.2.7).

6.2.1 Initialization methods comparison: SST vs MST

In this paragraph, the goal is to evaluate the performances of SST method and MST method applied to the same network.

Let us consider the camera system introduced in the previous case study (see Figure 6.2), with the same agents displacement and communication links; however a 3D space is now considered where the rotations still occur in a single plane, while the translations are free to move in three dimensions.

Both initialization algorithms require the knowledge of the noisy relative poses. In Matlab environment, these ones can be computed starting from the absolute poses (manually imposed) to which Gaussian noise is added. In this simulation, the variance is imposed to 2 degree for the rotations; instead, it is set equal to 0.1 meters for the translations. It is important to emphasize that the noisy relative poses are generated in order to satisfy the assumption required by the MST method, i.e. $\tilde{g}_{ij} \neq \tilde{g}_{ji}^{-1}$, $\forall\, i, j \in \mathcal{V}$.

The primary step in both the procedures is the construction of the spanning trees. In all the cases node 1 is chosen as root: in this way, the position of the observer coincides with camera 1 for both the initialization strategies. In the SST case, the resulting tree has two unbalanced branches: the left one is formed by nodes 2 and 3, while the right includes only the node 4. Concerning the MST method, it is useful to refer to Figure 4.3 since the network considered has the same topology as that in the example given in Chapter 4. Figure 6.8 shows all the absolute poses calculated for the various cameras in the network, as well as the results obtained by averaging them. It is important to remember that the mean of the translations is arithmetically calculated, while the computation of the rotation one is performed according to Algorithm 1.

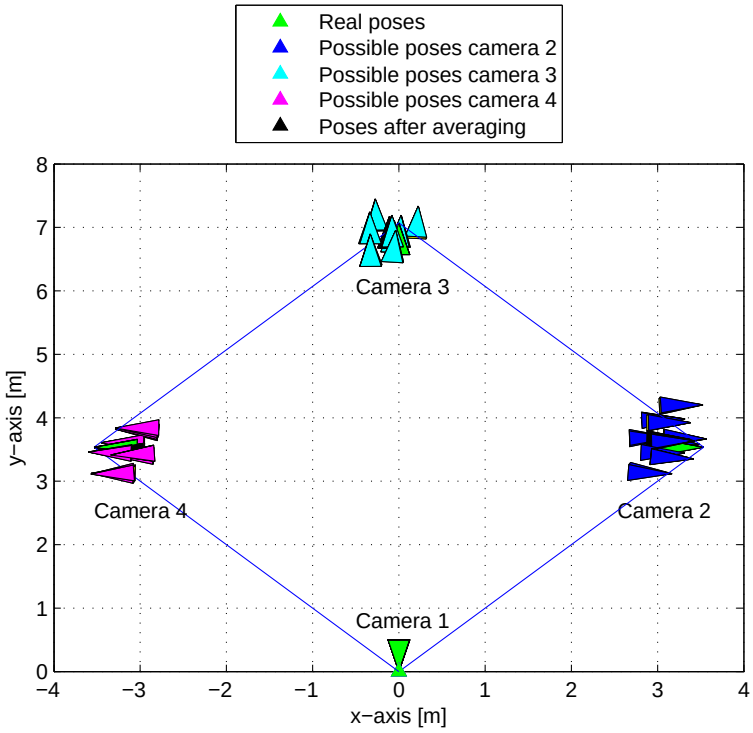


Figure 6.8.: All possible camera poses calculated through MST method

Figure 6.9 displays the results obtained with the two different initialization methods: it can be observed that the estimated poses are pretty similar in the two cases and they are also very close to the real ones according to the errors reported in Table 6.3.

The multi spanning trees procedure allows to obtain the best estimates regarding both rotations and translations, in fact the mean errors $\bar{e}_R$ and $\bar{e}_T$ are smaller than in the SST case.

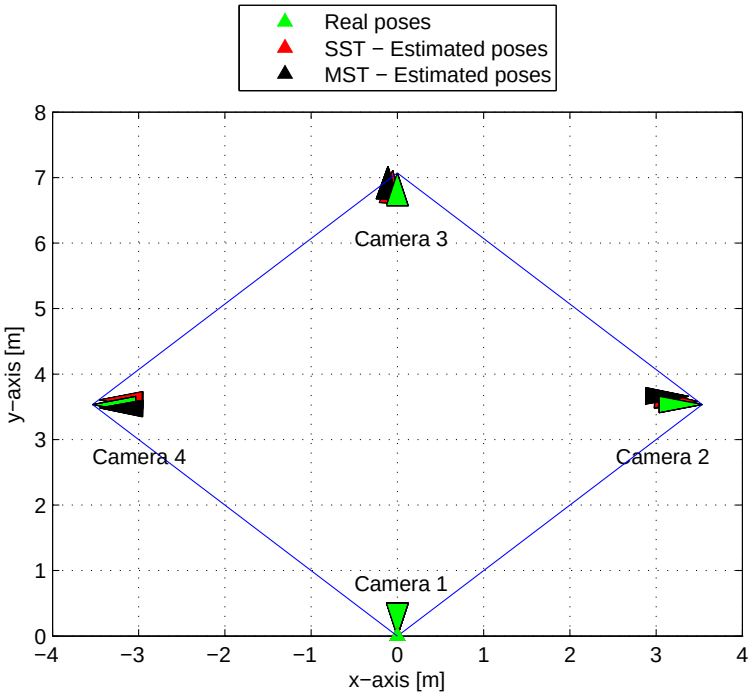


Figure 6.9.: Comparison of SST and MST results

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}
SST method	0.063	0	0.097	0.101	0.053	0.170	0	0.172	0.224	0.285
MST method	0.029	0	0.036	0.037	0.043	0.118	0	0.210	0.161	0.100

Table 6.3.: Errors on rotations and translations obtained applying SST and MST methods

Table 6.3 also shows the values of the errors calculated on the poses of the single cameras. It can be noted that in the SST case the error on rotations increases moving away from the root. This behavior does not emerge by assessing the errors on translations: a possible explanation lies in the definition of noisy relative poses whose uncertainty is generated in Gaussian manner. Finally, it is necessary to emphasize that the error on the pose of the camera 1 is always null since it is chosen as reference in both the initialization strategies. As far as the SST method is concerned, it is necessary to underline that camera 1 has a particular orientation and position that coincide with the initial conditions imposed for the root of the spanning trees, while in MST case, its absolute pose is assumed to be known.

6.2.2 Initialization methods comparison: SST vs MSTVC

In this paragraph, the SST strategy is compared with the MSTVC one: the scenario considered is different from that of the previous comparison because of the introduction of a virtual camera in the network that defines a new position for the observer in the

scene.

Let us consider again a network similar to that of Figure 6.2. In Figure 6.10 the real cameras are represented by green triangles, whereas the blue lines define the communication pattern. The yellow triangle represents the virtual camera that is placed outside the network in the lower left corner. The magenta lines indicate the characteristic property of the additional artificial node: the capacity of communicating with all the other devices in the network. This feature of the virtual camera corresponds to the knowledge of the relative poses between it and all the other agents. It is worth of notice the positions of the real cameras in the environment: they are translated with respect to Figure 6.2. Specifically, the particular position of the camera 1 is now occupied by the virtual camera.

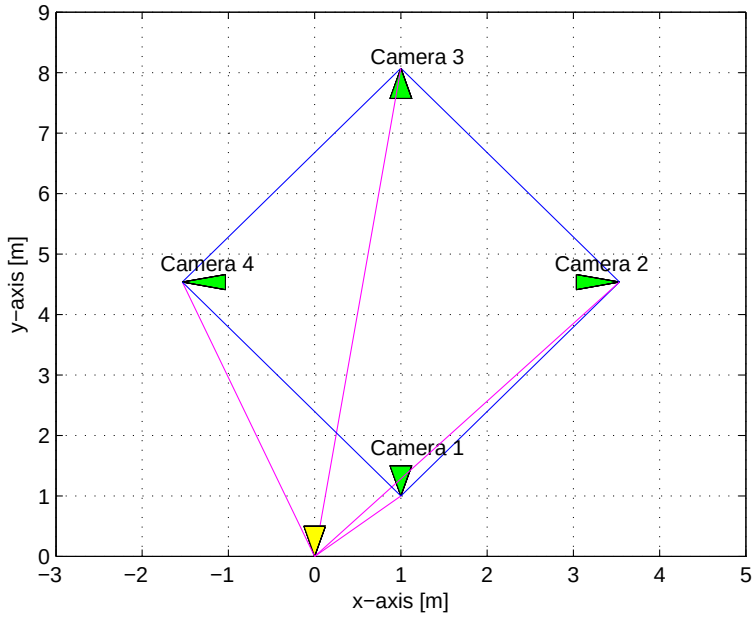


Figure 6.10.: 3D camera network with artificial node

As the SST strategy, the MSTVC method requires the noisy relative poses between the real cameras as inputs, in addition it also needs those between the artificial and the real devices. The former are determined as described in the previous paragraph, keeping the same values of the variances; the latter are calculated using the pose of the virtual camera without addicting noise.

The basic step of the construction of spanning trees is carried out in a different manner for the two methods. In the SST case, the configuration is the same as described in the previous comparison: the node 1 is chosen as root, while the nodes 2 and 3 constitute the left branch and the node 4 represents the right branch. The method MSTVC expects to build more trees possibly balanced having all the artificial node as root. The configurations considered in this case are those in Figure 6.11.

The absolute poses estimated at the end of both initialization procedures are shown in Figure 6.12. It can be observed that the results achieved through the SST method are more satisfying. This fact is partially confirmed by the numerical values of the errors reported in Table 6.4.

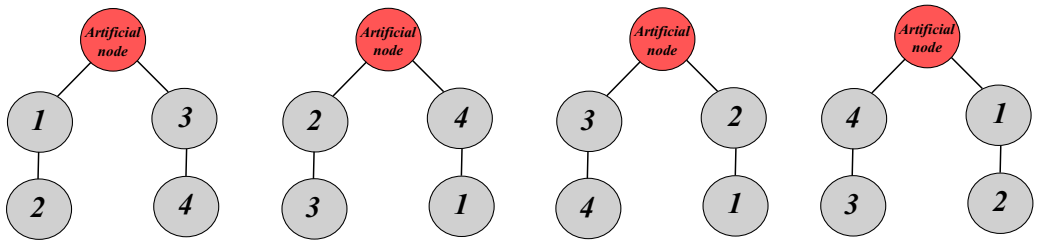


Figure 6.11.: Spanning trees - MSTVC strategy

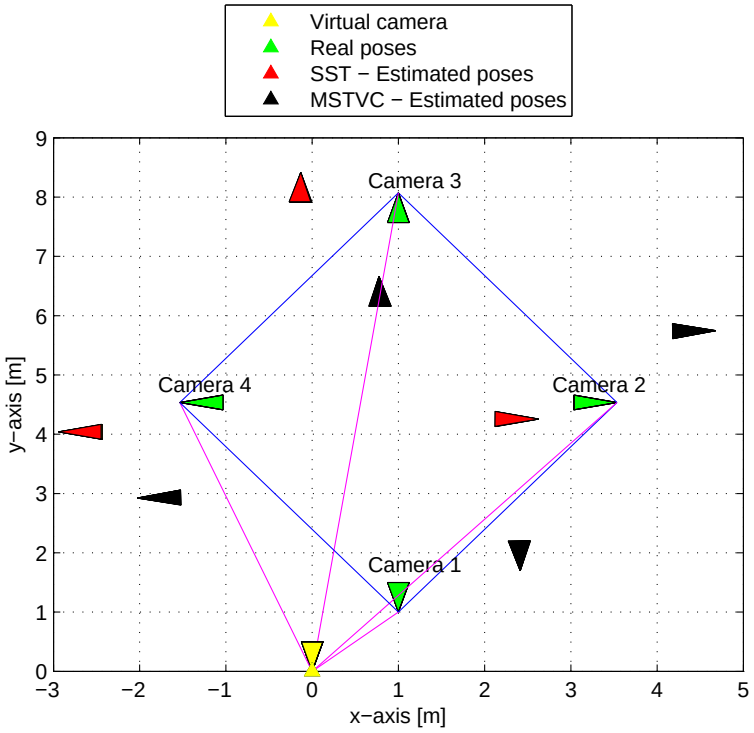


Figure 6.12.: Comparison of SST and MSTVC results

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}
SST method	0.061	0	0.056	0.103	0.084	1.270	1.414	0.954	1.209	1.501
MSTVC method	0.051	0.058	0.030	0.062	0.056	1.590	1.577	1.668	1.429	1.687

Table 6.4.: Errors on rotations and translations obtained applying SST and MSTVC methods

The low resolution of Figure 6.12 does not permit to grasp the small improvement in the estimation of rotations relative to MSTVC method. As far as the translations are concerned, the initialization based on the construction of a single spanning tree leads to a lower mean error. Nevertheless, it is important to emphasize that the MSTVC strategy allows to uniformly distribute the uncertainty as clearly revealed by the errors on the single T_i. Finally, it is interesting to note that e_{R_1}, unlike e_{T_1}, continues to be null in the SST case.

This effect is explained by considering that camera 1 is chosen as a root in the spanning tree but is shifted in position with respect to the zero of the plan (virtual camera position), so the initialization specified by the method obviously involves an error. This observation also explains the overall error increment on translations with respect to the previous paragraph.

6.2.3 Initialization methods comparison: MST vs MSTVC

In the last comparison on the initialization methods the attention is focused on the performance obtained by using or not the artificial node in the MST approach.

Let us still consider the network in Figure 6.10 in which all the real cameras communicate with the virtual one. The estimated poses through the MST and MSTVC methods are shown in Figure 6.13. It is worth underlining that the variances used in the calculation of relative poses are equal to those of the previous cases, while the spanning trees considered are those of Figure 4.3 for the MST approach and of Figure 6.11 for the MSTVC strategy.

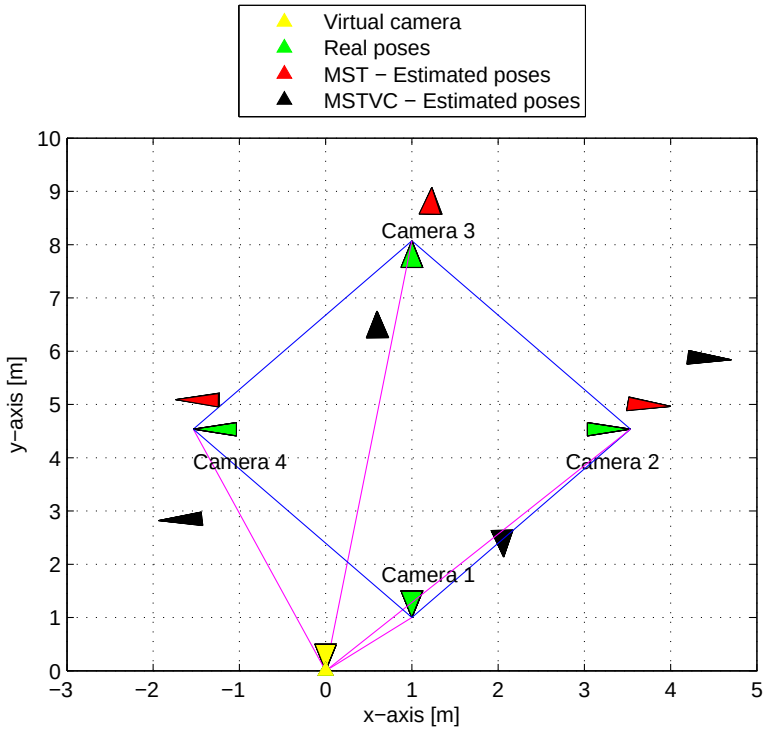


Figure 6.13.: Comparison of MST and MSTVC results

The obtained results are consistent with the observations previously made: visibly the addiction of a virtual node implies additional error. Table 6.5 confirms this fact.

