7

Visual SLAM

Jakob Engel, Juan D. Tardós, Javier Civera, Margarita Chli,
Stefan Leutenegger, Frank Dellaert, and Daniel Cremers

Reconstructing the world and the sensor motion from cameras is a challenge referred to as Visual Simultaneous Localization and Mapping, shortened to Visual SLAM or VSLAM. With cameras being omni-present, cheap and power-efficient, the potential of visual SLAM for autonomous robots, self-driving cars, or mixed and augmented reality is endless.

7.1 Historic Background and Terminology

7.1.1 From Photogrammetry to Bundle Adjustment and Visual SLAM

Visual SLAM's history builds upon general SLAM methods, but it is also rooted in advances from the photogrammetry and computer vision communities. We highlight here the historical connection with these two last fields, the connection with SLAM being already addressed throughout the handbook.

In 1822, Nicéphore Niépce invented modern photography with the oldest surviving photograph being "A View from a Window at Le Gras" in 1827. The history of reconstructing the world from cameras started a few decades after the invention of photography. The French army officer Aimé Laussedat is often considered the inventor of photogrammetry as he pioneered the use of terrestrial photographs for topographic mapping around 1851. In 1867 the German engineer Albrecht Meydenbauer further developed photogrammetry for architectural surveys. From 1890 onward the German mathematician and mountaineer Sebastian Finsterwalder (president of the German Mathematical Society from 1915 onward) pioneered the use of aerial imagery for photogrammetric reconstruction of glaciers in the Alps and also advocated techniques of projective geometry [365]. His doctoral student Otto von Gruber formalized the mathematical framework of bundle adjustment for reconstruction of structure and motion from a set of corresponding points observed in multiple images. These concepts were developed around the beginning of the 20th century, well before the advent of computers. The deployment on computers was later pioneered by Hellmut H. Schmid, a German rocket scientist who developed matrix computation techniques for bundle adjustment and teamed up with Ameri-

can Duane C. Brown in the 1950s to deploy these methods on the largest computers of their time.

Reconstruction methods and the field of structure from motion research build on various camera models starting from the pinhole camera model and the use of projective geometry to capture the relationship between 2D point observations and their corresponding 3D world coordinates. In the early 1990s Tomasi and Kanade [1106] developed matrix factorization techniques for the reconstruction of static scenes under the simplified assumption of an orthographic projection. Whereas earlier approaches often focused on the reconstruction from two views, in the 2000s the community shifted to the problem of multiview structure from motion. Traditionally, the reconstruction pipeline involved feature extraction, correspondence estimation, the use of minimal solutions for obtaining an initial camera configuration and a subsequent bundle adjustment to obtain a globally consistent reconstruction. Much effort was therefore dedicated to the development of feature extraction and matching algorithms with feature descriptors such as SIFT [706] or SURF [69], and more recently a multitude of learning-based descriptors. In order to cope with incorrect point correspondences researchers developed sampling strategies such as RANSAC [336] that allowed the method to revisit the correspondence estimation in alternation with model fitting.

Whereas structure from motion is often focused on the accurate (generally offline) reconstruction of large-scale 3D worlds from an unordered collection of images, visual SLAM typically focuses on the online and realtime reconstruction from a moving camera. A prerequisite for such online and realtime approaches was therefore the development of causal methods such as [217] which focus on the challenge of optimal structure from motion given only the past images (as opposed to the entire image collection or video). The first real-time capable methods for structure from motion / visual SLAM emerged around 2000 [531, 253].

7.1.2 Terminology

The following terms are often used interchangeably to describe similar processes; however, place different emphasis depending on application and community they are used in.

Photogrammetry is the science of extracting accurate measurements, spatial information, and 3D reconstructions from 2D photographs. By analyzing overlapping images taken from different viewpoints, photogrammetry enables the creation of geometric representations of objects or environments. This technique is widely used in mapping, surveying, and 3D modeling, forming the foundation for many visual odometry and SLAM algorithms.

Bundle adjustment (BA) is a mathematical optimization method used to refine 3D reconstructions. It adjusts the positions, orientations, and optionally intrinsic parameters of cameras, along with the 3D positions of observed points, to minimize

the *reprojection error*—the difference between observed image points and the points projected from the 3D model. Optimization is typically performed using second-order methods such as Gauss-Newton or Levenberg-Marquardt and requires careful initialization and robust outlier rejection to converge effectively. Recent research has also explored *initialization-free bundle adjustment*, which aims to simplify the optimization process.

