16.485: Visual Navigation for Autonomous Vehicles (VNAV) **Fall 2021**

Lecture 23: Simultaneous Localization and Mapping I

Lecturer: Luca Carlone *Scribes: -*

This lecture focuses on Simultaneous Localization and Mapping (SLAM) and discusses:

- the typical structure of a SLAM system, including a SLAM front-end and a SLAM back-end,
- common formulations for the SLAM back-end, including landmark-based SLAM and pose graph optimization,
- and the sparsity structure of the resulting iterative solvers, which enables fast optimization.

The presentation is based on the survey [1], while here we mostly focus on different back-end formulations. The interested reader is referred to [1] for a more comprehensive overview of the state of the art and open problems. Another excellent reference on the topic is [3], which provides a broader review of the mathematical aspects.

23.1 SLAM Front-end and Back-end

Simultaneous Localization and Mapping (SLAM) is the problem of estimating the pose (or trajectory) of the robot and a representation of the environment using sensor data and without prior knowledge of the environment. In this sense, SLAM is an estimation problem where one wants to compute a variable of interest $\boldsymbol{x}$ (describing the state of the robot and a parametrization of the environment) using sensor data.

In previous lectures, we have introduced Maximum Likelihood estimation (MLE) and Maximum a Posteriori (MAP) estimation as two general estimation frameworks that allow framing an estimation problem in terms of mathematical optimization. The main ingredient of MLE and MAP estimation consists in the definition of measurement models that describe the likelihood of taking a measurement $\boldsymbol{y}_i$ from a state $\boldsymbol{x}$:

$$\boldsymbol{y}_i = h_i(\boldsymbol{x}, \boldsymbol{\epsilon}_i) \tag{23.1}$$

where $h_i(\cdot)$ is a known function (the "measurement model") and $\boldsymbol{\epsilon}_i$ is a random variable describing measurement noise.[1]

Now the main issue is that the data produced by key sensors used for robot navigation is difficult to directly describe as in (23.1). How can we describe the intensity of every pixel in an image with a function as in (23.1)? how can we describe every point in a dense point cloud produced by a lidar with a measurement model? Even if the design of a measurement model were viable, it would lead to extremely complex estimation problems. For instance, in order to describe the intensity of every pixel in the image as in (23.1), one would have to include as part of the state $\boldsymbol{x}$ the position and intensity of the light source, a parametrization of the geometry of the portion of the environment seen by the camera (e.g., a 3D mesh), the material properties that impact

[1] In MAP one also needs to define suitable priors on $\boldsymbol{x}$, which can also be written in a form akin to (23.1) and often have simple expressions, so we will focus on how to build measurement models in the rest of these notes.

the reflection of light, among other aspects. This would induce several issues: (i) it would lead to a very high-dimensional $\boldsymbol{x}$ (again, we would need to include all the variables of interest in $\boldsymbol{x}$), (ii) it would lead to a very large set of measurements (e.g., one measurement per pixel), which would prevent the optimization problem to be solved efficiently; furthermore, many of the quantities in $\boldsymbol{x}$ may not even be observable (e.g., can we uniquely identify material properties from images?), which implies that the resulting problem would have an infinite number of solutions.

To alleviate this problem, the common approach in computer vision and robotics (and in many other fields) is to extract "intermediate representations" that are easier to describe mathematically in the form (23.1). This creates a natural split in common SLAM architectures (Fig. 23.1): the sensor data is first passed to a set of algorithms (the "SLAM front-end") in charge of extracting such intermediate representations, and then the resulting intermediate representations are framed as in (23.1) and passed to a MLE/MAP estimator (the "SLAM back-end"). While the notion of "intermediate representations" is still vague, the next section shows that several computer vision techniques we have seen in VNAV can be already understood as a SLAM front-end. For instance, point features can be understood as intermediate representations of images that can be more easily described mathematically.

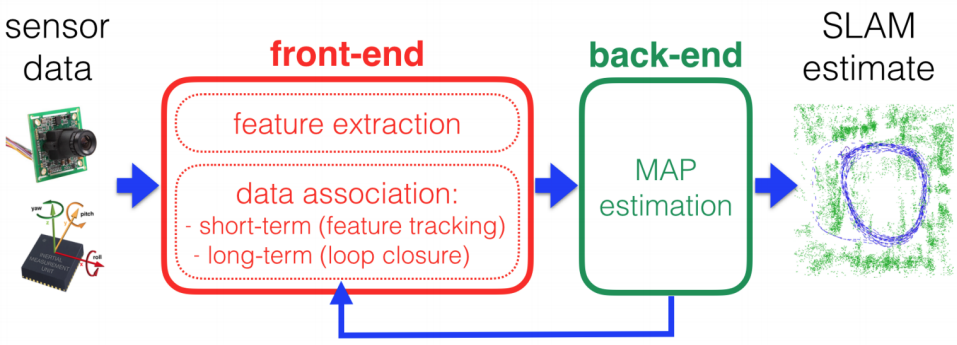

Figure 23.1: Front-end and back-end in a typical SLAM system.