Both the estimates of the rotations and translations are best in the MST case. However, it is important to emphasize that e_{R_1} and e_{T_1} are null using the MST strategy only because the node 1 is chosen as a reference and its absolute pose is assumed to be known.

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}
MST method	0.086	0	0.146	0.100	0.097	0.572	0	0.658	1.035	0.594
MSTVC method	0.096	0.077	0.157	0.043	0.107	1.615	1.561	1.753	1.378	1.765

Table 6.5.: Errors on rotations and translations obtained applying MST and MSTVC methods

6.2.4 Algorithm implementation: different initialization methods

After having assessed the results obtained by the different initialization methods, the attention here is on the analysis of the final poses estimates which are computed at the end of the whole procedure described in Chapter 4. More specifically, the aim is to compare the final mean errors among them and in relation to the initial ones for the different initialization strategies. Considering the networks of Sections 6.2.1, 6.2.2 and 6.2.3, the analysis is carried out according to the order of the preceding paragraphs.

Therefore, the plot of Figure 6.14 provides a graphical comparison between the final estimates found after the application of the Tron-Vidal algorithm initialized through the SST and MST methods. In particular, the number of iterations for the computation of rotations and translations estimate is set to 250 in both the cases, while the last step of estimates refinement is avoided as it has been empirically proved that it does not improve the results already obtained. The step-size parameter ε is fixed to 0.01.

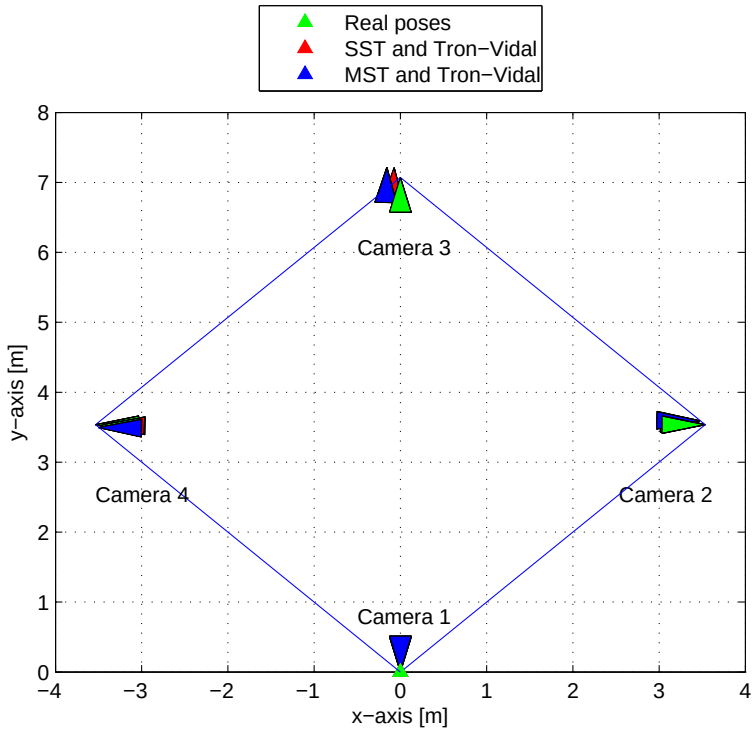


Figure 6.14.: Comparison of the final results achieved by Tron-Vidal algorithm initialized through SST and MST methods

The poses calculated in the two cases are very close to each other and to the real ones. Greater insight on the results is provided by Table 6.6. It reports the final mean error and the initial ones in order to evaluate which case seems to be more effective. It is easy to verify that the final mean errors are smaller when the Tron-Vidal algorithm is initialized by the MST procedure, although $\bar{e}_R$ does not improve with respect to the initial value and $\bar{e}_T$ increases. On the other hand, in the case of SST initialization, the mean error decreases for rotations but increases for the translations.

		$\bar{e}_R$	$\bar{e}_T$
SST and Tron-Vidal	initial estimates	0.063	0.170
	final estimates	0.059	0.193
MST and Tron-Vidal	initial estimates	0.029	0.118
	final estimates	0.029	0.127

Table 6.6.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm initialized though SST and MST methods

Figure 6.15 compares the final estimates attained by implementing the iterative procedure outlined in Chapter 4 starting from the poses computed though the SST and MSTVC methods. The network considered is that of Figure 6.10 and the number of iterations performed by Tron-Vidal algorithm is equal to the previous case (250), as well as the value of ε (0.01).

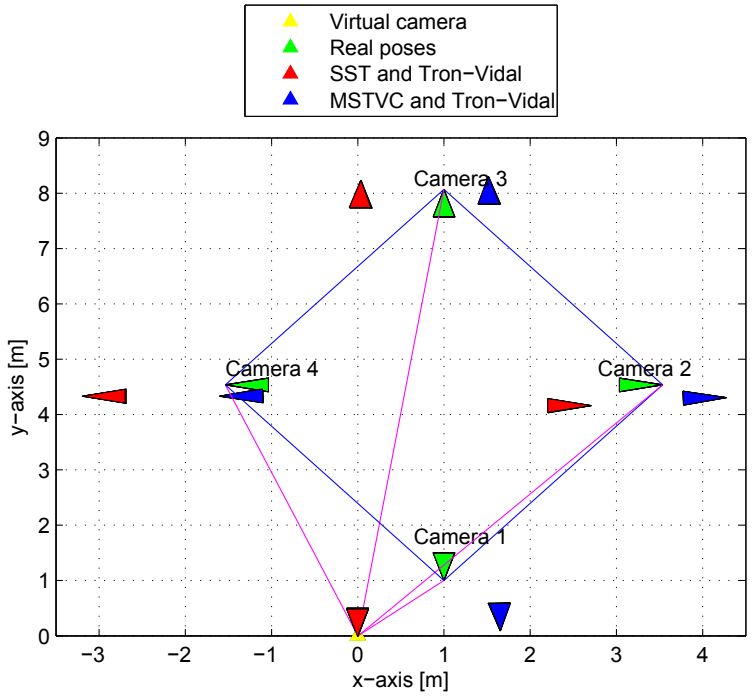


Figure 6.15.: Comparison of the final results achieved by Tron-Vidal algorithm initialized through SST and MSTVC methods

Observing the plot, it is easy to understand that once again the initialization strategy based on multiple spanning trees allows to obtain the best final estimates.

This fact is confirmed by the values of the mean errors in Table 6.7. The mean error on rotations remains almost unchanged in both the cases and it is lower after the initialization with the MSTVC method. The main difference is related to translations: when the algorithm is initialized though the SST strategy the estimates of the translations improves but the value of $\bar{e}_T$ remains high; on the contrary, it decreases substantially when the multi spanning trees approach is employed. Probably, this behavior is due to the fact that the method based on multi spanning trees initially distributes the uncertainty in a uniform manner. Therefore in the course of the algorithm the errors on the single poses are mitigated. Moreover, a not negligible factor is the position of the observer set in the virtual camera and not in the camera 1 as previously.

		$\bar{e}_R$	$\bar{e}_T$
SST and Tron-Vidal	initial estimates	0.061	1.270
	final estimates	0.061	1.256
MSTVC and Tron-Vidal	initial estimates	0.051	1.590
	final estimates	0.051	0.673

Table 6.7.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm initialized through SST and MSTVC methods

Finally, the analysis focuses on the comparison of the final estimates which are obtained when the algorithm proposed by Tron and Vidal is initialized through the MST and MSTVC methods.

The results are depicted in Figure 6.16: keeping the step-size fixed to 0.01 and performing 250 iterations for both the estimation of rotations and translations, the best poses seem to be achieved with the strategy that exploits the virtual camera.

Truthfully, Table 6.8 shows that the final mean errors are very similar using the two different initialization methods. Nevertheless, the greater improvement is obtained in the estimation of translations when the algorithm is initialized through the MSTVC strategy: the value of $\bar{e}_T$ decreases from 1.615 to 0.575.

		$\bar{e}_R$	$\bar{e}_T$
MST and Tron-Vidal	initial estimates	0.086	0.572
	final estimates	0.086	0.621
MSTVC and Tron-Vidal	initial estimates	0.096	1.615
	final estimates	0.081	0.575

Table 6.8.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm initialized with MST and MSTVC methods

In conclusion, it is possible to affirm that the algorithm presented in Chapter 4 seems to have better results when initialized through the multi spanning trees method with the introduction of the virtual camera as reference.

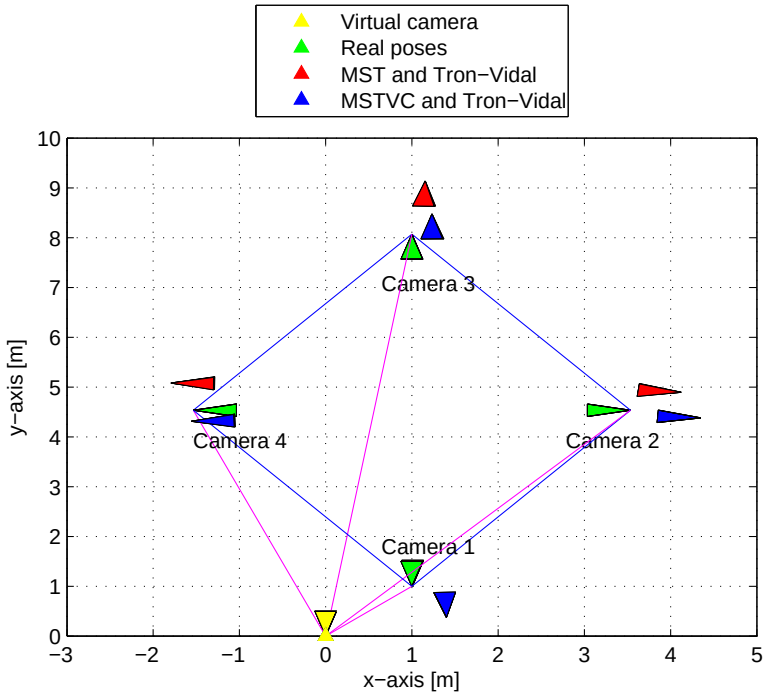


Figure 6.16.: Comparison of the final results achieved by Tron-Vidal algorithm initialized through MST and MSTVC methods

However, it is necessary to make some clarifications about the errors reported in Tables 6.6, 6.7 and 6.8. The values are different in each comparison even if the initialization method is the same: the considered achievements, in fact, are different because of the noise distribution and virtual node presence. Furthermore, the numerical results are related to a single simulation although they are consistent with a high number of tests.

6.2.5 Algorithm implementation: noise effect

In this paragraph, it is assumed that one of the initial relative poses in input to the procedure described in Chapter 4 is affected by higher uncertainty than the others. The purpose is to evaluate how this affects the results obtained at the end of the Tron-Vidal algorithm. In particular, the focus is centered on the differences between the poses estimated after the initialization phase and the final ones achieved varying the number of iterations performed by the estimation procedure.

It is important to emphasize that the algorithm is initialized with the MSTVC method as the results gained by the previous analysis seems to show that this is the best approach in terms of mean error on rotations and translations.

Let us consider the network displayed in Figure 6.10. The initial relative poses between the cameras are generated in Matlab environment starting from the real ones, manually imposed, through the addition of Gaussian noise. However, the variance on rotations is set to 2 degrees and that on translations is fixed to 0.1 meters for all the poses except for

$\tilde{g}_{12}$ and $\tilde{g}_{21}$. Indeed, it is presumed that the initial relative poses between the cameras 1 and 2 and viceversa are more noisy than the others: the measurements of the variances are quintupled. It is predictable that these initial conditions lead to a less accurate estimate of the absolute pose for the devices 1 and 2. This intuition is confirmed by the following plots (Figures 6.17 and 6.18).

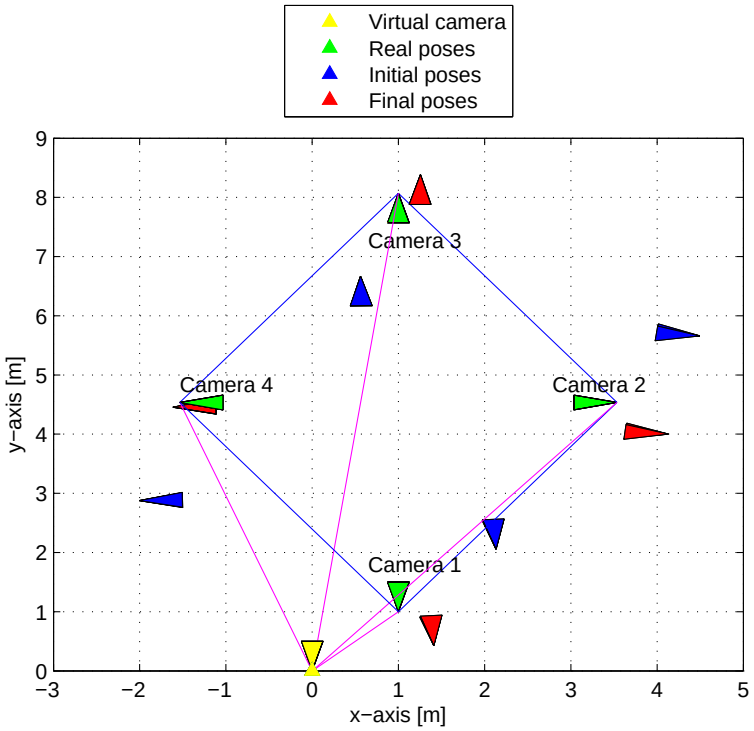


Figure 6.17.: Effect of unbalanced noise distribution on the results of Tron-Vidal algorithm (250 iterations)

Figure 6.17 presents the results achieved after the initialization procedure based on the construction of multiple spanning trees with the virtual camera as reference, but also the poses estimated at the end of the algorithm execution. The number of performed iterations is set to 250 for both the rotation and the translation part. Once again, the last step of the optimization on the whole pose is neglected and the step-size is set to 0.01.

It is easy to notice that the initial poses represented by blue triangles differ from the real ones (green triangles) more or less equally for all the cameras. The situation changes for the final estimates (red triangles): the errors on the poses are greater in correspondence to the devices 1 and 2. To quantitatively evaluate this change, it is useful to refer to Table 6.9.

The mean errors $\bar{e}_R$ and $\bar{e}_T$ decrease by implementing the algorithm, however, considering the errors on the pose of each camera, the information derived from Figure 6.17 is confirmed. In fact, the errors on T_1, T_2 have an order of magnitude similar to all the others at the end of the initialization step, while they are grater than the mean value in relation to the final estimates. The same behavior is partially shown by the errors on

R_1 and R_2, in particular e_{R_1} increases after the application of Tron-Vidal algorithm.