Structure from Motion (SfM) refers to the process of reconstructing 3D structures from a collection of 2D images taken from different perspectives. Unlike photogrammetry, which often assumes known camera positions, SfM simultaneously estimates both the 3D structure of the scene and the motion of the cameras. SfM is typically applied in non-causal, non-real-time scenarios, where images may come from diverse sources (e.g., internet photo collections) rather than a continuous video stream. Typically in the final stage SfM will revert to bundle adjustment for optimization. Landmark projects, such as *Building Rome in a Day* [21], demonstrate its scalability and potential for large-scale applications.

Visual Odometry (VO) focuses on estimating the motion of a camera by analyzing sequential visual frames in a video. It identifies visual correspondences between consecutive images to measure the relative motion of the sensor over time. VO primarily deals with local motion estimation and operates within a sliding window of recent observations, without building a global map. However, VO is often combined with a mapping component to form a complete visual SLAM system.

Visual SLAM is a computational technique that enables a system to simultaneously localize itself within an unknown environment and build a map of that environment in real time. It combines elements of photogrammetry, visual odometry, and structure from motion to process image data, track camera motion, and construct detailed 3D maps. In contrast to VO, it will typically employ an explicit functionality of recognizing previously visited places, re-localize relative to them, and optionally adjust pose estimates around such a "loop" – a process referred to as *loop closure*. Visual SLAM is a cornerstone technology for robotics, autonomous vehicles, and augmented reality, where precise navigation and environmental understanding are essential.

7.2 Visual SLAM Fundamentals

Before we go into detail on visual SLAM let us introduce a few fundamentals.

7.2.1 Camera Model

A parameterized description of the sensor that models image formation from the observed scene should include a **geometric component** (also called the projection function), which describes how 3D points are mapped to 2D pixels, and a **photo-**

Figure 7.1 A central requirement for visual SLAM systems is the choice of a suitable lens. Shown here are BF2M2020S23 (195°), BF5M13720 (183°), BM4018S118 (126°), BM2820 (122°), and a GoPro replacement lens (150°). Fish-eye and wide angle lenses offer a wider field of view, but require suitable projection models. Popular choices are the Brown-Conrady (BC) model [291], the Kannala-Brandt (KB) model [545] and the Double Sphere (DS) model [1121]. The 6-parameter DS model provides a comparable reprojection accuracy as the 8-parameter KB model while offering around five times faster computation time for the projection function.

metric component, which describes how physical light intensity (radiance) maps to pixel values.

7.2.1.1 Geometric Camera Models

Perspective Cameras

The projection function is generically referred to as:

$$\mathbf{z} = \pi(\mathbf{x}^c, \boldsymbol{\xi}),$$

where $\mathbf{x}^c = \begin{bmatrix} x & y & z \end{bmatrix}^\top \in \mathbb{R}^3$ is a 3D point in camera coordinates, $\mathbf{z} = \begin{bmatrix} u & v \end{bmatrix}^\top \in \Omega \subset \mathbb{R}^2$ are the coordinates of the corresponding 2D point in the image domain Ω, and $\boldsymbol{\xi} \in \mathbb{R}^n$ represents the intrinsic parameters of the camera, typically pre-calibrated in visual SLAM. The dimensionality of $\boldsymbol{\xi}$ depends on the used *camera model*.

Conversely, the unprojection operation is denoted as:

$$\mathbf{x}^c = \pi^{-1}(\mathbf{z}, \boldsymbol{\xi}),$$

which reconstructs a ray in 3D from a 2D image point $\mathbf{z}$. Since the depth of the point remains unknown, the resulting $\mathbf{x}^c$ is only known up to scale.

In practice, there exists a wide range of projection functions suitable for different lens geometries and camera types (see some, for example, in [694]). Rectilinear models (also known as pinhole- or perspective camera model) are the simplest and can be used for narrow-angle lenses without distortion

$$\boldsymbol{\xi}_p = \begin{bmatrix} f_u & f_v & u_0 & v_0 \end{bmatrix}^\top,\ \pi_p(\mathbf{x}^c, \boldsymbol{\xi}) = \begin{bmatrix} f_u \frac{x}{z} + u_0 \\ f_v \frac{y}{z} + v_0 \end{bmatrix},\ \pi_p^{-1}(\mathbf{z}, \boldsymbol{\xi}) \sim \begin{bmatrix} \frac{u-u_0}{f_u} \\ \frac{v-v_0}{f_v} \\ 1 \end{bmatrix},$$

where f_u and f_v stand for the focal length in horizontal and vertical direction, in pixel units. u_0 and v_0 stand for the 2D coordinates of the principal point, and we assumed rectangular pixels. With typical, square pixels, we expect $f_u \approx f_v$.