23.2 Examples of SLAM Front-ends

We have already seen many algorithms that abstract the sensor data into intermediate representations that are easier to describe mathematically:

- 2-view geometry and visual odometry abstract a pair (or a subset) of images into motion estimates. In other words, they convert the raw sensor data (the images) into an intermediate representation, the relative pose $\bar{\boldsymbol{T}}_{t+1}^{t}$ of the camera between consecutive images.
- 3D registration abstracts a pair of point clouds into a relative pose between the corresponding observation poses. In other words, they convert the raw sensor data (two consecutive point clouds) into an intermediate representation, e.g., the relative pose $\bar{\boldsymbol{T}}_{t+1}^{t}$ of the lidar between consecutive scans.
- feature detection and tracking/matching: abstracts image pixels into a sparse set of pixels corresponding to the projection of 3D points $\boldsymbol{l}_k^{\mathrm{w}}$ onto the camera image.[2]
- place recognition identifies that two images, say i and j, are observing the same scene, from which (using 2-view geometry) we can compute the relative pose of the corresponding cameras $\bar{\boldsymbol{T}}_{j}^{i}$.

[2]In the symbol $\boldsymbol{l}_k^{\mathrm{w}}$, k is the index of the point, w denotes that the corresponding vector is expressed in the world frame.

These are only few examples of typical SLAM front-ends. In general, the choice of front-end depends on the sensor (e.g., we may need to use a different front-end for feature detection in a lidar scan, compared to a camera), and on the intermediate representation we desire as output of the front-end. For instance, while we talked about point feature detection, research efforts have been devoted to the development of front-end extracting other representations, including lines, planes, object poses, and other geometric primitives.

23.3 SLAM Back-end Formulations

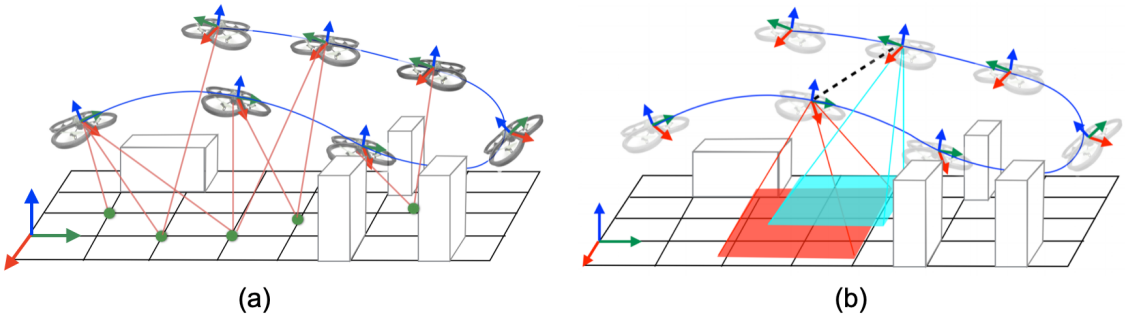

Figure 23.2: (a) Landmark-based SLAM. (b) Pose Graph Optimization (PGO).

So far, we have seen that the SLAM front-end produces an intermediate representation of the sensor data that can be expressed mathematically in a form akin to (23.1). Then the back-end computes an estimate give these intermediate representations using MLE or MAP estimation. In the following we review common choices of intermediate representations that lead to popular back-end formulations, summarized in Fig. 23.2.

23.3.1 Landmark-based SLAM

In landmark-based SLAM, the front-end produces odometry information (e.g., from visual odometry, visual-inertial odometry, or even wheel odometry for wheeled robots) and also produces detections of external *landmarks*. The most common case is the one in which landmarks are 3D points, but, as mentioned above, landmarks can be also more complex geometric primitives (e.g., lines, planes, objects). These intermediate representations can be easily described as in (23.1). The odometry $\bar{\boldsymbol{T}}^t_{t+1}$ describes the change in the pose between time t and $t+1$ and can be described using the following measurement model:

$$\bar{\boldsymbol{T}}^t_{t+1} = (\boldsymbol{T}^{\mathrm{w}}_t)^{-1} \, (\boldsymbol{T}^{\mathrm{w}}_{t+1}) \, \boldsymbol{T}_{\boldsymbol{\epsilon}_o} \tag{23.2}$$