These observations are justified by the fact that the MSTVC method is based on the average of the absolute poses obtained by building multiple spanning trees having the same root: the uncertainty is distributed in a uniform manner across all the nodes in the network, above all for the translations. Instead, the next steps of the Tron-Vidal algorithm involve again the initial noisy measures, therefore the estimates of the poses of the cameras 3 and 4 generally improve than those of the cameras 1 and 2.

	$\bar{e}_R$	$\mathbf{e}_{R_1}$	$\mathbf{e}_{R_2}$	e_{R_3}	e_{R_4}	$\bar{e}_T$	$\mathbf{e}_{T_1}$	$\mathbf{e}_{T_2}$	e_{T_3}	e_{T_4}
Initial poses	0.125	**0.137**	**0.269**	0.041	0.052	1.559	**1.551**	**1.493**	1.472	1.723
Final poses	0.112	**0.164**	**0.197**	0.049	0.041	0.537	**0.711**	**0.802**	0.427	0.206

Table 6.9.: Errors on rotations and translations obtained applying Tron-Vidal algorithm in presence of unbalanced noise distribution (250 iterations)

Remarkably, the highlighted differences about the behavior of the estimates are accentuated when the number of iterations performed by the Tron-Vidal algorithm grows. Figure 6.18 displays the results that are achieved when it is increased to 400 for both rotations and translations: the red triangles of the final estimates of the cameras 1 and 2 are very distant from the green ones that represent the real poses. On the contrary, the initial estimates (blue triangles) are all equally far.

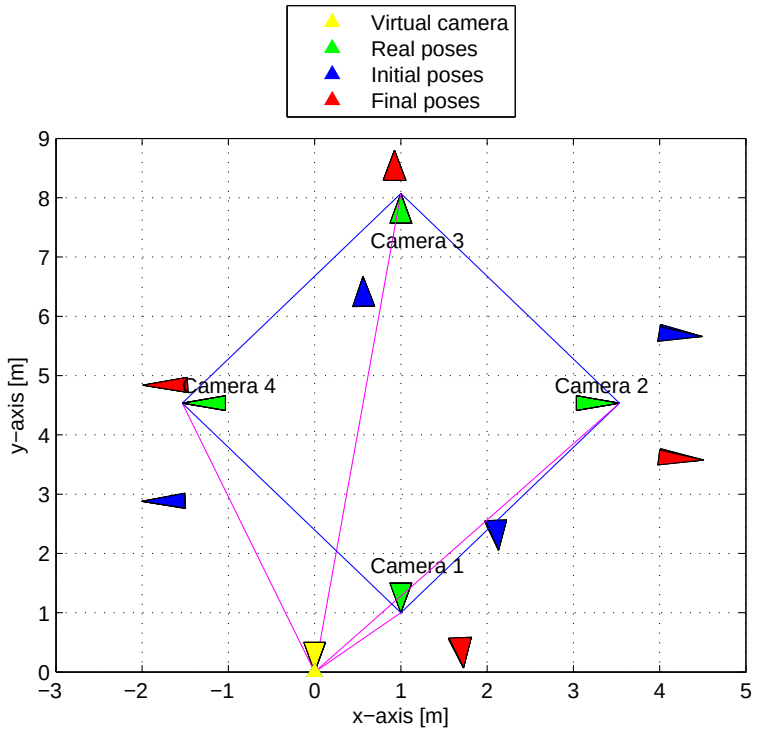


Figure 6.18.: Effect of unbalanced noise distribution on the results of Tron-Vidal algorithm (400 iterations)

These observations are confirmed again by the values of the errors reported in Table 6.10. Especially concerning the translations, the difference in the distribution of the uncertainty between the initial and final poses becomes greater: the errors on the poses of the cameras 1 and 2 become more substantial in relation to the ones of the others devices.

In particular, a high number of iterations entails a deterioration of the estimates of all poses: the mean errors increase compared to the ones of the previous case. The reason is that the final estimates are now more influenced by the noisy initial measures.

	$\bar{e}_R$	$\mathbf{e}_{R_1}$	$\mathbf{e}_{R_2}$	e_{R_3}	e_{R_4}	$\bar{e}_T$	$\mathbf{e}_{T_1}$	$\mathbf{e}_{T_2}$	e_{T_3}	e_{T_4}
Initial poses	0.125	**0.137**	**0.269**	0.041	0.052	1.559	**1.551**	**1.493**	1.472	1.723
Final poses	0.125	**0.170**	**0.202**	0.065	0.060	0.969	**1.184**	**1.371**	0.750	0.572

Table 6.10.: Errors on rotations and translations obtained applying Tron-Vidal algorithm in presence of unbalanced noise distribution (400 iterations)

It is interesting to evaluate the trend of the cost function which regulates the algorithm convergence in the second case considered, i.e. when the number of iterations is set to 400. Although the resolution of Figure 6.19 is rather low, thanks to the Matlab functionalities, it is possible to analyze the values of the two components φ_R and φ_T.

The rotation part constantly decreases even if in the last iterations the trend seems to become steeper. The behavior of translation part is much more interesting and less intuitive: the function decreases to the minimum value but then grows of a few cents confirming the observations about the errors.

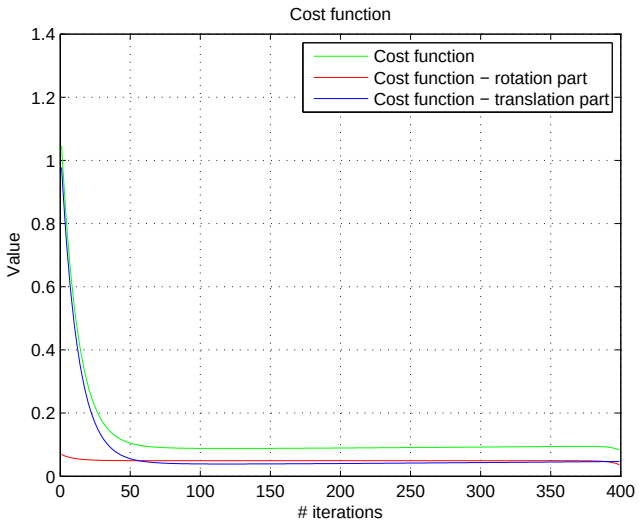


Figure 6.19.: Effect of unbalanced noise distribution on the trend of the cost function during the implementation of Tron-Vidal algorithm (400 iterations)

6.2.6 Algorithm implementation: additional communication links

In this paragraph, the simulations are carried out considering a more complex network topology: the aim is to analyze how the results of Tron-Vidal algorithm change when additional communication links are inserted in the configuration of Figure 6.10.

Let us consider the network displayed in Figure 6.20. The arrangement of the real and artificial cameras are unchanged from the figure cited above; nonetheless, the nodes 1 and 3 are now connected by a blue line, similarly 2 and 4: these devices therefore are considered able to communicate between them.

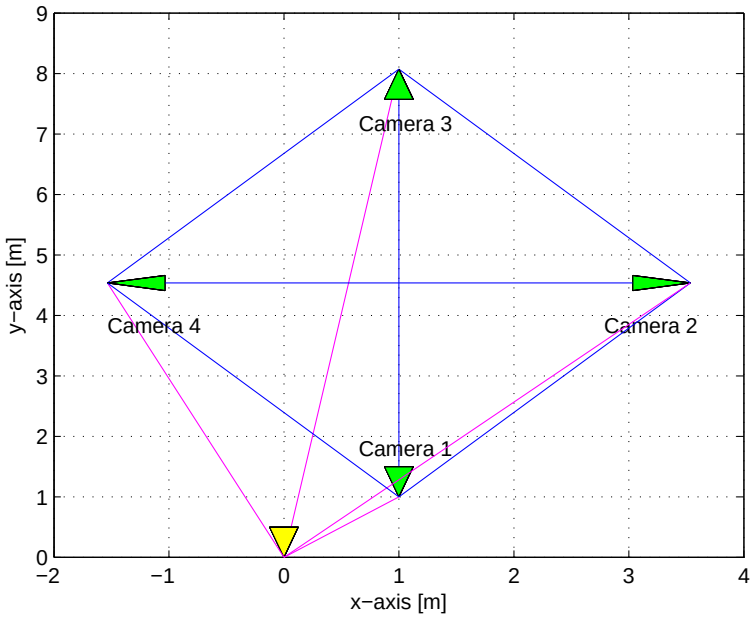


Figure 6.20.: 3D camera network with additional communication links

The communication pattern is different if compared to the previous cases: consequently also the graph associated to the network and the corresponding adjacency matrix differ. Previously, the agents were capable of exchanging information only with their neighbors, while now each camera interacts with all the other ones.

In graph theory, the considered network translates into a complete graph represented by an adjacency matrix having all unitary elements except for the zeros on the main diagonal (no self-loops).

Figure 6.21 shows the comparison between the results that are obtained by applying the algorithm described in Chapter 4 to this new configuration and to the standard one without the links between the cameras 1 and 3, 2 and 4.

It is worth noticing that the initialization is done with the MSTVC method, in particular the spanning trees considered are those of Figure 6.11. Moreover, the number of iterations to be performed is set to 250 for the estimation of the rotations, 250 for estimating the translations and 0 for the final refinement of the complete estimate. In closing, the value attributed to the step-size is ε=0.01.

In general, it is easy to verify that the final estimates of the camera poses deviate significantly more from the real ones if additional links are considered.

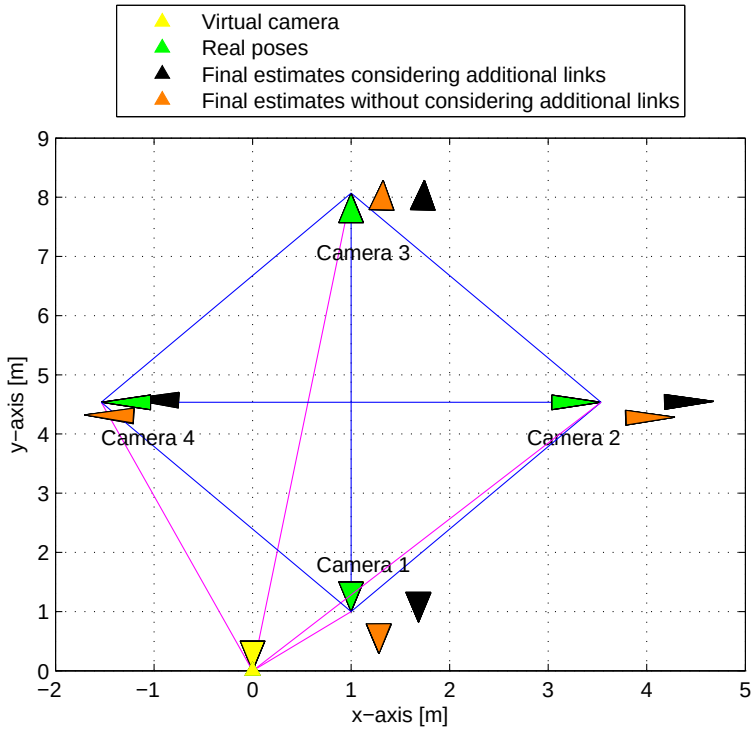


Figure 6.21.: Comparison between the results obtained applying Tron-Vidal algorithm to the network with and without additional links

This observation is confirmed by the values of the mean errors on rotations and translations accessible in Table 6.11.

The errors $\bar{e}_R$ and $\bar{e}_T$ increase their value when the Tron-Vidal algorithm is implemented on the configuration of Figure 6.20 rather than on that of Figure 6.10.

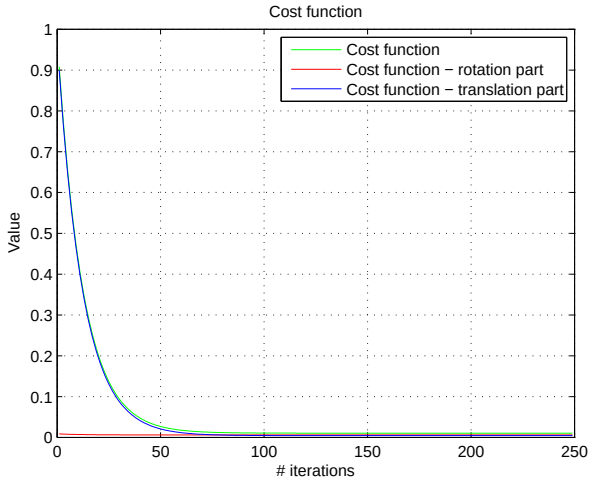
	$\bar{e}_R$	$\bar{e}_T$
Final poses considering additional links	0.074	0.874
Final poses without considering additional links	0.069	0.702

Table 6.11.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm to the network with and without additional communication links

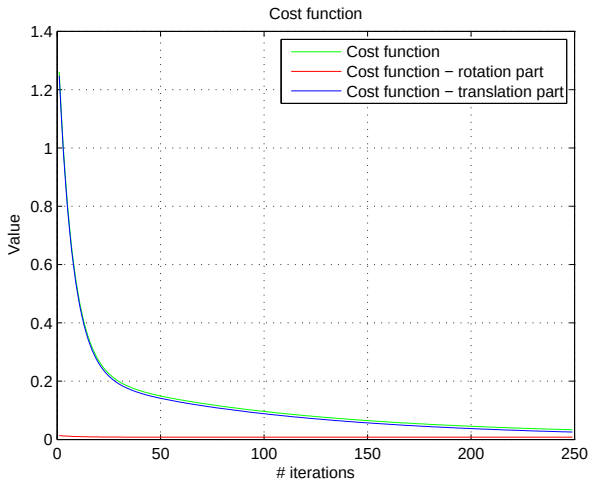
These observation are endorsed by the analysis of the trend of the cost function during the implementation of Tron-Vidal algorithm on the different network configurations.

Figures 6.22 (a) and (b) show the differences in the trend of φ (green line) expressed as the sum of its rotation (red line) and translation part (blue line). It is interesting to observe that the initial value assumed by the function cost is not the same in the two plots, moreover the decrease of φ_T is evidently different: for these reasons the minimum achieved by φ is 0.010 in Figure (a) and 0.033 in (b).

To improve the results in the case of a topology like that of Figure 6.20, it is possible to proceed as follows.



(a) Network without additional links



(b) Network with additional links

Figure 6.22.: Comparison of the trends of the cost function during the implementation of Tron-Vidal algorithm on the network with and without additional links

Initially, it is convenient to divide the network into subgraphs:

- ignoring the link that connects the nodes 2 and 4, it is possible to evaluate the sub-networks formed by the cameras 1-2-3 and 1-3-4, respectively denominated the *right sub-network* and the *left sub-network*;
- similarly, the *upper sub-network* and the *lower sub-network* are identified neglecting the link between the cameras 1 and 3, therefore the former is composed by the nodes 2-3-4 and the latter is made of the nodes 1-2-4;
- finally, the *external sub-network* is considered: it coincides with the standard configuration without any additional links.

The idea is to implement the Tron-Vidal algorithm separately on the subgraphs in order to reduce the symmetry of the system and then to average the absolute poses derived for the devices.

Let us consider then the situation illustrated in Figure 6.23, where the dashed lines indicate the subgraphs just defined. It is worth highlighting that the virtual camera is assumed present in all the cases.