To account for some amount of lens distortion, the radial-tangential model is most commonly used:

$$\boldsymbol{\xi}_{RT} = \begin{bmatrix}\boldsymbol{\xi}_p^\top & k_1 & k_2 & p_1 & p_2\end{bmatrix}^\top, \ \begin{bmatrix}x' \\ y'\end{bmatrix} = \begin{bmatrix}\frac{x}{z} \\ \frac{y}{z}\end{bmatrix}, \ r = \sqrt{x'^2 + y'^2},$$

$$\begin{bmatrix}x'' \\ y''\end{bmatrix} = \begin{bmatrix}x'(1 + k_1 r^2 + k_2 r^4) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\ y'(1 + k_1 r^2 + k_2 r^4) + p_1(r^2 + 2y'^2) + 2p_2 x' y'\end{bmatrix}, \ \pi_p(\mathbf{x}^c, \boldsymbol{\xi}) = \begin{bmatrix}f_u x'' + u_0 \\ f_v y'' + v_0\end{bmatrix},$$

Note that we cannot extract an analytical expression for the unprojection model $\pi_{RT}^{-1}(\mathbf{z}, \boldsymbol{\xi})$, since there is no analytical solution for x', y' as a function of x'', y'' (i.e. undistortion). We may, however, resort to iterative approaches e.g. the Newton-Raphson method.

Wide-angle and Fisheye Cameras

For wide-angle lenses – see Figure 7.1 – up to fields of view (FOV) of 180°, pinhole models with a few radial distortion coefficients typically suffice. The Brown-Conrady model [291] amounts to the radial-tangential model described above without tangential coefficients, i.e. $\boldsymbol{\xi}_{BC} = \begin{bmatrix}\boldsymbol{\xi}_p^\top & k_1 & k_2\end{bmatrix}$. Note that despite the simplification relative to the radial-tangential model, no analytical undistortion exists.

For fish-eye lenses with FOVs larger than 180°, the Kannala-Brandt (KB) model [545] has been used, for example in [148].

$$\boldsymbol{\xi}_{KB} = \begin{bmatrix}\boldsymbol{\xi}_p^\top & \boldsymbol{k}_{KB}^\top\end{bmatrix}^\top, \ \pi_{KB}(\mathbf{x}^c, \boldsymbol{\xi}) = \begin{bmatrix}f_u r(\theta)\cos\psi + u_0 \\ f_v r(\theta)\sin\psi + v_0\end{bmatrix}^\top$$

Here, the incoming ray is parametrized by the angles $\theta = \arctan\frac{\sqrt{x^2+y^2}}{z}$ and $\psi = \arctan\frac{y}{x}$, distortion is parametrized by four coefficients $\boldsymbol{k}_{KB} = \begin{bmatrix}k_1 & \dots & k_4\end{bmatrix}^\top$ and the distortion expression is $r(\theta) = \theta + \sum_1^4 k_n \theta^{2n+1}$.

For fish-eye and wide angle lenses, a popular alternative is the Double Sphere (DS) model [1121] as it offers a good compromise of accuracy and speed. In the DS model a point is consecutively projected onto two unit spheres with centers shifted by γ. Then, the point is projected onto the image plane using the pinhole model shifted by $\frac{\alpha}{1-\alpha}$. This leads to:

$$\pi_{DS}(\mathbf{x}^c, \boldsymbol{\xi}) = \begin{bmatrix}f_u \frac{u}{\alpha d_2 + (1-\alpha)(\gamma d_1 + z)} + c_u \\ f_v \frac{v}{\alpha d_2 + (1-\alpha)(\gamma d_1 + z)} + c_v\end{bmatrix}^\top, \text{with } \boldsymbol{\xi} = \begin{bmatrix}\boldsymbol{\xi}_p^\top & c_u & c_v & \gamma & \alpha\end{bmatrix}.^\top$$

As shown in [1121], this model offers a closed form unprojection solution. As a

consequence, the 6-parameter DS model is around five times faster to compute than the 8-parameter KB model while offering a comparable reprojection error.

7.2.1.2 Photometric Models

The photometric calibration maps irradiance to pixel values:

$$I = f(E, T),$$

where I is the pixel intensity, E is the irradiance, and T contains other camera properties that affect this mapping such as exposure time, analog and digital gain, gamma correction, de-bayering, and lens vignetting. This photometric calibration is typically only important up to scale and for methods that aim to obtain dense, textured scene representations or that rely on photo-consistency across images.

7.2.1.3 Time-Dependent Effects

In situations where the camera is moving while the image is taken–which is always almost the case for SLAM or Visual Odometry systems–it is important to also consider the effects of this motion on the image formation process. Most modern consumer cameras use a rolling shutter, which captures image-rows sequentially in time. In contrast, global shutter cameras, which are often used for machine perception applications, capture all image rows simultaneously.

In practice, it is advisable to either use global shutter cameras, or include the rolling shutter effect into the camera model by varying the camera pose for individual image rows. Note that this becomes significantly more practical in visual-inertial systems, where the IMU effectively measures local motion with a cameras exposure window, and thus simplifies the use and modeling of rolling shutter cameras significantly.