where $\boldsymbol{T}_{\boldsymbol{\epsilon}_o}$ is a random variable describing the odometry measurement noise with covariance $\boldsymbol{\Sigma}_o$. Similarly, the measurements of a point landmark $\boldsymbol{l}^{\mathrm{w}}_k$ at time t can be written as:

$$\bar{\boldsymbol{y}}_{k,t} = h_i(\boldsymbol{T}^{\mathrm{w}}_t, \boldsymbol{l}^{\mathrm{w}}_k) + \boldsymbol{\epsilon}_l \tag{23.3}$$

where $\bar{\boldsymbol{y}}_{k,t}$ is the measurement and $\boldsymbol{\epsilon}_l$ is the measurement noise with covariance $\boldsymbol{\Sigma}_l$. For a monocular camera, $h_i(\cdot)$ would simply be the projection of the 3D point $\boldsymbol{l}^{\mathrm{w}}_k$ onto the camera at pose $\boldsymbol{T}^{\mathrm{w}}_t$. From now on, we will assume that all quantities (poses, points) are expressed in the world frame, and we will omit the superscript "w" to simplify the notation.

As we have seen when discussing MLE/MAP estimation, and assuming the noise is Gaussian (or wrapped Gaussian for the pose $\boldsymbol{T}_{\boldsymbol{\epsilon}_o}$), the measurement models (23.8)-(23.3) lead to the following estimator:

$$\min_{\substack{\boldsymbol{T}_t, t=1,\dots,n \\ \boldsymbol{l}_k, k=1,\dots,K}} \sum_{t=1,\dots,n-1} \|\log([\boldsymbol{T}_t^{-1}\,\boldsymbol{T}_{t+1}]^{-1}\bar{\boldsymbol{T}}_{t+1}^{t})^{\vee}\|_{\boldsymbol{\Sigma}_o}^2 + \sum_{k=1,\dots,K}\sum_{t\in\mathcal{S}_k} \|\bar{\boldsymbol{y}}_{k,t} - h_i(\boldsymbol{T}_t, \boldsymbol{l}_k)\|_{\boldsymbol{\Sigma}_l}^2 \tag{23.4}$$

where $\mathcal{S}_k$ is the subset of poses that observe landmark k, and for a vector $\boldsymbol{v}$, we have defined the weighted squared norm $\|\boldsymbol{v}\|_{\boldsymbol{\Sigma}}^2 = \boldsymbol{v}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{v}$. Using the shorthand notation $\boldsymbol{T}_1 \boxminus \boldsymbol{T}_2 \triangleq \log(\boldsymbol{T}_1^{-1}\boldsymbol{T}_2)^{\vee}$ (which computes a tangent-space representation of the relative pose between two poses $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$), the previous expression can be written more succinctly as:

$$\min_{\substack{\boldsymbol{T}_t, t=1,\dots,n \\ \boldsymbol{l}_k, k=1,\dots,K}} \sum_{t=1,\dots,n-1} \|(\boldsymbol{T}_t^{-1}\,\boldsymbol{T}_{t+1}) \boxminus \bar{\boldsymbol{T}}_{t+1}^{t}\|_{\boldsymbol{\Sigma}_o}^2 + \sum_{k=1,\dots,K}\sum_{t\in\mathcal{S}_k} \|\bar{\boldsymbol{y}}_{k,t} - h_i(\boldsymbol{T}_t, \boldsymbol{l}_k)\|_{\boldsymbol{\Sigma}_l}^2 \tag{23.5}$$

If the rotation noise is assumed to follow a Langevin distribution, the estimator becomes:

$$\min_{\substack{\boldsymbol{T}_t, t=1,\dots,n \\ \boldsymbol{l}_k, k=1,\dots,K}} \sum_{t=1,\dots,n-1} \|\boldsymbol{T}_t^{-1}\,\boldsymbol{T}_{t+1} - \bar{\boldsymbol{T}}_{t+1}^{t}\|_{\boldsymbol{\Sigma}_o}^2 + \sum_{k=1,\dots,K}\sum_{t\in\mathcal{S}_k} \|\bar{\boldsymbol{y}}_{k,t} - h_i(\boldsymbol{T}_t, \boldsymbol{l}_k)\|_{\boldsymbol{\Sigma}_l}^2 \tag{23.6}$$

where for a pose $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}$ and the covariance $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_R^2\mathbf{I}_3 & \\ & \boldsymbol{\Sigma}_t \end{bmatrix}$, we have defined the weighted squared matrix norm $\|\boldsymbol{M}\|_{\boldsymbol{\Sigma}}^2 = \boldsymbol{t}^{\mathsf{T}}\boldsymbol{\Sigma}_t^{-1}\boldsymbol{t} + \frac{1}{\sigma_R^2}\|\boldsymbol{R}\|_F^2$. Note that in the case of Langevin noise, the rotation noise is assumed *isotropic*, meaning that the covariance is a multiple of the identity matrix, $\sigma_R^2\mathbf{I}_3$.