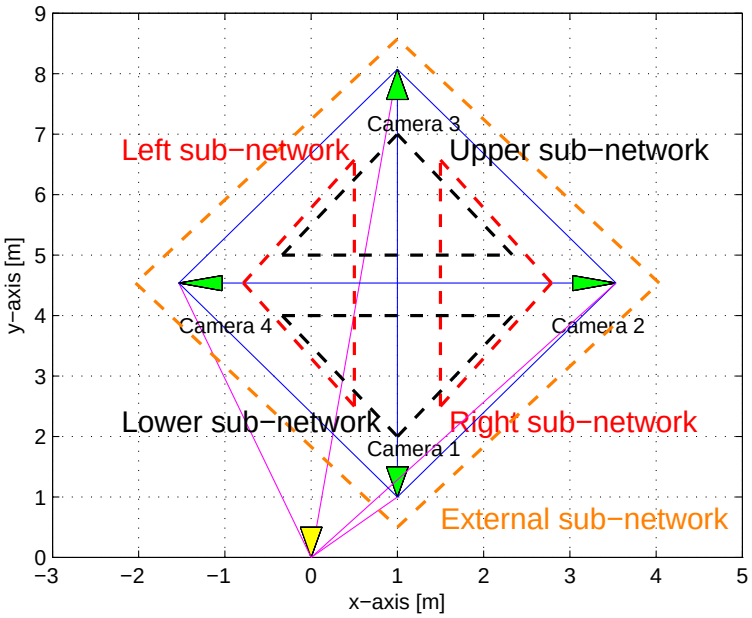
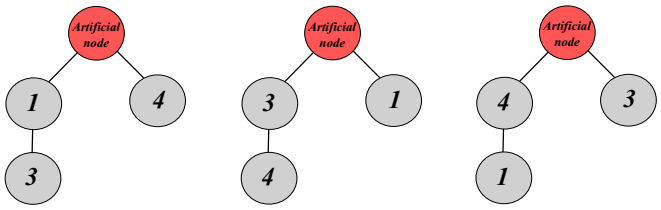
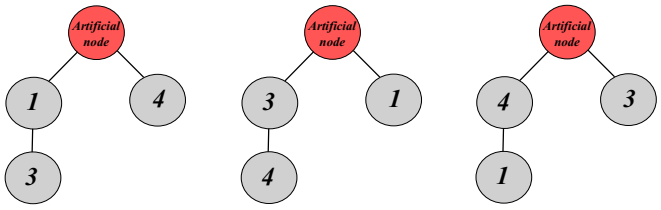


Figure 6.23.: Sub-networks identified in the camera network with additional links

The application of the algorithm described in the Chapter 4 on the subgraphs is performed using the MSTVC method in the initialization phase. The spanning trees constructed are different than those shown in Figure 6.11, except for the external sub-network. For the right and left sub-networks the configurations considered are shown in Figures 6.24 (a) and (b). As far as the upper and lower subnetworks are concerned, the initialization phase proceeds in the analogous manner so the spanning trees are not reported.



(a) Right sub-network



(b) Left sub-network

Figure 6.24.: Spanning trees - right and left sub-networks

Subsequently, the final estimate of the poses is determined through the application of the iterative procedure described by Tron and Vidal.

The value of the step-size is maintained equal to 0.01, whereas the number of iterations is reduced to 150 for both the part of the rotations and that of the translations. This choice is the consequence of some numerical problems tied to Matlab functionalities but it does not determinatively affect the value assumed by the cost function whose greater decrease occurs in the first tens of iterations.

Finally, the final poses of the cameras obtained from the implementation of the algorithm on the sub-networks are averaged in order to get a better and unique result for the entire network. Figure 6.25 shows the results achieved for each subgraph and computing the mean of the poses estimated in the different cases.

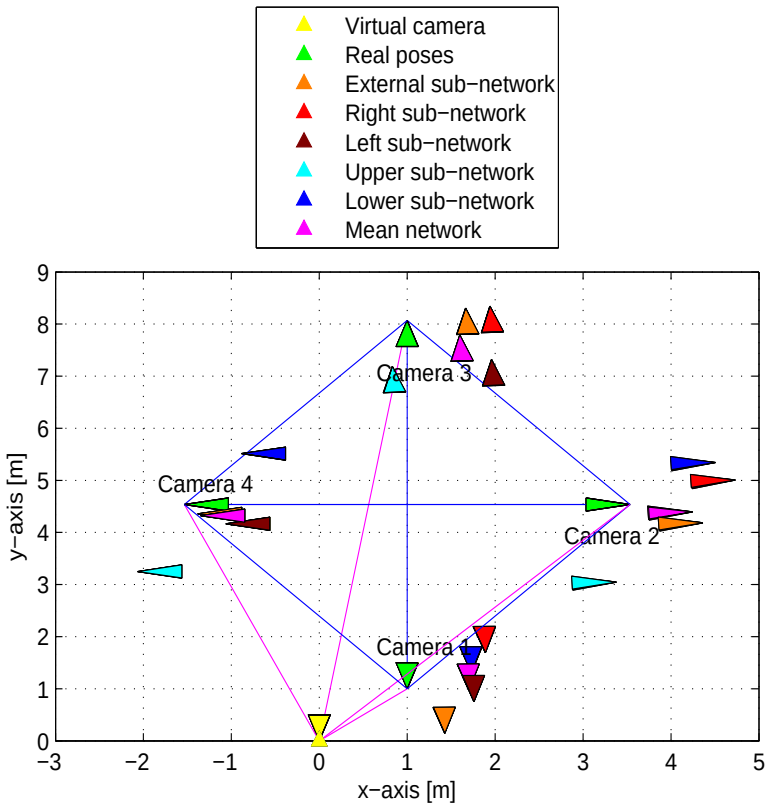


Figure 6.25.: Results obtained applying Tron-Vidal algorithm to different sub-networks

Focusing on the first and last row of Table 6.12 of the mean errors, it can be observed an improvement on the estimation of the translations. Similarly, the mean error on rotations slightly decreases. It is also worth underlining that the errors committed when the external sub-network is evaluated are practically the same of the case in which the additional communication links are not considered even if the number of iteration performed are different.

Figure 6.26 confirms what has been observed: it demonstrates that the estimates obtained by following the strategy illustrated are better than those obtained by simply considering the whole network with all the additional links.

	$\bar{e}_R$	$\bar{e}_T$
Complete network	**0.074**	**0.874**
External sub-network	0.069	0.701
Right sub-network	0.107	1.134
Left sub-network	0.049	0.873
Upper sub-network	0.075	1.262
Lower sub-network	0.064	1.075
Complete averaging network	**0.072**	**0.594**

Table 6.12.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm implementing or not the sub-networks strategy

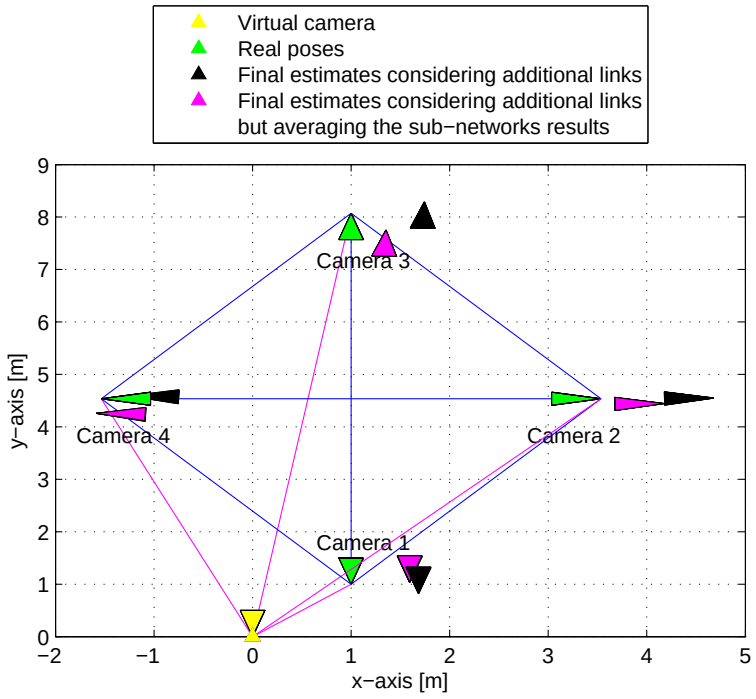


Figure 6.26.: Comparison between the results obtained applying Tron-Vidal algorithm on the network with additional links implementing or not the sub-networks strategy

In conclusion, it is interesting to analyze how the results vary considering the additional links and adopting the sub-networks strategy respect to the case where the communications protocol is the original one where each camera only exchanges information with its neighbors. Looking at Figure 6.27, it is not immediate to draw conclusions about the question, it is more convenient to consider the numerical values of the mean errors available in Table 6.13.

Thanks to the procedure described, a more complex communication pattern entails smaller mean error on translations. While, the error $\bar{e}_R$ is slightly higher than in the case in which the additional communication links are not considered.

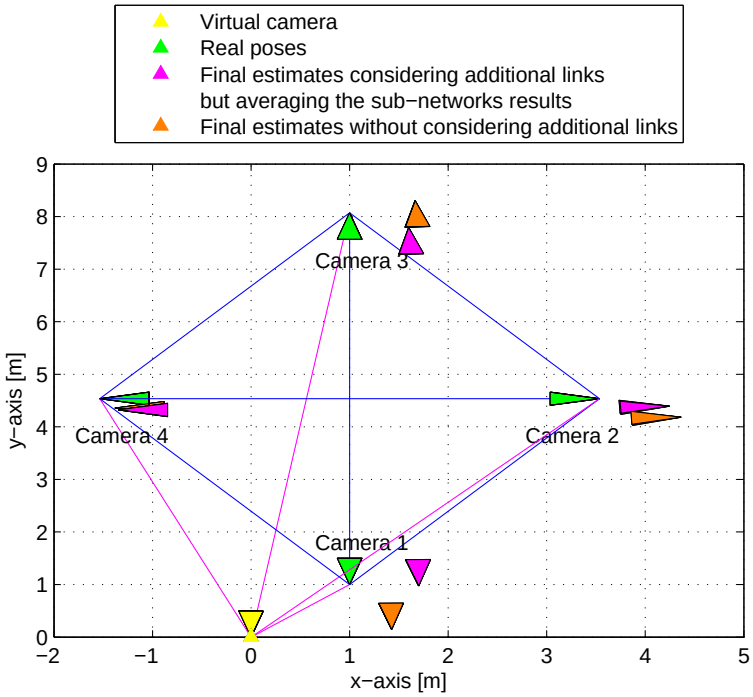


Figure 6.27.: Comparison between the results obtained applying Tron-Vidal algorithm on the network with and without additional links but implementing the sub-networks strategy

	$\bar{e}_R$	$\bar{e}_T$
Final estimates considering additional links but averaging the sub-networks results	0.072	0.594
Final estimates without considering additional links	0.069	0.702

Table 6.13.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm on the network with and without additional links but implementing the sub-networks strategy

Consequently, the conclusion that can be drawn is that a greater exchange of information does not correspond to a better estimate of the poses even exploiting a strategy based on subgraphs. This fact can be explained by highlighting that the links added are noisy anyway. This issue is in any case an open problem to be analyzed in a more formal way with respect to the topology graph of the communication links.

6.2.7 Algorithm implementation: step-size setting

Until now, the value of parameter ε has always been kept equal to 0.01. It is fair to question whether it has been a reasonable choice. For this purpose, in this paragraph the analysis focuses on how the trend of the cost function φ changes when the step-size varies.

In the following, let us consider the network in Figure 6.10 in which the virtual camera can communicate with all the other devices while every real camera exchanges information only with its neighbors.

The Tron-Vidal algorithm is always implemented considering 250 iterations for both the estimation of rotations and translations, whereas the final complete refinement is still neglected. Moreover, the initialization strategy chosen is the MSTVC method: the spanning trees built are those of Figure 6.11.

Figure 6.28 shows the trend of the cost function for three values of the parameter not too far between them, i.e. ε=0.05, 0.01, 0.005.

It is easy to observe that at the begin the descent of φ is less rapid when the value of the step-size is the smaller than the one considered in the previous simulations, i.e. ε=0.005. However, the minimum reached by the cost function at the end of the iterations is the same when ε= 0.01 and it halves its value.

Finally, the behavior in correspondence of ε=0.05 is very curious and suggests the existence of a critical value for the step-size included in the range $[0.01, 0.05]$ beyond which it is no longer possible to find a consistent solution causing the divergence of the cost function.

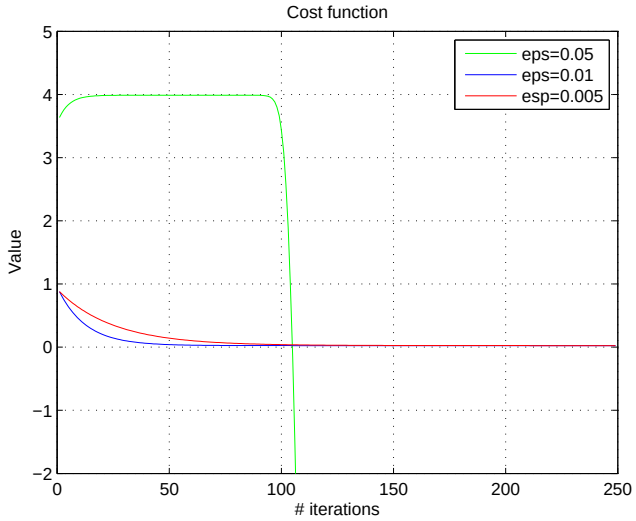


Figure 6.28.: Trend of cost function during the implementation of Tron-Vidal algorithm with three different values of the step-size

Observing Figure 6.29, it is possible to realize that the critical value is much close to 0.01. In fact, the plot shows the trends of the cost function for values of the step-size between 0.01 and 0.05: the case ε=0.01 is the only one in which φ converges.

To better understand the behavior of the algorithm for the values of the step-size higher than 0.01, let us consider a specific case: ε=0.02.

Analyzing the trend of the black line in Figure 6.29, it is worth noticing that the initial value of the cost function is 1.519. Then, the value of φ grows up until its maximum (2.019) reached at the 172-th iteration. Finally, after approximately 210 iterations, the cost function begins to decrease rapidly to the final value $\varphi = -9.489$.

It is important to point out that a very similar behavior is also observed for the trends relating to the other values of ε considered in Figure 6.29 although the diverging trend

begins after a number of iterations inversely proportional to the value of the step-size suggesting a relation between the two quantities, probably also influenced by numerical issues of Matlab.

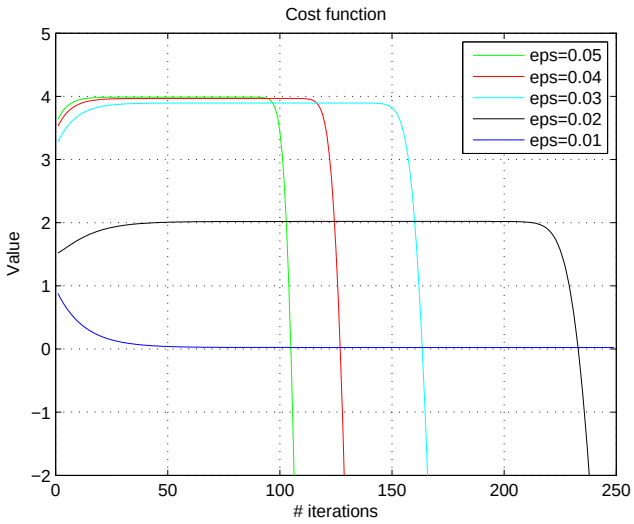


Figure 6.29.: Trend of cost function during the implementation of Tron-Vidal algorithm with $\varepsilon \in [0.01, 0.05]$

Figure 6.30 displays in detail the composition of φ (green line) as the sum of φ_R (red line) and φ_T (blue line). The divergence is related to the part of the cost function relative to rotations: while the blue line remains almost constant after the initial growth, the red line drops sharply after 210 iterations.