7.2.1.4 Practical Considerations

When selecting a camera model, the primary goal is to ensure that the parametric model can effectively and accurately approximate the behavior of the sensor and lens system. When using Fisheye lenses, using a spherical model is recommended–while for rectilinear lenses, linear base-models should be used. More generally, choosing a camera model involves balancing computational efficiency against precision: using an ill-suited camera model can introduce inaccuracies and systematic biases, significantly degrading the visual SLAM system's accuracy and robustness.

7.2.2 Keypoints

The success of SLAM systems in mapping environments and providing accurate localization hinges on their ability to detect, describe, and match key features in a scene—commonly referred to as *'keypoints'* or *'features'*. In vision-based SLAM,

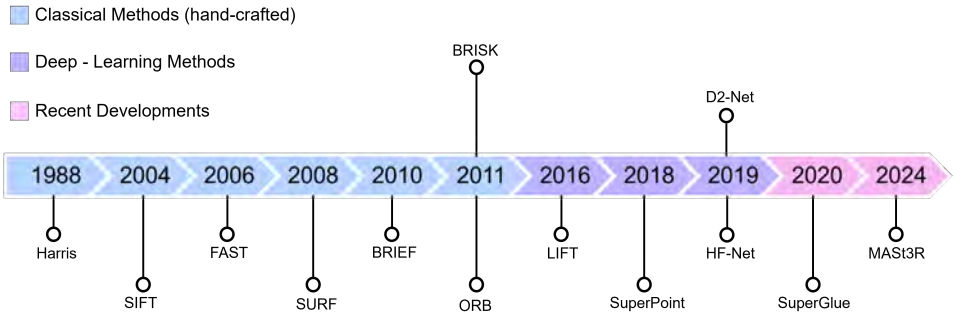

Figure 7.2 Timeline of some of the most prominent algorithms shaping the literature on visual keypoints for vision-based SLAM and image matching.

keypoint detection involves identifying salient image regions that are distinctive and repeatable, ensuring reliable re-detection from different viewpoints, across multiple runs, and under varying conditions. An ideal **keypoint detector** should maintain these properties regardless of changes in illumination, viewpoint, or occlusions. Once detected, a **keypoint descriptor** encodes the local appearance of the corresponding keypoint detection into compact and distinctive representations. Ideally, a descriptor should uniquely characterize a keypoint's surroundings while remaining invariant to transformations such as lighting changes, rotation, or scale variations. This ensures both high recall—matching the same keypoint across different conditions—and high precision—avoiding incorrect correspondences with similar-looking but unrelated features.

In practice, real-world challenges such as drastic illumination variations, textureless surfaces, and dynamic scenes introduce errors in detection and matching, directly impacting SLAM performance. Consequently, research has focused on designing keypoints with specific characteristics to enhance robustness. The most desirable properties include **scale and rotation invariance** (ensuring consistent detection despite viewpoint changes), **repeatability and distinctiveness** (allowing reliable re-detection and unique identification), and **efficiency** (enabling real-time operation under computational constraints). To meet these demands, keypoint detection and description have evolved from classical handcrafted techniques to deep-learning-based approaches more recently. The following subsections provide an overview of the most prominent keypoint methods in the literature, as illustrated in Figure 7.2, along with emerging trends in the field.

7.2.2.1 Classical Keypoint Detectors & Descriptors

Classical hand-crafted techniques have played a pivotal role in the evolution of feature detection and description. Among them, the Harris-Stephens keypoint detector [439], widely known as the 'Harris corner detector', is one of the most influential

methods. It identifies corners by analyzing the eigenvalues of the second-moment matrix within image patches, classifying a patch as a keypoint when both eigenvalues are large, indicating strong intensity variations in two orthogonal directions. The Shi-Tomasi corner detector [1010] is built on the same principle but directly uses the smaller eigenvalue for corner selection. In contrast, Harris defines a 'cornerness' response function to approximate the process for efficiency. While robust and computationally efficient, the Harris detector lacks scale invariance, a limitation that later methods, such as SIFT and SURF, sought to address.

A seminal milestone in keypoint detection was the introduction of the Scale Invariant Feature Transform (SIFT) [706], which set a new standard for precision and recall across challenging settings. SIFT is highly invariant to scale and rotation and partially invariant to illumination changes. It follows a structured three-stage process: (1) detecting keypoints as extrema in a scale-space Difference of Gaussians (DoG) pyramid and refining their localization through a 3D quadratic fit, (2) assigning orientation based on the dominant local image gradient, and (3) computing a 128-dimensional descriptor from a histogram of discretized gradient orientations. However, SIFT's exceptional robustness comes at a high computational cost, making it less suitable for real-time applications.