So far we have kept complete generality with respect to the measurement model $h_i(\cdot)$. For instance, the estimators (23.5)-(23.6), with a suitable choice of $h_i(\cdot)$, can represent a visual-SLAM problem (where $h_i(\cdot)$ is a projection model) or a range-SLAM problem using radio beacons (where $h_i(\cdot)$ measures the range between the robot and radio beacons at locations $\boldsymbol{l}_k$ in the environment).

SLAM vs. Structure from Motion. When the measurements are produced by a monocular camera, we obtain the following visual-SLAM back-end:

$$\min_{\substack{\boldsymbol{T}_t, t=1,\dots,n \\ \boldsymbol{l}_k, k=1,\dots,K}} \sum_{t=1,\dots,n-1} \|\boldsymbol{T}_t^{-1}\,\boldsymbol{T}_{t+1} - \bar{\boldsymbol{T}}_{t+1}^{t}\|_{\boldsymbol{\Sigma}_o}^2 + \sum_{k=1,\dots,K}\sum_{t\in\mathcal{S}_k} \|\bar{\boldsymbol{y}}_{k,t} - \pi(\boldsymbol{T}_t, \boldsymbol{l}_k)\|_{\boldsymbol{\Sigma}_l}^2 \tag{23.7}$$

where π is the perspective projection function. Comparing this expression with the one we discussed in the visual odometry lecture, it becomes clear that the visual-SLAM problem is very related to *bundle adjustment* in Structure from Motion (SfM). The key differences are as follows:

- the SLAM problem typically has the odometry terms, since the data is collecting by a single robot moving in the environment (in SfM, the poses correspond to different cameras picturing the same scene, e.g., multiple tourists taking a picture of a monument);
- in SLAM, the data is presented incrementally: at each time stamp the robot collects new data and has to solve the problem again, while in SfM the problem is solved only one time on the entire batch of data;
- in SLAM, the problem has to be solved in close-to-real-time, while in SfM the reconstruction may be allowed to require a few hours.
- in SLAM, the images are collected by the robot over time, which makes feature tracking possible/easier. On the other hand, feature matching is the approach of choice when finding correspondences in SfM.

23.3.2 Pose Graph Optimization

Pose Graph Optimization (PGO), also called pose-based SLAM and pose-graph SLAM, assumes a different intermediate representation from the front-end. In particular, in PGO, the front-end produces odometry information (e.g., same as in the previous case), but then uses place recognition to compute relative pose measurements between the current pose and a previous observation pose; such a relative pose is called a "loop closure" and can be established when two non-consecutive poses observe the same place.

The odometry $\bar{\boldsymbol{T}}_{t+1}^{t}$ describes the change in the pose between time t and $t+1$ and can be described as in (23.8). The loop closure measurements can be also described as relative pose measurements between two poses $\boldsymbol{T}_i$ and $\boldsymbol{T}_j$

$$\bar{\boldsymbol{T}}_{j}^{i} = \boldsymbol{T}_i^{-1}\, \boldsymbol{T}_j\, \boldsymbol{T}_{\boldsymbol{\epsilon}_{lc}} \tag{23.8}$$

Assuming the noise follows a wrapped Gaussian distribution, the PGO estimator becomes:

$$\min_{\boldsymbol{T}_t, t=1,\ldots,n} \sum_{t=1,\ldots,n-1} \|(\boldsymbol{T}_t^{-1}\, \boldsymbol{T}_{t+1}) \boxminus \bar{\boldsymbol{T}}_{t+1}^{t}\|_{\boldsymbol{\Sigma}_o}^2 + \sum_{(i,j)\in\mathcal{E}_{lc}} \|(\boldsymbol{T}_i^{-1}\, \boldsymbol{T}_j) \boxminus \bar{\boldsymbol{T}}_{j}^{i}\|_{\boldsymbol{\Sigma}_{lc}}^2 \tag{23.9}$$

where $\mathcal{E}_{lc}$ is the set of pose pairs (i,j) for which we have a loop closure between pose i and j.

Since the two terms in (23.9) have the same expression, we can write them together as:

$$\min_{\boldsymbol{T}_t, t=1,\ldots,n} \sum_{(i,j)\in\mathcal{E}} \|(\boldsymbol{T}_i^{-1}\, \boldsymbol{T}_j) \boxminus \bar{\boldsymbol{T}}_{j}^{i}\|_{\boldsymbol{\Sigma}_{ij}}^2 \tag{23.10}$$

where $\mathcal{E} = (1,2) \cup (2,3) \cup \ldots \cup (n-1,n) \cup \mathcal{E}_{lc}$ and collects pairs of poses for which we have odometry measurements or loop closures. This is the cost implemented when using `betweenFactor<Pose3>` in GTSAM.