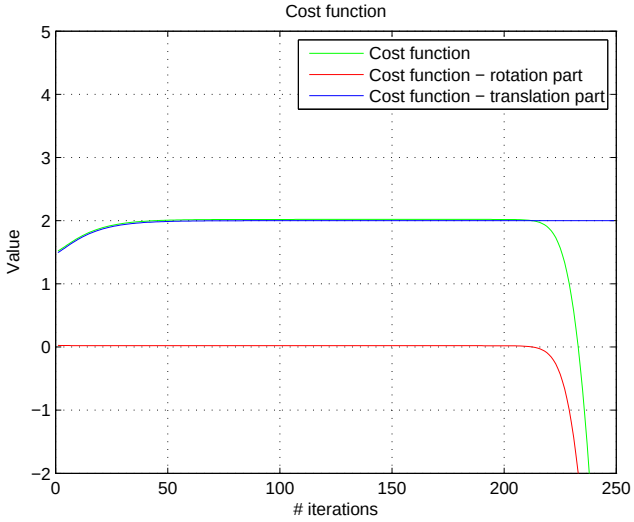


Figure 6.30.: Trend of cost function during the implementation of Tron-Vidal algorithm with ε=0.02

It is therefore interesting to observe the evolution of the mean error on rotations. Figure 6.31 reports the trend of $\bar{e}_R$ when ε=0.02 and ε=0.01.

When the step-size has the higher value, the error diverges consistently with the previous observations. Similarly, when ε= 0.01, the trend is convergent towards a value

close to zero according to the optimization strategy of the Tron-Vidal algorithm based on least-squares criterion.

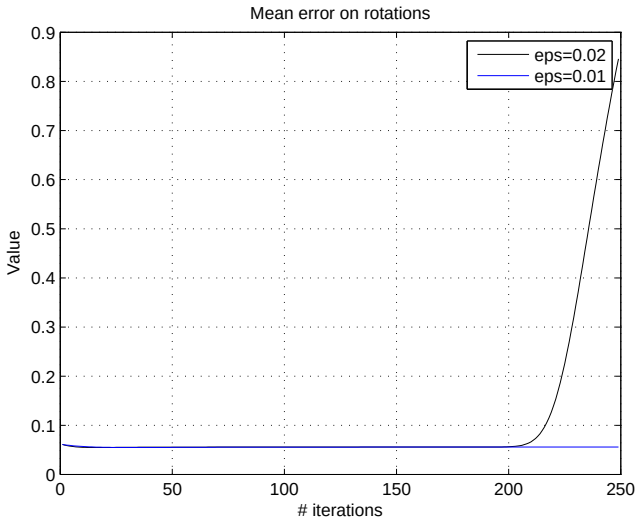


Figure 6.31.: Trend of mean error on rotations when ε=0.02 and ε=0.01

More explicitly, the rotation matrices estimated at the end of the iterative procedure with ε= 0.02 (left) and ε=0.01 (right) are

$$\hat{R}_1 = \begin{bmatrix} 0.509 & -0.071 & -0.072 \\ -0.038 & 0.471 & 0.087 \\ -0.053 & 0.077 & 0.548 \end{bmatrix}, \quad \hat{R}_1 = \begin{bmatrix} 0.999 & -0.033 & -0.015 \\ 0.03 & 0.999 & 0.011 \\ 0.014 & -0.011 & 0.999 \end{bmatrix},$$

$$\hat{R}_2 = \begin{bmatrix} -0.077 & -0.064 & -0.509 \\ 0.087 & 0.470 & 0.044 \\ 0.549 & 0.077 & 0.049 \end{bmatrix}, \quad \hat{R}_2 = \begin{bmatrix} -0.024 & -0.019 & -0.999 \\ 0.010 & 0.999 & -0.019 \\ 0.999 & -0.011 & -0.023 \end{bmatrix},$$

$$\hat{R}_3 = \begin{bmatrix} -0.513 & -0.053 & 0.057 \\ 0.058 & 0.469 & -0.085 \\ 0.072 & 0.076 & -0.547 \end{bmatrix}, \quad \hat{R}_3 = \begin{bmatrix} -0.999 & 0.003 & -0.014 \\ 0.004 & 1.000 & -0.009 \\ 0.014 & -0.009 & -0.999 \end{bmatrix},$$

$$\hat{R}_4 = \begin{bmatrix} 0.097 & -0.056 & 0.507 \\ -0.100 & 0.478 & -0.046 \\ -0.552 & 0.063 & -0.030 \end{bmatrix}, \quad \hat{R}_4 = \begin{bmatrix} 0.060 & -0.007 & 0.998 \\ -0.034 & 0.999 & 0.009 \\ -0.998 & -0.034 & 0.060 \end{bmatrix}.$$

When ε=0.02, the final estimated matrices are very different than the real ones: the diagonals have values that differ significantly from 1.

In conclusion, it can be stated that setting ε equal to 0.01 prevents strange behaviors in the algorithm implementation.

6.3 Experimental results on a real scenario

In this last section, the results obtained through the analysis on synthetic data are compared with those achieved by implementing the algorithm described in the Chapter

4 on a real network, specifically the considered experimental setup is the one described in [10]. Figure 6.32 shows the layout of the system: it is composed by four PTZ devices and two fixed cameras. Each agent shares its field of view with its neighbors, so the network topology is represented by a circulant graph. In the figure, it is also reported the set of images used for the initial calibration through Bouguet's toolbox of Matlab. For more details on the images acquisition and calibration phases it is recommended to consult the paper [10].

To proceed with the analysis, it is simply assumed that the noisy relative poses between the different cameras are available. Nevertheless, it is important to report that the authors of [10] underline that they have experimentally noted that the initial estimates of normalized translations are more error prone than the same estimates of rotations.

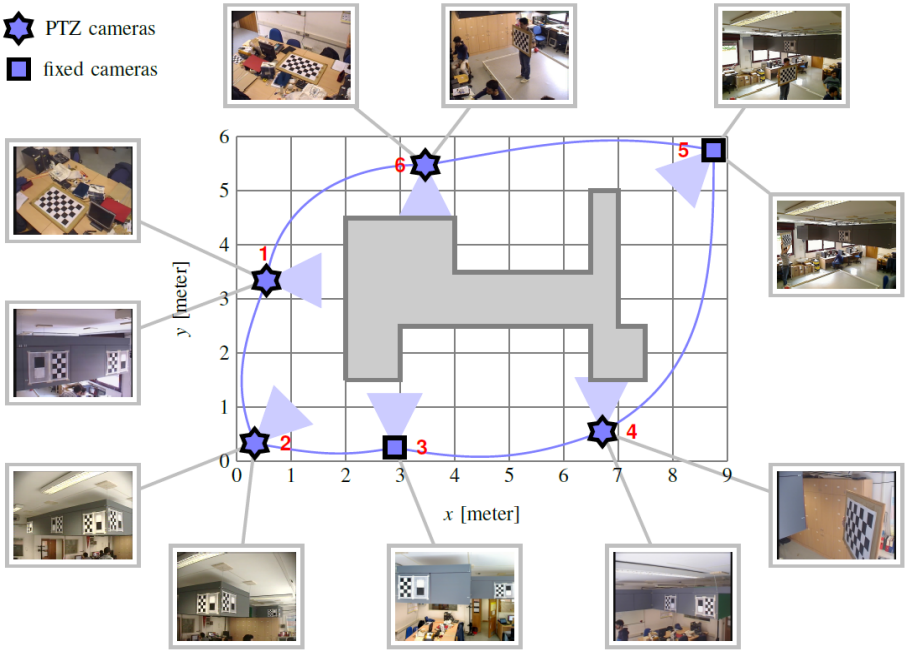


Figure 6.32.: Experimental camera network layout (top view)

In the following paragraphs, the Tron-Vidal algorithm is executed in standard conditions, in the case in which a relative pose (e.g. $\tilde{g}_{12}$ and/or $\tilde{g}_{21}$) in input is more noisy than the other ones and finally considering one of the additional links existing between the cameras but not shown in Figure 6.32.

In all the considered situations, the initialization is performed by applying the method of the single spanning tree. This choice is justified by the inability to place in the scene a virtual camera having the field of view overlapping with all the other devices in the network. In addition, the MST method is not used as the calculation of the poses becomes too expensive: since the network is formed by six cameras associated with a circulant graph, it would be necessary to determine more than one hundred poses.

As far as the another design parameters are concerned, it is worth emphasizing that the step-size ε is always kept equal to 0.001. This value represents a trade-off between the number of iterations performed by the algorithm and the final value reached by the cost function. Experimentally it is possible to observe that increasing the number of

iterations means having to reduce the step-size due to Matlab numerical problems: the descent of the cost function becomes slower but the high number of iterations allows to reach a sufficiently small value.

Concerning the scale factors, the strategy used is that of the previous simulations: the λ_{ij} are initially set to 1 for all $(i, j) \in \mathcal{E}$ and then all the results are scaled by a unique factor λ=3.5.

The analysis focuses on the goodness of the estimates of the camera poses, evaluated in terms of mean errors but also errors on the rotation and translation of each device. These performance metrics are just been defined, however it is necessary to highlight that in subsequent tests the reference system is always placed in order to coincide exactly with that of the camera chosen as root in the initialization spanning tree. For this reason, the error on the absolute pose of the reference node is always null.

6.3.1 Experimental results: convergence

To start, let us consider the results obtained applying the iterative procedure described in Chapter 4 on the experimental network in standard conditions. The attention is focused on the effects of the reference system choice.

Figure 6.33 displays the results of the implementation of the Tron-Vidal algorithm on the experimental network choosing the camera 1 as reference, i.e. building an initialization spanning tree in which the node 1 is the root and there are two unbalanced branches, the left is formed by the nodes 2, 3, 4 while the right includes the nodes 5, 6. The number of iterations performed by the procedure is imposed equal to 1000 for both rotations and translations estimate, whereas the final complete refinement is disregarded.

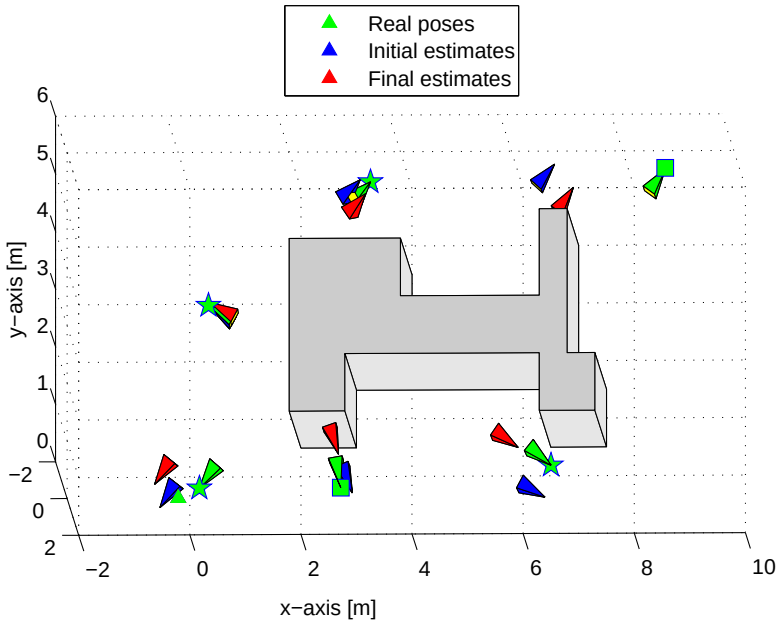


Figure 6.33.: Results obtained applying the Tron-Vidal algorithm to the experimental network choosing the node 1 as reference (1000 iterations)

From Table 6.14 of the errors and from Figure 6.34 that shows the trend of the translation part of the cost function, it is possible to guess that the number of iterations set for the estimate of the translations is not sufficiently high.

Although it is five times higher than the value chosen in the previous paragraphs, the mean error $\bar{e}_T$ does not decrease during the performance of the algorithm and the trend of φ_T is not yet sufficiently flat after 1000 iterations, i.e the value reached is not enough close to the minimum.

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	e_{R_5}	e_{R_6}
Initial poses	0.145	0	0.072	0.281	0.265	0.187	0.063
Final poses	0.123	0.023	0.051	0.210	0.130	0.149	0.114
	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}	e_{T_5}	e_{T_6}
Initial poses	0.632	0	0.783	0.226	0.548	2.022	0.215
Final poses	0.685	0.058	0.817	0.610	0.724	1.688	0.215

Table 6.14.: Errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network choosing the node 1 as reference (1000 iterations)

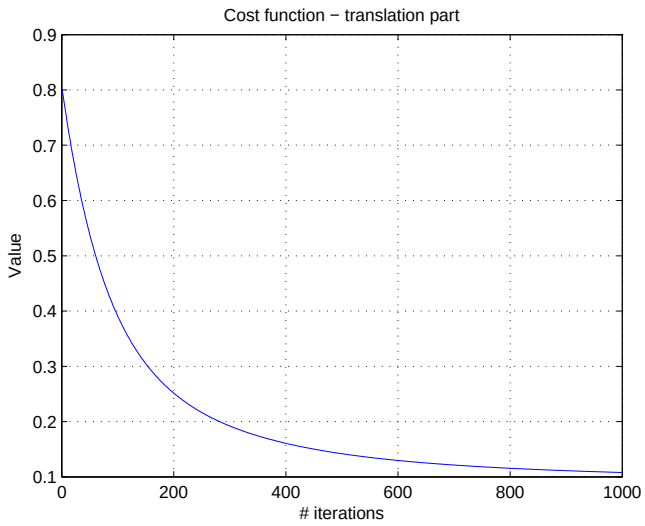


Figure 6.34.: Trend of translation part of cost function during the implementation of the Tron-Vidal algorithm on the experimental network choosing the node 1 as reference (1000 iterations)

For these reasons, Figure 6.35 shows the improvement that is achieved when the number of iterations relative to translations increased by an order of magnitude (10000). For the rotations, this change is not possible because of the numerical limits imposed by Matlab, besides it is not necessary in terms of trend of cost function.

Evaluating Tables 6.14 and 6.15, it is possible to note that, with respect to the initial poses, the nodes closer to the root (2 and 6) have generally smaller errors in agreement with the theory. Truthfully, e_{T_2} does not reflect this behavior. Similarly, the furthest nodes (4 and 5) should have greater errors but this is not apparent.

These observations are justified by the fact that the considered setup is real, therefore, the noise on the relative measures is not evenly distributed.