To improve efficiency, Features from Accelerated Segment Test (FAST) [951] was developed as a high-speed corner detector. It assesses pixel intensities in a circular neighborhood around the candidate keypoint location and employs an early rejection strategy to minimize computations. FAST further enhances speed using a decision tree and non-maximum suppression. However, unlike SIFT, it lacks scale invariance, making it sensitive to significant transformations. Speeded-Up Robust Features (SURF) [70] later improved efficiency by approximating SIFT using integral images and box filters instead of Gaussian derivatives, achieving a better balance between speed and robustness.

The need for fast yet robust descriptors led to the development of binary-based methods. Binary Robust Independent Elementary Features (BRIEF) [147] introduced a compact descriptor using binary intensity comparisons, enabling fast descriptor matching using the Hamming distance. While BRIEF lacks scale and rotation invariance, it demonstrated that local gradients, fundamental to SIFT's robustness, could be effectively captured through simplified binary tests. Inspired by this success, ORB [958] was proposed, building upon FAST keypoint detection and BRIEF description, while adding scale and rotation invariance through an image pyramid and intensity centroid method. At the same time, BRISK [650] was proposed, employing FAST or Harris corners on a scale-space pyramid, and incorporating invariance to rotation changes within a binary descriptor by identifying a dominant keypoint direction similar to SIFT. While the added rotation- and scale-invariance of methods such as ORB and BRISK have a small impact on the distinctiveness of the output keypoints, they have proven effective in vision-based SLAM and real-time robotics applications. When computational cost, however, is

not a requirement (e.g. in Computer Vision applications), SIFT and SURF might still be preferable as they still offer greater robustness under challenging lighting or perspective changes.

With several variants of these methods appearing in the literature, classical hand-crafted keypoint detection and description methods have been foundational in Computer Vision and Robotics, carefully balancing robustness, efficiency, and invariance. However, the demand for keypoints that adapt to increasingly complex, dynamic environments remains an ongoing challenge. This has driven research toward learning-based approaches, which aim to automatically optimize keypoints for real-world applications, setting the stage for the next generation of feature detection techniques.

7.2.2.2 Deep-Learning-based Keypoint Detection & Description

By the late 2010s, deep learning-based approaches for image keypoints began to gain traction, utilizing large-scale datasets and Convolutional Neural Networks (CNNs) to learn robust features directly from data. Unlike manually designed keypoints, these methods automatically discover and optimize feature representations, demonstrating unparalleled adaptability and accuracy in scenarios that were previously infeasible. For instance, they have shown remarkable success in detecting stable keypoints even under extreme illumination changes–an area where classical hand-crafted methods struggled. Consequently, the visual SLAM community has increasingly shifted away from traditional feature engineering, embracing data-driven representation learning for keypoint detection and description.

The Learned Invariant Feature Transform (LIFT) [1240] was one of the first deep learning-based approaches to integrate keypoint detection, orientation estimation, and descriptor computation into an end-to-end trainable pipeline. Using CNNs to extract features from small image patches, LIFT achieved greater robustness to scale, illumination, and rotation changes than classical methods. It employs a sequential learning strategy, training descriptors first, followed by orientation estimation and then keypoint detection, ensuring stable and effective feature extraction. Similarly, SuperPoint [275] introduced a self-supervised framework that detects keypoints and computes descriptors in a single forward pass, making it efficient for real-time applications. Unlike patch-based networks, SuperPoint operates on entire images and leverages homographic adaptation, a self-supervised learning technique to generate pseudo-ground truth keypoints. This approach significantly improves keypoint repeatability and descriptor quality compared to classical methods, such as SIFT, ORB, and FAST, especially under illumination changes.

Building on these advancements, HF-Net [975] introduced a hierarchical localization approach that combines global image retrieval with precise local feature matching. HF-Net improves computational efficiency while maintaining high robustness by integrating keypoint detection, local descriptors, and global descriptors into a single CNN. This architecture reduces runtime by limiting the number of images

used for matching, making it particularly effective for large-scale SLAM and real-time applications, even under extreme appearance variations such as night-time scenes. HF-Net's learned features are sparser but more discriminative than those of SuperPoint, rendering it a preferred choice for deep learning-based SLAM systems such as DX-SLAM [656].

Interestingly, D2-Net [303] introduced a describe-and-detect approach that reverses the traditional keypoint detection and descriptor extraction order. Instead of detecting keypoints first, D2-Net computes dense feature maps using a CNN and then identifies keypoints as local maxima within these maps. This method captures high-level semantic information, making it robust to extreme lighting changes and weakly textured environments. Unlike SuperPoint, which separates detection and description, D2-Net jointly optimizes both tasks, enhancing descriptor consistency. However, while this dense approach improves robustness, it is computationally more demanding than classical sparse methods. Despite this trade-off, D2-Net remains highly effective for visual localization and Structure from Motion (SfM) tasks, pushing the boundaries of deep learning-based feature extraction.