Assuming that the rotation noise follows a Langevin distribution, it is possible to reformulate the PGO problem using the chordal distance [4]:

$$\min_{\boldsymbol{T}_t \in \mathrm{SE}(3), t=1,\ldots,n} \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{T}_i^{-1}\, \boldsymbol{T}_j - \bar{\boldsymbol{T}}_{j}^{i}\|_{\boldsymbol{\Sigma}_{ij}}^2 \tag{23.11}$$

which is often more convenient due to the quadratic nature of the cost function. This is the cost implemented when using `betweenChordalFactor` in GTSAM.

While we presented PGO as the result of a robot using an odometry-and-place-recognition front-end, the PGO model also applies to other SLAM problems.

Multi-robot SLAM as PGO. Assume that we have two robots α and β performing collaborative SLAM. The robots exchange images to establish loop closure between the corresponding images. Mathematically, the problem can be still formulated as a variant of PGO (in this case, using the chordal distance):

$$\min_{\boldsymbol{T}_{\alpha,t}, \boldsymbol{T}_{\beta,t}, t=1,\ldots,n} \sum_{(i,j)\in\mathcal{E}_\alpha} \|\boldsymbol{T}_{\alpha,i}^{-1}\, \boldsymbol{T}_{\alpha,j} - \bar{\boldsymbol{T}}_{\alpha,j}^{\alpha,i}\|_{\boldsymbol{\Sigma}_{ij}}^2 + \sum_{(i,j)\in\mathcal{E}_\beta} \|\boldsymbol{T}_{\beta,i}^{-1}\, \boldsymbol{T}_{\beta,j} - \bar{\boldsymbol{T}}_{\beta,j}^{\beta,i}\|_{\boldsymbol{\Sigma}_{ij}}^2 + \sum_{(i,j)\in\mathcal{E}_{\alpha\beta}} \|\boldsymbol{T}_{\alpha,i}^{-1}\, \boldsymbol{T}_{\beta,j} - \bar{\boldsymbol{T}}_{\beta,j}^{\alpha,i}\|_{\boldsymbol{\Sigma}_{ij}}^2 \tag{23.12}$$

where $\mathcal{E}_{\alpha\beta}$ is the set of loop closures (sometimes referred to as inter-robot loop closures [2]) between the robots. Note that the structure of problem (23.12) is essentially identical to (23.11) but the problem simultaneously estimates the poses of both robots. The formulation can be easily extended to an arbitrary number of robots.

Object-based SLAM as PGO. We can also use PGO as a back-end for the case where the SLAM front-end produces odometry measurements as well as measures the relative pose of external objects. In that case, PGO can be used to simultaneously estimate the pose of the robot and the poses of the external objects.

23.3.3 Some Remarks

Robustness and Outliers. One important observation is that to derive our MLE/MAP estimator we have assumed Gaussian noise (of Langevin noise, which can be understood as the equivalent of a Gaussian distribution on the group of rotations). However, the front-end can sometime make mistakes (e.g., feature tracking/matching might fail to detect the correct pixels in the images, or place recognition may match images of different places). These incorrect measurements produced by the front-end, often referred to as *outliers*, have large errors and violate the assumption of Gaussian noise, inducing large errors in the estimation. This observation stresses the importance of two aspects:

- it is important to vet the measurements produced by the front-end before passing them to the back-end. This motivates the use of RANSAC (or similar techniques) in the front-end that have the goal of removing outliers;
- it is important to consider more general back-ends having cost functions that are robust to the presence of outliers. The cost functions lead to the theory of M-estimation (where "M" stands for "Maximum Likelihood type" estimator), which we are going to discuss later in VNAV. See [5] for a discussion.

Priors. In this section we have only considered optimization problems where each term in the objective corresponds to a measurement (these are Maximum Likelihood estimators). In practice, it is customary to also add priors (e.g., a `priorFactor` in GTSAM), leading to MAP estimation. Such priors are used to constrain the problem and force it to have a unique solution. For instance, since relative poses are preserved when rigidly rotating a trajectory, all the optimization problems above have infinite solutions corresponding to arbitrary rigid transformations of the robot trajectory (and possibly of the landmarks). To avoid this ambiguity it is common to add a prior that fixes one of the poses (typically the first pose) to an arbitrary value (typically the identity pose), thus "anchoring" the trajectory. Note that this ambiguity does not arise when we have sensors measuring absolute quantities (e.g., a GPS).