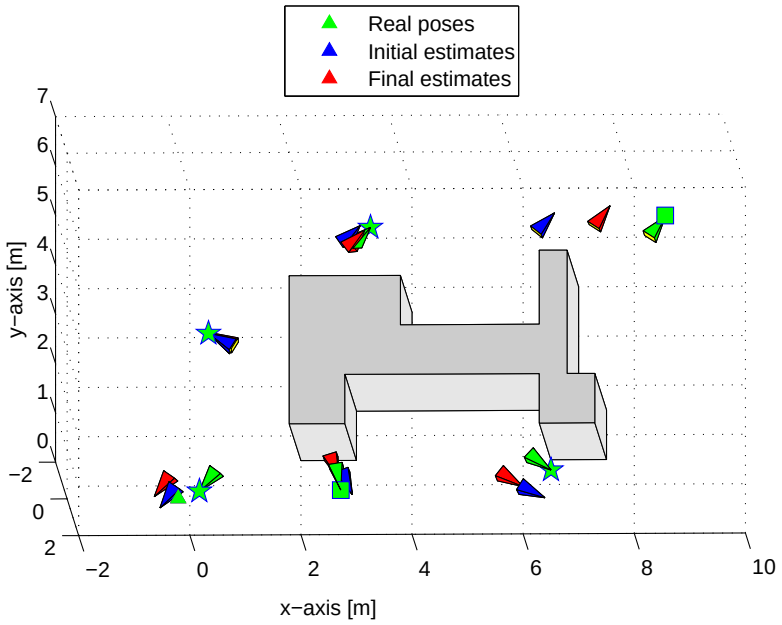


Figure 6.35.: Results obtained applying the Tron-Vidal algorithm to the experimental network choosing the node 1 as reference (1000 iterations for translation estimates)

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	e_{R_5}	e_{R_6}
Initial poses	0.145	0	0.072	0.281	0.265	0.187	0.063
Final poses	0.123	0.023	0.051	0.210	0.130	0.149	0.114
	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}	e_{T_5}	e_{T_6}
Initial poses	0.632	0	0.783	0.226	0.548	2.022	0.215
Final poses	0.481	0.037	0.822	0.256	0.610	1.043	0.117

Table 6.15.: Errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network choosing the node 1 as reference (10000 iterations for translations estimate)

Since the distribution of noise on initial relative poses is not uniform, it is easy to verify that the errors can be very different if the Tron-Vidal algorithm is applied to the real network in standard conditions but choosing a different camera as reference.

Figure 6.36 shows the estimated poses when the initialization spanning tree has the node 5 as root. The simulation is performed keeping unaltered the numbers of iterations compared to the previous case.

The error values available in Table 6.16 confirm the information that can be derived from the plot: the initial estimates are far from the real ones, however, at the end of the iterative procedure, the results obtained are quite satisfactory but not better than the case in which the observer position is in correspondence with the camera 1.

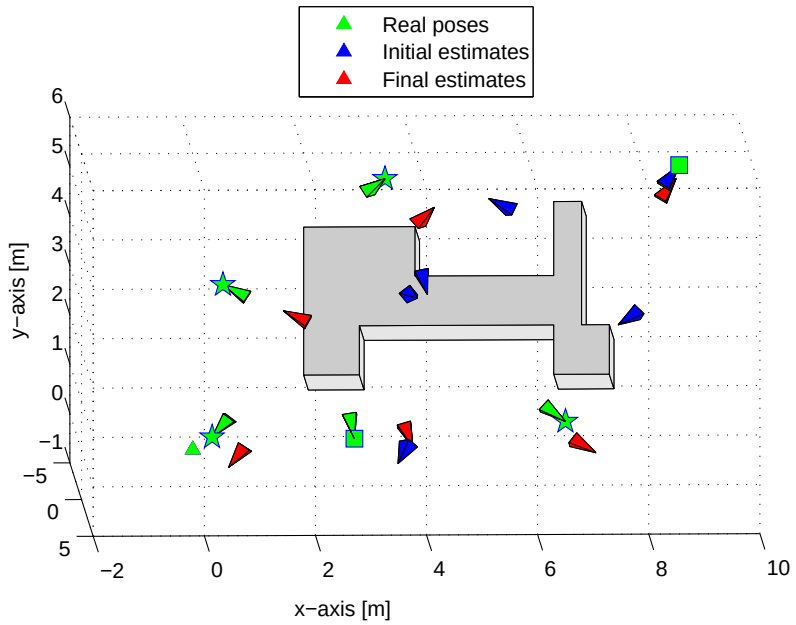


Figure 6.36.: Results obtained applying the Tron-Vidal algorithm to the experimental network choosing the node 5 as reference

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	e_{R_5}	e_{R_6}
Initial poses	1.447	2.495	1.232	0.238	2.167	0	2.548
Final poses	0.177	0.151	0.161	0.283	0.213	0.023	0.228
	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}	e_{T_5}	e_{T_6}
Initial poses	2.778	4.374	5.045	3.169	2.155	0	1.922
Final poses	0.994	1.306	1.029	1.250	1.029	0.265	1.086

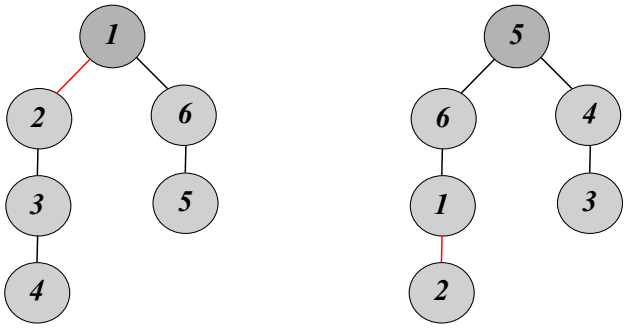
Table 6.16.: Errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network choosing the node 5 as reference

6.3.2 Experimental results: noise effect

In this paragraph the aim is to analyze the results of the application of the algorithm proposed by Tron and Vidal when one of the relative poses required as inputs has greater uncertainty than the others.

A Gaussian noise with variance of 2 degrees for rotations and 0.1 meters for the translations is added to the just noisy poses $\tilde{g}_{12}$ and $\tilde{g}_{21}$. Consequently, it is interesting to evaluate the estimates calculated when the reference is placed in correspondence with the cameras 1 and 5.

This choice is justified considering the spanning trees built in the initialization phase. The Figures 6.37 (a) and (b) show that the much noisy edge between 1 and 2 is located in distinct and significant positions when the roots are the nodes 1 e 5, respectively.



(a) Node 1 as root (b) Node 5 as root

Figure 6.37.: Spanning trees - different roots

Figure 6.38 exhibits the initial and final estimates when the reference system is that of the camera 1. The number of iterations is fixed to 1000 for the rotations estimate and to 10000 for the translations one.

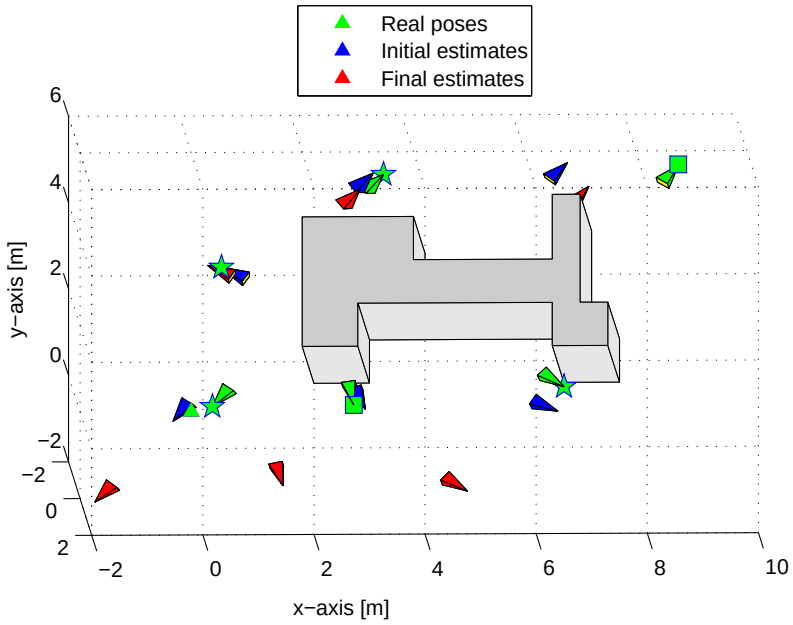


Figure 6.38.: Effect of unbalanced noise distribution on the results of Tron-Vidal algorithm applied to the experimental network choosing the node 1 as reference

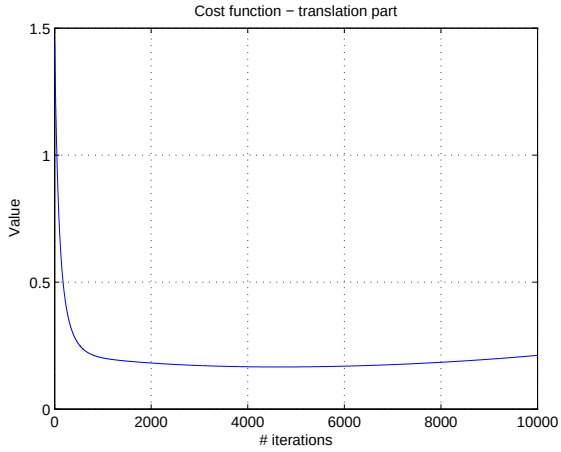
Examining the results obtained at the end of the iterative procedure (red triangles), it is easy to verify that the worse estimates are those of the devices in the lower part of the plot, i.e. cameras 2, 3 and 4. The errors on the translations of these devices are the highest, as can be seen in Table 6.17. Concerning the rotations, the behavior is less evident. The highest error is that on the camera 5 but this fact can be justified by observing that it is one of the fixed devices whose resolution is lower respect to the PTZ cameras. It is also important to focus on the fact that, while $\bar{e}_R$ decreases, the mean error on translations grows considerably by applying the Tron-Vidal algorithm.

Remembering that the translations are more prone to errors, it is convenient to analyze the behavior of the translation component in the cost function.

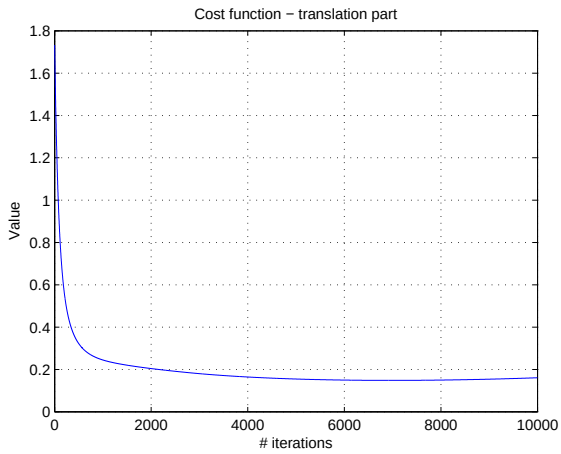
	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	e_{R_5}	e_{R_6}
Initial poses	0.145	0	0.072	0.281	0.265	0.187	0.063
Final poses	0.125	0.084	0.094	0.135	0.122	0.212	0.100
	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}	e_{T_5}	e_{T_6}
Initial poses	0.632	0	0.783	0.226	0.548	2.022	0.215
Final poses	2.177	0.357	3.741	2.842	3.439	1.984	0.697

Table 6.17.: Errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network in presence of unbalanced noise distribution choosing the node 1 as reference

Figure 6.39 (a) reveals that the behavior of φ_T is identical to that observed in the synthetic setup analysis. Adding uncertainty to the relative pose between two cameras, the function rapidly decreases but then, after reaching the minimum, slightly increases.



(a) Node 1 as reference



(b) Node 5 as reference

Figure 6.39.: Effect of unbalanced noise distribution on the trend of translation part of the cost function during the implementation of Tron-Vidal algorithm on the experimental network

A similar behavior, although less accentuated, is also encountered in the case in which the observer position coincides with that of the camera 5. The trend of φ_T in Figure 6.39(b) exhibits a small growth in the past iterations. This issue requires a more profound analysis in the future development of this work.

Table 6.18 of errors demonstrates that, in opposition to the previous case, both $\bar{e}_R$ and $\bar{e}_T$ decrease for the final poses; nevertheless their values are greater than in the case without additional noise.

	$\bar{e}_R$	e_{R_1}	e_{R_2}	e_{R_3}	e_{R_4}	e_{R_5}	e_{R_6}
Initial poses	1.447	2.495	1.231	0.238	2.167	0	2.548
Final poses	0.183	0.139	0.195	0.286	0.220	0.031	0.224
	$\bar{e}_T$	e_{T_1}	e_{T_2}	e_{T_3}	e_{T_4}	e_{T_5}	e_{T_6}
Initial poses	2.778	4.374	5.045	3.169	2.155	0	1.922
Final poses	1.419	1.536	2.029	1.654	1.680	0.341	1.278

Table 6.18.: Errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network in presence of unbalanced noise distribution choosing the node 5 as reference

Moreover, observing Figure 6.40, it occurs that the worst initial poses are those of the devices 1 and 2 above all concerning the translations. In particular, the error e_{T_2} remains much high even after the implementation of the Tron-Vidal algorithm. Instead, the final errors on the camera 1 are not so large probably because of its particular position in the network and with respect to the object observed.

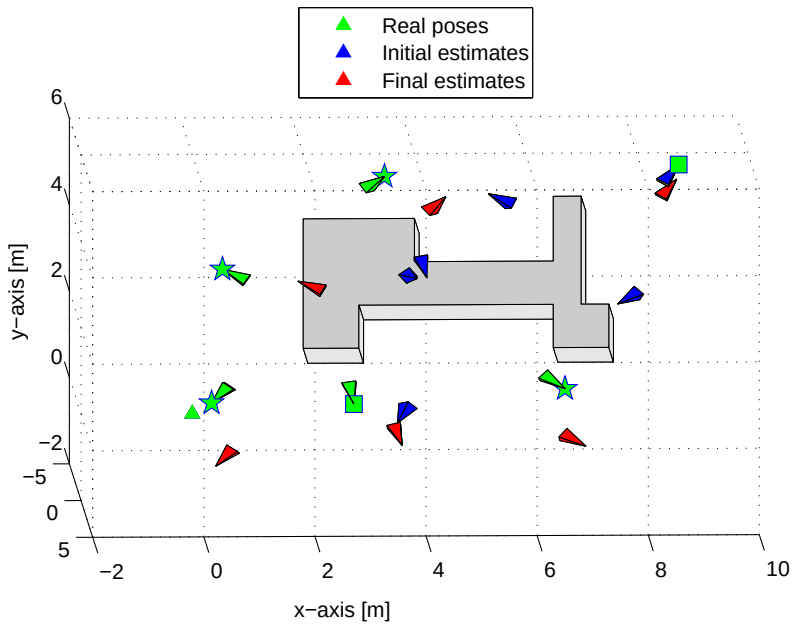


Figure 6.40.: Effect of unbalanced noise distribution on the results of Tron-Vidal algorithm applied to the experimental network choosing the node 5 as reference

In conclusion, it is possible to say that the tests on real and synthetic data bring similar considerations with respect to the performance of the translation part of the cost function, whereas the results about the poses and relating errors are different. This observation is justified by the fact that the uncertainty on real measurements is unevenly, differently from what happens in the previous simulations.

6.3.3 Experimental results: additional links

To conclude this chapter on the validation of the proposed algorithm, the analysis focuses on the effect of the addition of a communication link in the real camera network.