7.2.2.3 Latest algorithms & trends

With the unprecedented invariance and adaptability of learning-based keypoints over traditional handcrafted counterparts in specific settings, deep-learning-based techniques for keypoint extraction have been transforming the field. This shift has not only enhanced keypoint detection and description but has also driven significant advancements in keypoint matching techniques. Rather than relying on manually designed heuristics for descriptor matching, as in classical methods, learning-based approaches now incorporate global spatial awareness to establish correspondences more robustly. In particular, techniques such as SuperGlue [976] and MASt3R [649] leverage Graph Neural Networks (GNNs) and Transformers to refine matches by considering the broader image structure. These methods address fundamental limitations of traditional matchers by adapting to extreme viewpoint changes, illumination variations, and occlusions, where conventional local descriptor comparisons typically struggle.

SuperGlue enhances feature matching by integrating self-attention and cross-attention mechanisms, allowing it to resolve ambiguities caused by repetitive textures and occlusions. Unlike traditional keypoint matchers that operate on local descriptors alone, SuperGlue incorporates contextual information, significantly improving accuracy in both indoor and outdoor environments. Its optimal matching layer further ensures that keypoint correspondences are established robustly while allowing unmatched keypoints when necessary, making it highly effective for real-world scenarios. Similarly, MASt3R extends this idea into 3D-aware feature matching, reconstructing a 3D scene representation from two images to improve performance in textureless regions and under extreme viewpoint changes. Built upon this, MASt3R-SLAM [795] integrates these advancements into a SLAM pipeline,

improving camera pose estimation, global map consistency, and loop closure strategies. By transitioning from handcrafted 2D feature matching to learning-based, context-aware, and 3D-informed approaches, these methods mark a paradigm shift in visual SLAM, enhancing scene understanding, tracking robustness, and long-term stability in complex environments.

Overall, the evolution of keypoint extraction for image matching and vision-based SLAM has been remarkable from traditional hand-crafted methods to deep-learning-based approaches that promise greater robustness in challenging conditions. While classical methods remain very relevant to date due to their efficiency and interpretability, they struggle in scenarios with textureless surfaces, extreme viewpoint changes, and illumination variations. Deep learning has addressed these limitations, driving research towards more adaptive and context-aware keypoint detection and matching techniques. However, challenges remain in balancing computational efficiency with real-time performance, particularly on resource-constrained platforms, and ensuring the availability of diverse, unbiased training datasets. Future research will likely focus on optimizing these techniques to maximize both accuracy and efficiency, making them more viable for large-scale real-world applications.

7.2.3 Reprojection Error

The visual reprojection error measures the discrepancy between observed image points $\boldsymbol{z}_j \in \mathbb{R}^2$ and reprojected points $\pi(\boldsymbol{x}_j^C, \boldsymbol{\xi}) \in \mathbb{R}^2$:

$$\boldsymbol{e}_{\text{reproj}} = \boldsymbol{z}_j - \pi(\boldsymbol{x}_j^C, \boldsymbol{\xi}),$$

where $\boldsymbol{z}_j$ is the observed 2D image point, $\boldsymbol{x}_j^C \in \mathbb{R}^3$ is the 3D point in camera coordinates, $\boldsymbol{\xi}$ are the camera intrinsic calibration parameters, and π is the projection function.

Assuming the observed image points $\boldsymbol{z}_j$ are perturbed by Gaussian noise, the likelihood function can be expressed as:

$$p(\boldsymbol{z}_j|\boldsymbol{x}_j^C, \boldsymbol{\xi}) \sim \mathcal{N}\left(\pi(\boldsymbol{x}_j^C, \boldsymbol{\xi}), \boldsymbol{\Sigma}_j\right),$$

where $\boldsymbol{\Sigma}_j$ is the covariance of the Gaussian noise of the position of the feature in the image. In the simplest case, the noise is assumed to be isotropic and constant along the image $\boldsymbol{\Sigma}_j = \sigma \boldsymbol{I}_2$.