23.4 Leveraging Sparsity to Speed-Up Computation

In this section we derive the expression of the Gauss-Newton (GN) update for a Pose Graph Optimization (PGO) problem. Then, we comment on the fact that the Normal equations arising from GN are sparse and have a precise sparsity pattern (and the same holds for landmark-based SLAM). This enables the use of fast sparse linear solvers that allow solving problems with thousands of variables in seconds.

23.4.1 Pose Graph Optimization: Normal Equations and Sparsity

Let us start from the PGO formulation we introduced in eq. (23.10) (where we make the $\boxminus$ explicit):

$$\min_{\boldsymbol{T}_t, t=1,\dots,n} \sum_{(i,j)\in\mathcal{E}} \|\log([\boldsymbol{T}_i^{-1}\,\boldsymbol{T}_j]^{-1}\bar{\boldsymbol{T}}_j^i)^\vee\|^2_{\boldsymbol{\Sigma}_{ij}} \tag{23.13}$$

Let us now derive the expression of a Gauss-Newton (GN) update for eq. (23.13), assuming that we are given an initial guess $\hat{\boldsymbol{T}}_t, t = 1,\dots,n$ for the poses in the problem.

As we mentioned in the optimization lectures, we first reformulate the problem into one defined in the tangent space at $\hat{\boldsymbol{T}}_t, t = 1,\dots,n$ by introducing a retraction $\mathcal{R}(\cdot)$ that transforms a vector in the tangent

space to a 3D pose:

$$\min_{\boldsymbol{\delta}_t, t=1,\dots,n} \sum_{(i,j)\in\mathcal{E}} \|\log([(\hat{\boldsymbol{T}}_i \cdot \mathcal{R}(\boldsymbol{\delta}_i))^{-1} \, (\hat{\boldsymbol{T}}_j \cdot \mathcal{R}(\boldsymbol{\delta}_j))]^{-1} \bar{\boldsymbol{T}}_j^i)^\vee\|^2_{\boldsymbol{\Sigma}_{ij}} \tag{23.14}$$

Now we note that for a vector $\boldsymbol{x}$ and a covariance $\boldsymbol{P}$, $\|\boldsymbol{x}\|^2_{\boldsymbol{P}} = \boldsymbol{x}^\mathsf{T}\boldsymbol{P}^{-1}\boldsymbol{x} = (\boldsymbol{P}^{-\frac{1}{2}}\boldsymbol{x})^\mathsf{T}(\boldsymbol{P}^{-\frac{1}{2}}\boldsymbol{x}) = \|(\boldsymbol{P}^{-\frac{1}{2}}\boldsymbol{x})\|^2$, hence we can rewrite (23.15) as:

$$\min_{\boldsymbol{\delta}_t, t=1,\dots,n} \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{\Sigma}_{ij}^{-\frac{1}{2}} \log([(\hat{\boldsymbol{T}}_i \cdot \mathcal{R}(\boldsymbol{\delta}_i))^{-1} \, (\hat{\boldsymbol{T}}_j \cdot \mathcal{R}(\boldsymbol{\delta}_j))]^{-1} \bar{\boldsymbol{T}}_j^i)^\vee\|^2 = \min_{\boldsymbol{\delta}_t, t=1,\dots,n} \|r_{ij}(\boldsymbol{\delta}_i, \boldsymbol{\delta}_j)\|^2 \tag{23.15}$$

where on the right-hand-side we simply noted that the term inside the squared norm is a function of $\boldsymbol{\delta}_i$ and $\boldsymbol{\delta}_j$; we denoted this function as $r_{ij} : \mathbb{R}^6 \times \mathbb{R}^6 \mapsto \mathbb{R}^6$, which is often called the *(whitened) residual error*.

Now we take a Taylor expansion of the terms inside the norm in (23.15):

$$\min_{\boldsymbol{\delta}\in\mathbb{R}^{6n}} \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{J}_{ij}\boldsymbol{\delta} - \boldsymbol{b}_{ij}\|^2 \tag{23.16}$$

where we defined:

- the vector $\boldsymbol{\delta}$, which includes all the unknown $\boldsymbol{\delta}_t \in \mathbb{R}^6, t = 1, \dots, n$. Mathematically:

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_1 \\ \boldsymbol{\delta}_2 \\ \vdots \\ \boldsymbol{\delta}_n \end{bmatrix} \in \mathbb{R}^{6n} \tag{23.17}$$