As already pointed out, the experimental layout of Figure 6.32 is such that every device is able to communicate only with its neighbors, by associating the network to a circulant graph. Truthfully, some existing links have been omitted because of the difficulties encountered in the calibration of the cameras involved.

In this section, a more complex topology is analyzed, considering the fact that the fields of view of the cameras 4 and 6 overlap since in the structure (gray polygon) there is an open path not shown in Figure 6.32.

Evaluating a synthetic setup, it has been observed that the presence of additional communication links results in a deterioration of the estimates of the absolute poses of the cameras in the network (Figure 6.21 and Table 6.11). Similarly, in the real case a more complex communication pattern does not allow to improve the results achieved at the end of the implementation of the Tron-Vidal algorithm.

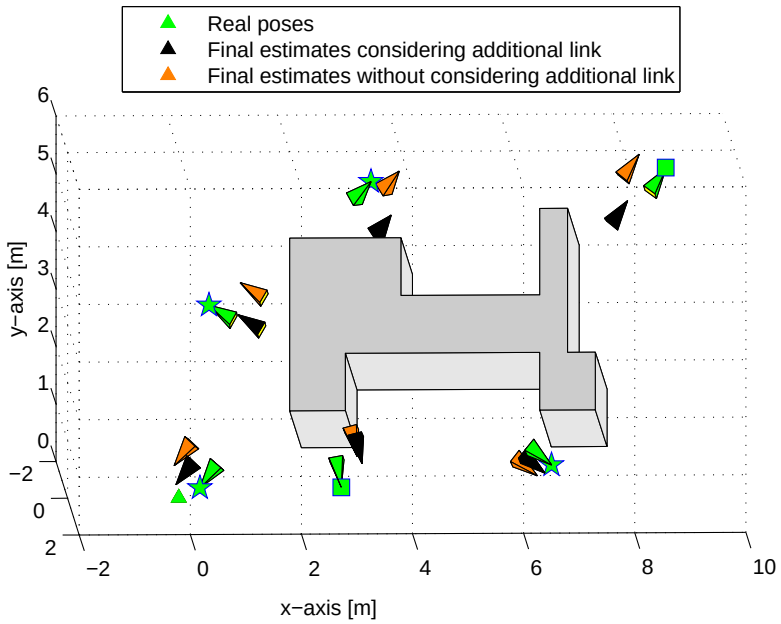


Figure 6.41.: Comparison between the results obtained applying Tron-Vidal algorithm on the experimental network with and without the additional link

Let us consider the Figure 6.41 obtained by implementing the iterative procedure described in Chapter 4 on the experimental network choosing the system of camera 4

as reference and imposing 1000 iterations for the estimation of rotations and 10000 for that of translations.

In the plot the absolute poses estimated without considering the link between the cameras 4 and 6 are represented by orange triangles which seem to be located close to the green one (real poses) in half of the cases. For devices 1, 2 and 4 the best reconstruction seems to be that obtained by considering the more complex communication pattern.

A clear picture of the situation can be gained by assessing errors in numerical terms. Table 6.19 proves that the mean errors on rotations and translations are greater when the new additional link is considered, even if the difference is very small respect to the values assumes by $\bar{e}_R$ and $\bar{e}_T$ when the simpler topology is evaluated. This behavior is analogous to that highlighted in synthetic data analysis.

	$\bar{e}_R$	$\bar{e}_T$
Final poses considering additional links	0.102	0.574
Final poses without considering additional links	0.098	0.559

Table 6.19.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network with and without the additional link

Likewise the synthetic case, it is interesting to observe the trends of the cost function by comparing the two network topologies. Figures 6.42 and 6.43 illustrate the behavior of the rotational component and the translational component of φ in correspondence to the presence (black line) or not (orange line) of the additional link.

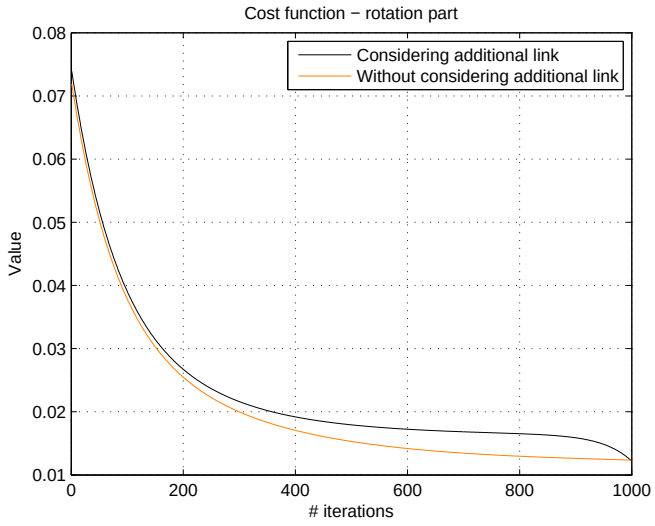


Figure 6.42.: Comparison of the trends of the rotational part of the cost function during the implementation of Tron-Vidal algorithm on the experimental network with and without the additional link

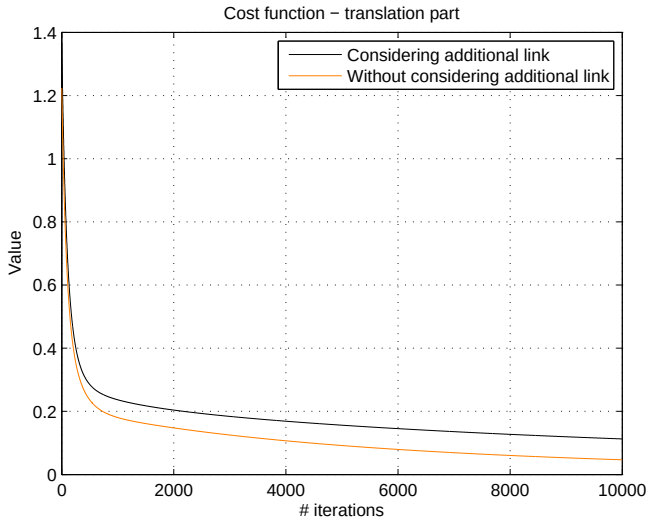


Figure 6.43.: Comparison of the trends of the translational part the cost function during the implementation of Tron-Vidal algorithm on the experimental network with and without the additional link

It is worth of notice that the trend represented by black line is worse for both φ_R and φ_T. When a more complex communication pattern is considered the descent of the cost function becomes less pronounced. Furthermore, the rotational component has an unexpected curvature in the last iterations.

These observations are consistent with those made in the case of the synthetic data analysis: the introduction of extra links in the network results in a slowdown in the achievement of the minimum of the cost function.

In order to improve the results achieved when the additional link is considered, a strategy based on the identification of sub-networks is follow, as in the synthetic case.

The first step is dividing the network into three subgraph:

- the *external sub-network* coincides with the original topology without considering the information exchange between the nodes 4 and 6;
- the *left sub-network* is composed by the nodes 1, 2 , 3, 4 and 6, ignoring the presence of the node 5 and its communication properties;
- the *right sub-network* is made only of the nodes 4, 5 and 6 in order to pinpoint a complete subgraph in which every device is capable of interacting with all the others.

The main idea is to apply the Tron-Vidal algorithm to the different sub-networks choosing always the node 4 as reference and then to average the results obtained with the purpose of determining a unique set of absolute poses for the all devices in the network.

Figure 6.44 displays the results achieved by applying this strategy. It is important to emphasize that the number of iterations has been considerably lowered due to numerical problems related to the management of the orthogonal matrices in Matlab: for the estimation of rotations it is been fixed to 500, while the iterations for the translations estimation are 5000.

Furthermore, it is worth to highlight that the spanning trees built in the initialization phase are balanced for left and right subnetworks. Maintaining the node 4 as the root, in the first case the branches are formed by the nodes 2-3 and 5-6, in the second case the left branch is constituted by the node 5 while the right branch coincides with the node 6.

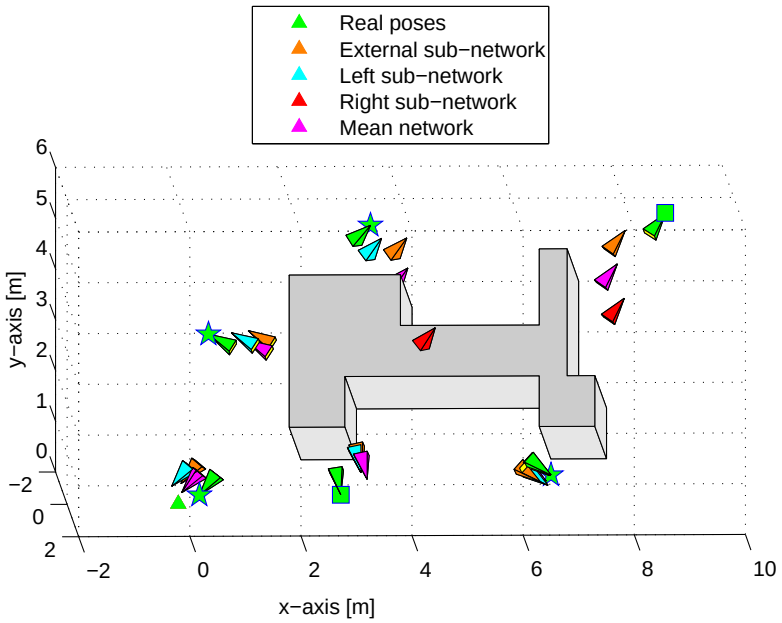


Figure 6.44.: Results obtained applying Tron-Vidal algorithm to different sub-networks

Observing the first and last rows of Table 6.20 relative to the mean errors committed in the implementation of the procedure based on the sub-networks, it is easy to verify that the $\bar{e}_R$ decreases contrary to $\bar{e}_T$. This fact does not reflect the assertions done in the synthetic data analysis, but now the attention is focused on an experimental network in which it was pointed out several times that the translations are more error prone.

	$\bar{e}_R$	$\bar{e}_T$
Complete network	**0.102**	**0.574**
External sub-network	0.098	0.598
Left sub-network	0.078	0.415
Right sub-network	0.113	1.415
Complete averaging network	**0.097**	**0.697**

Table 6.20.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network with the additional link implementing or not the sub-networks strategy

The deterioration in the estimation of the translations related to the application of the proposed strategy is evident in Figure 6.45 where it is not possible to appreciate an improvement with respect to rotations.

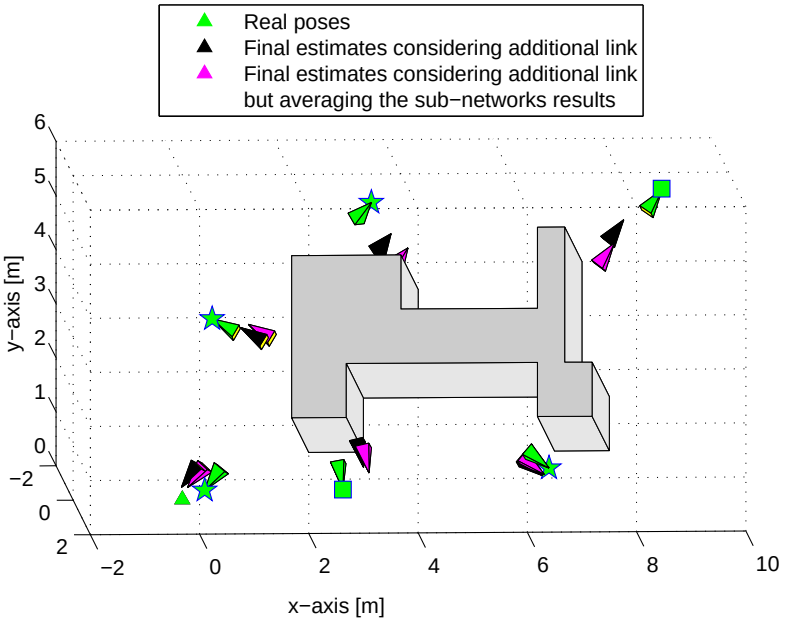


Figure 6.45.: Comparison between the results obtained applying Tron-Vidal algorithm to the experimental network with the additional link implementing or not the sub-networks strategy

Finally, Figure 6.46 and Table 6.21 show, as done in the synthetic data analysis, the comparison between the results obtained without considering the additional link and evaluating a more complex communication pattern but using the method of splitting the network into subgraphs.

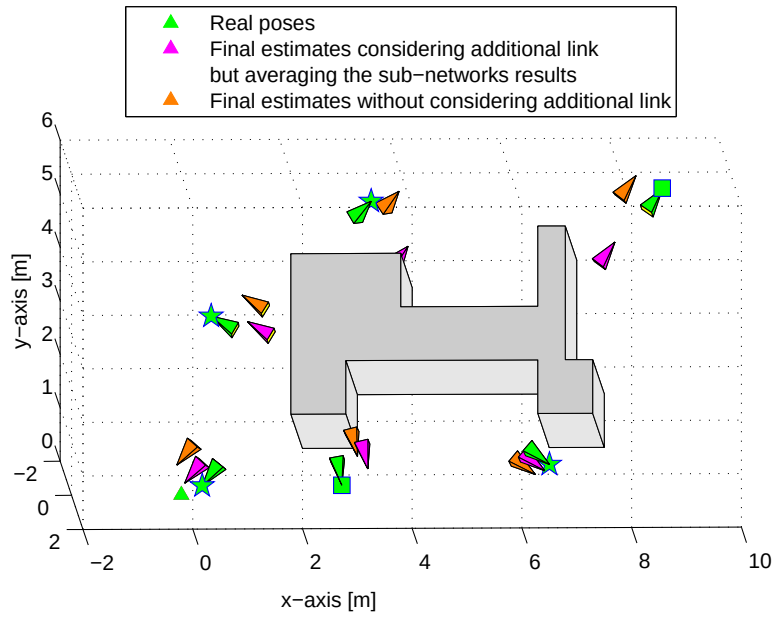


Figure 6.46.: Comparison between the results obtained applying Tron-Vidal algorithm to the experimental network with and without the additional link but implementing the sub-networks strategy

The criticality of the translations estimate remains obvious: concerning the easiest network topology the error $\bar{e}_T$ is always lower. The rotations estimation very slightly improves when the sub-networks strategy is implemented on a more complex communication layout.

	$\bar{e}_R$	$\bar{e}_T$
Final estimates considering additional links but averaging the sub-networks results	0.097	0.697
Final estimates without considering additional links	0.098	0.559

Table 6.21.: Mean errors on rotations and translations obtained applying Tron-Vidal algorithm to the experimental network with and without the additional link but implementing the sub-networks strategy

In conclusion, it can be said that what has been observed in the simulations carried out on synthetic setups does not exactly correspond to the results gained by the tests on real network. Nevertheless, it is fair to point out that the main issue is the estimate of translations, probably also because these are linked to the scale factors.

7

Conclusions

This final chapter summarizes the work done by focusing the attention on the main observations that can be drawn from the analysis of the obtained results. The profound study of the algorithm proposed by Tron and Vidal in [1] and described in Chapter 4 allows to achieve some interesting conclusions. At the same time, the criticalities about the procedure are several as well as the issues still open above all concerning the choice of the design parameters. To this end, an entire section is devoted to the ideas to develop in future works.