- the matrix $\boldsymbol{J}_{ij}$, which is the *Jacobian* of r_{ij} with respect to $\boldsymbol{\delta}$:

$$\boldsymbol{J}_{ij} = \begin{bmatrix} (\frac{\partial r_{ij}}{\partial \boldsymbol{\delta}_1})^\mathsf{T} & (\frac{\partial r_{ij}}{\partial \boldsymbol{\delta}_2})^\mathsf{T} & \dots & (\frac{\partial r_{ij}}{\partial \boldsymbol{\delta}_n})^\mathsf{T} \end{bmatrix} \in \mathbb{R}^{6\times 6n} \tag{23.18}$$

  Since r_{ij} only depends on $\boldsymbol{\delta}_i$ and $\boldsymbol{\delta}_j$, the Jacobian $\boldsymbol{J}_{ij}$ is sparse:

$$\boldsymbol{J}_{ij} = \begin{bmatrix} \boldsymbol{0} & \dots & \boldsymbol{0} & (\frac{\partial r_{ij}}{\partial \boldsymbol{\delta}_i})^\mathsf{T} & \boldsymbol{0} & \dots & \boldsymbol{0} & (\frac{\partial r_{ij}}{\partial \boldsymbol{\delta}_j})^\mathsf{T} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{6\times 6n} \tag{23.19}$$

- the vector $\boldsymbol{b}_{ij} = -r_{ij}(\boldsymbol{0}, \boldsymbol{0})$ (note that $r_{ij}(\boldsymbol{0}, \boldsymbol{0})$ is the residual error evaluated at the linearization point in our Taylor expansion).

Recalling that the squared norm of a vector is the sum of the squared norms of its sub-vectors, we succinctly rewrite (23.16) as:

$$\min_{\boldsymbol{\delta}\in\mathbb{R}^{6n}} \|\boldsymbol{J}\boldsymbol{\delta} - \boldsymbol{b}\|^2 \tag{23.20}$$

where $\boldsymbol{J}$ is a tall matrix stacking the Jacobians $\boldsymbol{J}_{ij}$ by rows, and $\boldsymbol{b}$ is a vector stacking all the $\boldsymbol{b}_{ij}$. As we have seen in previous lecture, the linear least squares problem (23.20) can be solved in closed form as:

$$\boldsymbol{\delta}^\star = (\boldsymbol{J}^\mathsf{T}\boldsymbol{J})^{-1}\boldsymbol{J}^\mathsf{T}\boldsymbol{b} \tag{23.21}$$

where the matrix $\boldsymbol{J}^\mathsf{T}\boldsymbol{J}$ is the Hessian of the problem. In practice, the inverse is never computed, but one rather uses linear solvers to solve the linear system

$$(\boldsymbol{J}^\mathsf{T}\boldsymbol{J})\boldsymbol{\delta}^\star = \boldsymbol{J}^\mathsf{T}\boldsymbol{b} \tag{23.22}$$

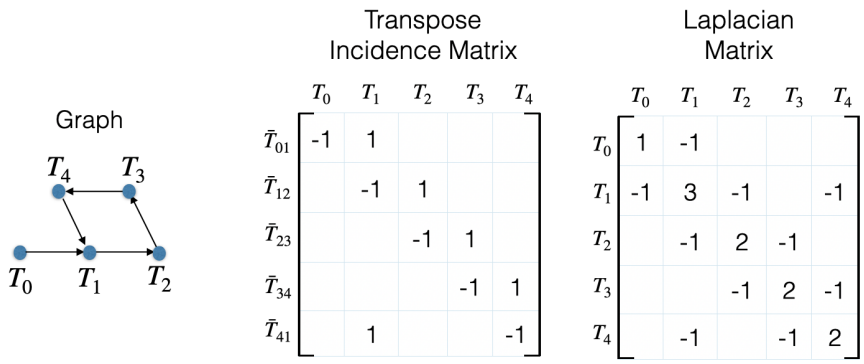

Figure 23.3: Transpose Incidence matrix and Laplacian matrix of a toy graph. The transpose incidence matrix is defined as follows: each row of the transpose incidence matrix describes an edge from node i to node j and has 2 non-zero elements, a -1 in position i and a $+1$ in position j. The Laplacian matrix is defined as follows: the i-th diagonal term of the matrix contains the node degree (number of edges incident on i) and an off-diagonal entry in position (i, j) is equal to -1 if there is an edge between node i to node j, or is zero otherwise. Calling $\boldsymbol{L}$ the Laplacian matrix and $\boldsymbol{A}$ the transpose incidence matrix, it holds $\boldsymbol{L} = \boldsymbol{A}^\mathsf{T}\boldsymbol{A}$.

which is often referred to as the *Normal equations*. A key observation, which directly follows from the structure of the Jacobians in (23.19), is that the matrix $\boldsymbol{J}$ (and hence the Hessian $\boldsymbol{J}^\mathsf{T}\boldsymbol{J}$) is very sparse (i.e., it has many entries equal to zero), which allows the use of very fast and scalable *sparse linear solvers*.

Sparsity of Jacobian and Hessian. Let us define a graph $\mathcal{G}$, where the nodes in $\mathcal{G}$ are the (indices of) the unknown poses in our pose graph, and the edges describe measurements between a pair of poses. For instance an edge (i, j) denotes that we have a measurement of the pose between pose i and pose j (either via odometry or loop closure). Now, since the Jacobian $\boldsymbol{J}$ is stacking the Jacobian $\boldsymbol{J}_{ij}$ of each measurement by rows and the only nonzero blocks in $\boldsymbol{J}_{ij}$ correspond to the variables involved in measurement (i, j), it is easy to see that the blocks of the Jacobian matrix $\boldsymbol{J}$ have the same sparsity pattern of the *transpose incidence matrix* of the graph $\mathcal{G}$ (Fig. 23.3). Similarly, the blocks of the Hessian matrix can be seen to have the same sparsity pattern of the *Laplacian matrix* of $\mathcal{G}$ (Fig. 23.3).

23.4.2 Landmark-based SLAM: Sparsity

For landmark-based SLAM problems, the derivation of the Normal equations in the Gauss-Newton update is similar to the one in the previous section (but includes landmarks which already live in Euclidean space), and again leads to sparse linear system. Fig. 23.4 and Fig. 23.5 show the sparsity patterns of the corresponding Jacobian and Hessian matrices in two toy problems. Fig. 23.4 considers a bundle adjustment problem with 3 cameras and 2 landmarks. Fig. 23.5 considers a similar visual SLAM problem where one is also given odometry measurements between consecutive camera poses.

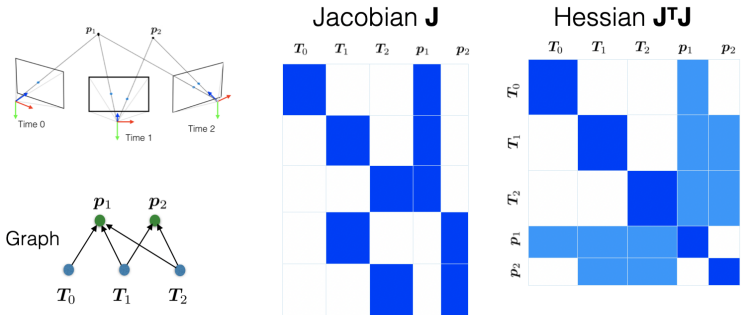

Figure 23.4: Bundle adjustment problem with 3 cameras and 2 landmarks. Sparsity patterns of the Jacobian and Hessian matrices.

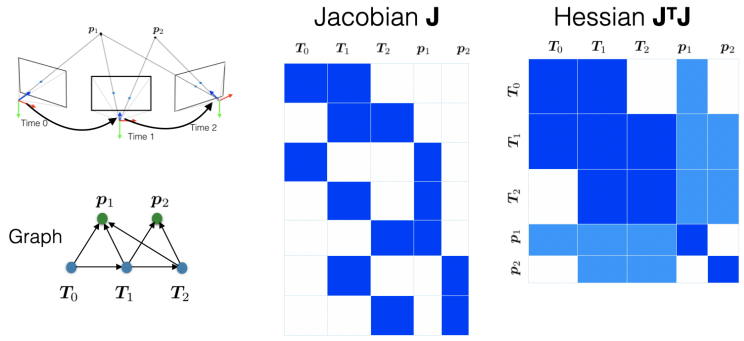

Figure 23.5: Visual SLAM problem with 3 cameras and 2 landmarks (with odometry). Sparsity patterns of the Jacobian and Hessian matrices.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robotics*, 32(6):1309–1332, 2016. arxiv preprint: 1606.05830, (pdf).

[2] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H.I. Christensen, and F. Dellaert. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models, accepted. *Intl. J. of Robotics Research*, 2017. arxiv preprint: 1702.03435, (pdf) (web) (code) (code) (code) (video) (video) (video) (video).

[3] F. Dellaert and M. Kaess. Factor graphs for robot perception. *Foundations and Trends in Robotics*, 6(1-2):1–139, 2017.

[4] D.M. Rosen, L. Carlone, A.S. Bandeira, and J.J. Leonard. SE-Sync: A certifiably correct algorithm for synchronization over the Special Euclidean group. In *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, San Francisco, CA, December 2016. extended arxiv preprint: 1611.00128, (pdf) (pdf) (code).

[5] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1127–1134, 2020. arXiv preprint arXiv:1909.08605 (with supplemental material), (pdf), .