Contents

7.1 Summary of results

The purpose of this work is solving the localization problem for a camera system, i.e. the estimate of the rotation matrix and the translation vector of each device expressed in an absolute and fixed reference system. The problem is tackled exploiting the algorithm proposed by Tron and Vidal based on the minimization of a suitable cost functional in $SE(3)$ in a distributed manner. In particular, the analysis of this iterative procedure is the core of the thesis itself.

In the opening chapters, the operating context is described in detail by providing the mathematical concepts needed to understand the issue and by clearly formulating the problem to be solved. In Chapter 4, the algorithm object of analysis is meticulously illustrated, in particular new initialization strategies are proposed. The main innovation consists in the fact that the single spanning tree method is replaced by two approaches based on the construction of multiple spanning trees, whose results are averaged in order to minimize and uniformly distribute the errors on initial estimates.

Chapter 6 is entirely devoted to the validation of the proposed algorithm considering some measures of error as performance metrics. The conclusions that can be drawn are several and interesting.

Firstly, the MSTVC initialization method seems to be capable of providing the best final estimates: the initial distribution of the uncertainty allows to compensate for errors during the iterative procedure. Moreover, the presence of a virtual camera overcomes the question of choosing and giving priority to a node of the network as reference: the position of the scene observer is set a priori in reliable manner. This initialization strategy still has some disadvantages, the main one is undoubtedly the computational load due to the need of calculating the average of the rotation that is not an analytical operation.

Tests conducted on simulative and experimental scenarios support the study also in the case of additional communication links and unbalanced noise distribution. More specifically, the comparison between the real and synthetic setups suggests that the error which affects the initial relative pose is a determining factor on the estimate of the final poses. Similarly, the choice of the value to be imposed to the step-size and the number of iterations to be performed are crucial for the convergence of the procedure.

In conclusion, it can be stated that the major difficulty in the localization problem for a camera system is the impossibility of separating the estimation of rotations from that of the translations. In particular, this last one is more problematic because of the dependence on the scale factors in the cost functional.

7.2 Future works

Since the camera networks are emerging on a large scale in the context of control systems, there are numerous possible future works aimed at optimizing the solution of the localization problem that is the key step of almost all the applications.

On the top of the list of ideas to develop there is certainly the resolution of the issues about the method that estimates the poses in the domain of Fourier transform, in order

to validate the hypothesis that the combination of the algorithms proposed in [1] and [11] allows to obtain better results in terms of absolute poses. More specifically, it would be interesting to mix not only the results but also the strategies operating in the spatial and frequency domains, creating an hybrid procedure.

As far as the Tron-Vidal algorithm is concerned, a more deeper analysis should be carried out with regard to the design parameters by studying in detail the role of the step-size and its relations with the number of iterations to perform.

In addition, more attention could be devoted to the initialization phase. In relation to MSTVC strategy, one could consider alternative methods for calculating the average of a set of rotations based on metrics different from the Riemannian one [19], as well as different virtual camera positions. In the future, then, it is necessary to find a manner to initialize the scale factors λ_{ij} trying to estimate the real distances between the devices on the network and the observed scene. The literature is rather lacking on this issue but the work suggests the importance of these parameters.

The validation of the algorithm illustrated in Chapter 4 may continue in several ways:

- analyzing the behavior of the translation part of the cost function by deriving an analytical formulation in a 2D scenario, as done for φ_R;

- generating the initial relative poses in Matlab environment with other types of additive noise different from the Gaussian one or studying how the final results change according to the variation of the variances initially imposed on the relative rotations and translations;

- considering more complex topologies in both synthetic and real case, evaluating more articulated communication patterns or real layouts afflicted by occlusions and low resolution.

A

Pinhole camera model

The most common geometric model of a camera is the pinhole model. It describes the mathematical relationships that link the 3D coordinates of a point in the scene space to the 2D coordinates of its projection on the acquired image of an ideal pinhole camera, i.e. a camera whose aperture is described as a point and no lenses are used to focus the light.

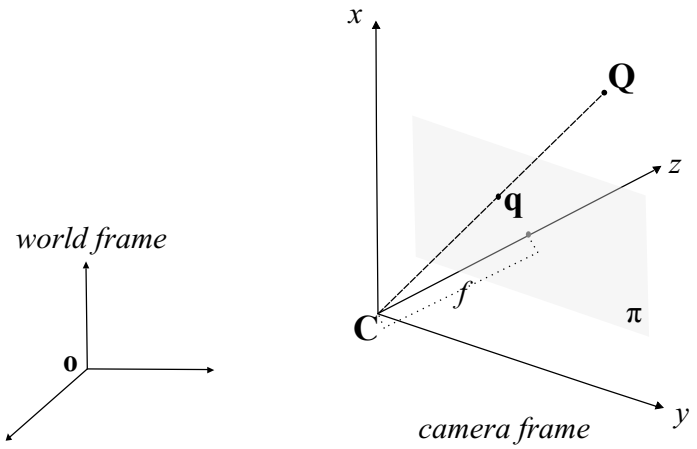


Figure A.1.: Pinhole camera model

The basic elements of the model (see Figure A.1) are:

- The 3D orthogonal coordinate system (*world frame*) whose origin is localized in the point **O** and whose axes are not named for the sake of simplicity in the notation.
- The 3D orthogonal coordinate system (*camera frame*) whose origin **C** is also the center or focus of projection of the camera. The three axes of this coordinate system are referred to as x, y, z. In particular, the z-axis is named *optical* or *principal axis*, as it is pointed in the viewing direction of the camera.
- The *image plane* π, i.e. the 3D plane parallel to axes x and y that contains the 3D points' projected images. It is located at distance f (*focal length*) from the origin **C** in the direction of the optical axis.
- The 3D point **Q** that is displaced in the scene space at coordinates (X_0, Y_0, Z_0) with respect to the world frame, whereas its coordinates relative to axes x, y, z are (X, Y, Z).
- The 2D point **q** that is the projection of the 3D point **Q** onto the image plane π.

- The 2D orthogonal coordinate system in the image plane, whose origin is placed at the intersection with the optical axes. The coordinates of the point **q** relative to this coordinate system are (u, v).

As far as the pinhole camera model is concerned, in the literature there are two main problems to be solved:

1. determining the relationship that links the 3D point **Q** coordinates in the world frame and the analogous ones in camera frame;

2. determining the relationship that links the coordinates relative to camera system of the 3D point **Q** and the coordinates of its projection onto the image plane, i.e. the 2D point **q**.

The solution of the first problem coincides with the determination of the expression of a rototranslation that links the coordinates (X_0, Y_0, Z_0) and (X, Y, Z), i.e. the calculation of the rotation matrix R and the translation vector T that appear in the following equality:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + T \quad \text{with} \quad (R, T) \in SE(3). \tag{A.1}$$

The second problem, instead, requires to solve the projection equations:

$$u = f\frac{X}{Z}, \tag{A.2}$$

$$v = f\frac{Y}{Z}. \tag{A.3}$$

It is worth of notice that, combining the previous equations, the transformation between (X_0, Y_0, Z_0) and (u, v) can be determined through the introduction of the homogeneous coordinates and the scale factor λ. Indeed, it is possible to write the following chain of relationships where the homogeneous coordinates are introduced both for the 2D point and the 3D point:

$$\begin{bmatrix} u \\ v \end{bmatrix} \sim \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{A.4}$$

The last matrix in (A.4) can be expressed as the product of two ones:

- *intrinsic parameters matrix (calibration matrix)*: $K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$;
- *standard projection matrix*: $\Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$.

Generally, the matrix K presents a more complex structure:

$$K = K_s K_f = \begin{bmatrix} S_x & S_\theta & O_x \\ 0 & S_y & O_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

where S_x and S_y (*horizontal* and *vertical scaling*) convert the real distances in terms of pixels, S_θ is the skew coefficient between the axes x and y, O_x and O_y (*horizontal* and *vertical offset*) represent the center point of the image plane.

As a consequence, the combination of the equations (A.1) and (A.4) allows to reach the following conclusion:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\Pi_0 \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K\Pi_0 \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} = K\Pi_0 g \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix}, \tag{A.6}$$

where g is referred as *extrinsic parameters matrix*.

Usually, the task of calculation the *camera matrix* $P = K\Pi_0 g$ is known as *calibration problem*: it is necessary to determine the six intrinsic parameters (S_x, S_y, S_θ, O_x, O_y and f) and the six extrinsic ones (three parameters for the rotation and three for the translation).

However, two other important matrices in pinhole model are the *essential matrix* and the *fundamental matrix*.

Given a pair of pinhole camera i and j, the essential matrix E is the 3×3 matrix that links the 2D points $\mathbf{q_i}$ and $\mathbf{q_j}$ seen by the i-th and j-th devices respectively. More explicitly

$$\mathbf{q_i}^T E \mathbf{q_j} = 0. \tag{A.7}$$

The matrix E can also be expressed in relation to the fundamental matrix F that describes the correspondence between the pair of cameras in more general and fundamental terms of projective geometry. Indeed, it results

$$E = K_i^T F K_j, \tag{A.8}$$

where K_i and K_j are the intrinsic calibration matrices of the two devices involved.

B

Rotation representations

In geometry various formalisms exist to express a rotation in three dimensions. The Euler's rotation theorem states that any displacement of a rigid body such that a point on it remains fixed (or a three-dimensional coordinate system with the fixed origin) is equivalent to a single rotation about some axis. As a consequence, the composition of two rotations is also a rotation, but above all a rotation may be uniquely described by a minimum of three real parameters. Nevertheless, many of the rotations representations known in literature involve more than the necessary minimum of three parameters, although the degrees of freedom are always at most three.

Rotation matrix In linear algebra, a rotation in three-dimensional Euclidean space is usually indicated using a rotation matrix, i.e. an element of the special orthogonal group $SO(3)$. This group includes all the 3×3 orthogonal matrices having unit determinant. Therefore, a rotation matrix $R \in \mathbb{R}^{3\times3}$ is such that $RR^T = I$ and $detR = +1$: these two properties assure that the orientation and the length are preserved in the transformation, according to the definition of rotation.

Axis-angle representation Another interesting notion is the so-called *axis-angle representation* or *equivalent axis representation*, where a 3D rotation is expressed as a single rotation ψ about an axis having the direction defined by the unit vector $\omega = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T \in \mathbb{R}^3$, $\|\omega\| = 1$. More explicitly $R = e^{\Omega\psi}$, where

$$\Omega = \Omega(\omega) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \in \mathfrak{so}(3) \tag{B.1}$$

is a skew-symmetric matrix associated to ω that belongs to $\mathfrak{so}(3) = \left\{S \in \mathbb{R}^{3\times3} : S^T = -S\right\}$. However, it is worth highlighting that this representation is not unique since choosing $\omega' = -\omega$ and $\psi' = 2\pi - \psi$ gives the same rotation as ω and ψ [11].

Angle representation Finally, according to Euler's rotation theorem, any rotation may be described by a sequence of rotations about some fixed axes using three angles (α, β, γ), called *Euler angles*. More explicitly,

- the angle α refers to a counterclockwise rotation about z-axis (yaw)

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}; \tag{B.2}$$

- the angle β indicates a counterclockwise rotation about the y-axis (pitch)

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}; \tag{B.3}$$

- the angle γ adverts to a counterclockwise rotation about the x-axis (roll)

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}. \tag{B.4}$$

Therefore, a rotation matrix can be expressed as the product of the previous three matrices in a certain order, e.g.

$$\begin{aligned} R(\alpha,\beta,\gamma) &= R_z(\alpha)R_y(\beta)R_x(\gamma) \\ &= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}. \end{aligned} \tag{B.5}$$

Bibliography

[1] R. Tron and R. Vidal, "Distributed image-based 3D localization of camera sensor networks," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, Dec. 2009, pp. 901–908.

[2] G. Bianchin, M. Luvisotto, and G. Michieletto, "Fault detection in sensor networks via cluster-based consensus," unpublished.

[3] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo, "A probabilistic approach to tof and stereo data fusion," in *3DPVT*, May 2010.

[4] J. Aspnes, T. Eren, D. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. O. Anderson, and P. Belhumeur, "A theory of network localization," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, Dec. 2006, pp. 1663–1678.

[5] W. E. Mantzel, H. Choi, and R. G. Baraniuk, "Distributed camera network localization," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, Nov. 2004, pp. 1381 – 1386.

[6] P. Rong and M. Sichitiu, "Angle of arrival localization for wireless sensor networks." in *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, 2006, pp. 374 – 382.

[7] G. Piovan, I. Shames, B. Fidanc, F. Bullo, and B. D. O. Anderson, "On frame and orientation localization for relative sensing networks," Dec. 2008, pp. 2326 – 2331.

[8] D. Devarajan and R. J. Radke, "Distributed metric calibration of large camera networks," in *in Proc. 1st Workshop on Broadband Advanced Sensor Networks*, 2004.

[9] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*, Apr. 2006, pp. 34 – 42.

[10] G. Baggio, M. Michielan, and S. Patron, "Distributed image-based localization of camera networks: a comparative analysis of different communication protocols," unpublished.

[11] L. Lucchese, G. Doretto, and G. M. Cortelazzo, "A frequency domain technique for range data registration," *Pattern Analysis and Machine Intelligence IEEE Transactions on*, vol. 24, no. 11, pp. 1468–1484, Nov. 2002.

[12] R. I. Hartley, "In defense of the eight-point algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 6, pp. 580–593, Jun. 1997.

[13] Q. Ji, M. S. Costa, R. M. Haralick, and L. G. Shapiro, "A robust linear least-squares estimation of camera exterior orientation using multiple geometric features," *ISPRS Journal of Photogrammetry & Remote Sensing*, vol. 55, no. 2, pp. 75–93, 2000.

[14] C. P. Lu, "Fast and globally convergent pose estimation from video images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 6, pp. 610–622, Jun. 2000.

[15] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment âĂŞ a modern synthesis," in *Vision Algorithms: Theory and Practice, LNCS*, 2000, pp. 298–375.

[16] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proceedings of the IEEE*, vol. 95, no. 1, Jan. 2007, pp. 215–233.

[17] R. Tron, B. Afsari, and R.Vidal, "Riemannian consensus for manifolds with bounded curvature," *Automatic Control, IEEE Transactions on*, vol. 58, no. 4, pp. 921–934, Apr. 2013.

[18] R. Tron and R. Vidal, "Distributed image-based 3D localization of camera sensor networks," Tech. Rep., Johns Hopkins University, 2009.

[19] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International Journal of Computer Vision*, vol. 103, no. 3, pp. 267–305, Jul. 2013.

[20] J. H. Manton, "A globally convergent numerical algorithm for computing the centre of mass on compact lie groups," in *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, vol. 3, Dec. 2004, pp. 2211 – 2216.

[21] M. Andreetto, L. Lucchese, and G. M. Cortelazzo, "Frequency domain registration of computer tomography data," in *Second International Symposium on 3D Data Processing Visualization and Transmission (3DPVT04)*, Sep. 2004, pp. 550–557.

[22] N. Benvenuto and M. Zorzi, *Principles of Communications Networks and Systems*. John Wiley & Sons Ltd, 2011.

[23] V. A. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & Systems (2Nd Ed.)*. Prentice-Hall, Inc.2, 1996.

[24] A. C. Benedetto, "Gestione immagini," unpublished.