Maximizing the likelihood of a set of observations is equivalent to minimizing their negative log-likelihood:

$$\mathcal{L} = -\sum_j \log p(\boldsymbol{z}_j|\boldsymbol{x}_j^C, \boldsymbol{\xi}) = \frac{1}{2}\sum_j \left\|\boldsymbol{z}_j - \pi(\boldsymbol{x}_j^C, \boldsymbol{\xi})\right\|_{\boldsymbol{\Sigma}_j}^2 + \text{const.}$$

Removing the constant, we get the weighted-squared reprojection error of the set of points observed in an image:

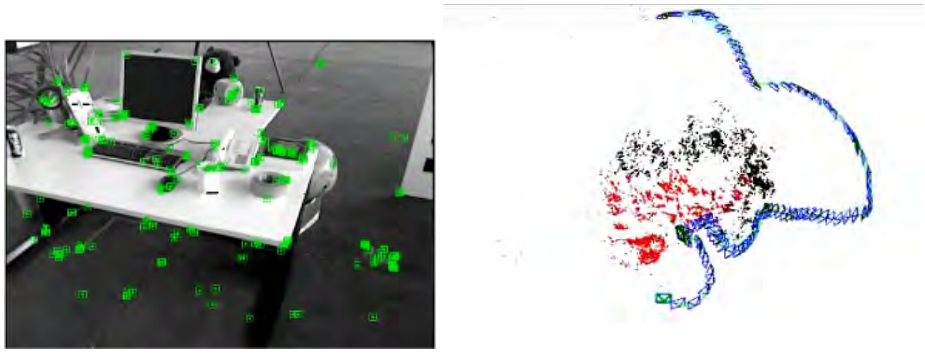

Figure 7.3 The feature-based visual SLAM problem: given a set of features matched in each image (left), estimate the position of their corresponding 3D points and the pose from where each image was acquired (right). Images generated with ORB-SLAM [793].

$$E_{\text{reproj}} = \frac{1}{2} \sum_j \left\| \boldsymbol{z}_j - \pi(\boldsymbol{x}_j^C, \boldsymbol{\xi}) \right\|^2_{\boldsymbol{\Sigma}_j} . \tag{7.1}$$

To handle outliers, robust kernels such as the Huber or Tukey kernel are applied. For example, the robust reprojection error can be expressed as:

$$E_{\text{robust}} = \frac{1}{2} \sum_j \rho \left(\left\| \boldsymbol{z}_j - \pi(\boldsymbol{x}_j^C, \boldsymbol{\xi}) \right\|_{\boldsymbol{\Sigma}_j} \right) , \tag{7.2}$$

where $\rho(\cdot)$ is a robust kernel function that reduces the influence of large residuals. In the following, for simplicity, we will drop the dependence with the camera intrinsics $\boldsymbol{\xi}$.

7.2.4 Keypoint-Based Visual SLAM

The core of feature-based visual SLAM is minimizing the reprojection error. Suppose we have a set of environment points $P_j \in \mathsf{P}$ that are observed in a set of images $C_i \in \mathsf{C}$ taken with a moving camera. To make notation simpler, we will just use the point and camera identifiers writing $j \in \mathsf{P}$ and $i \in \mathsf{C}$. The goal of visual SLAM is to estimate the point coordinates $\boldsymbol{x}_j^w \in \mathbb{R}^3$ and the camera poses $\boldsymbol{T}_w^i \in \text{SE}(3)$, both expressed in a world reference frame $\mathcal{F}^w$. In the monocular case, the observation of each point in each image is just the observed image coordinates $\boldsymbol{z}_{ij} \in \Omega_i \subset \mathbb{R}^2$ (Figure 7.3).

The minimization of the reprojection error, when applied to all the points and camera poses, is known as *full BA*:

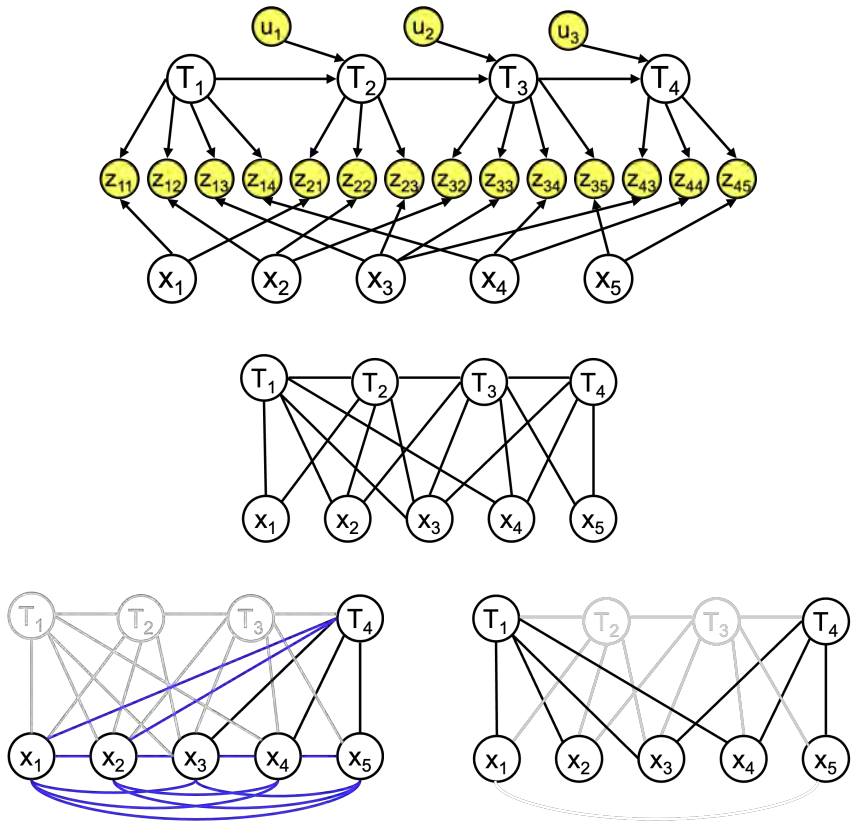

Figure 7.4 A visual SLAM example with four camera poses and five features: Bayes network (top), and its corresponding Markov random field (middle). EKF SLAM marginalizes out past camera poses, resulting in a dense graph (lower left). Keyframe SLAM keeps just a few camera poses and discards observations from intermediate images, keeping SLAM sparsity (lower right) [1044].

$$\{\boldsymbol{T}_w^{i\,*},\, \boldsymbol{x}_j^{w*} \mid i \in \mathsf{C},\, j \in \mathsf{P}\} = \underset{\boldsymbol{T}_w^i,\, \mathbf{x}_j^w}{\arg\min} \frac{1}{2} \sum_{i,j} \rho\left(\left\| \boldsymbol{z}_{ij} - \pi\left(\boldsymbol{R}_w^i \boldsymbol{x}_j^w + \boldsymbol{t}_w^i \right) \right\|_{\boldsymbol{\Sigma}_{ij}} \right). \quad (7.3)$$

Typically, researchers tackle the reprojection error optimization via several advanced methods, each offering unique trade-offs in computational complexity and robustness:

- **Batch Optimization**: The overall reprojection error is iteratively minimized over all observations w.r.t. all poses and landmarks, i.e. solving Equation (7.3). Popular algorithms for minimization are the Gauss-Newton (GN) and Levenberg-

Marquardt (LM). This method is the gold standard in SfM, but is too expensive to run for each image in real-time SLAM.

- **Filtering-Based Approaches**: Use sequential and causal methods like the Extended Kalman Filter (EKF) or Information Filter for real-time state estimation. They simplify the problem by keeping only the last camera pose, but unfortunately this destroys sparsity (Figure 7.4), limiting them to a few hundred features. Also, filtering methods do not re-linearize past observations, loosing accuracy.
- **Keyframe-Based Approaches**: They simplify the problem by keeping just a few images, called keyframes [591]. Intermediate images and their observations are discarded for the map estimation. For the same computational effort, they can build longer and more accurate maps than filtering methods [1044].
- **Factor Graphs**: All the above approaches can be formalized by means of factor graphs. To this end, one represents SLAM problems as graphs where nodes correspond to variables (e.g., poses, landmarks) and factors represent the reprojection errors and priors on calibration and/or camera poses, which is discussed in detail in part I of this book.

7.2.5 Photometric Error and Direct Methods

The photometric error provides an alternative to the reprojection error by offering to directly minimize the differences in pixel intensities between observed and projected image regions. This approach is rooted in the principle of photometric consistency, which assumes that corresponding pixels in consecutive frames represent the same scene point under consistent lighting conditions.

7.2.6 Visual Place Recognition and Global Localization

Visual Place Recognition consists of, given a query image, finding one from the same place in a database of registered images. This is typically solved by computing a per-image descriptor $\mathbf{d} = f(I) \in \mathbb{R}^d$ that summarizes the content of the image, and retrieving the closest one in this descriptor space via k-NN search. Galvez-Lopez and Tardos [370] introduced bag-of-words approaches that basically aggregate ORB or other descriptors by their quantization into visual clusters or "words". While they excel at small temporal and spatial ranges, they are limited by the low invariance of hand-crafted features to variations of visual textures. For these cases, deep architectures have been proposed and trained for feature extraction and aggregation with high invariance to visual appearance changes [42, 516].

7.2.7 Initialization

The minimization of the reprojection error of Equation 7.3 is typically addressed by non-linear iterative optimization, which requires sufficiently accurate initial guesses

to converge. The initialization of the visual SLAM states in the first frames of a video sequence is hence relevant for a correct tracking, in particular for monocular setups, where the full state is not observable from a single frame.

7.2.8 Common Steps

1 Feature Matching: The system matches current observations to landmarks or keypoints stored in the map. Efficient descriptor matching techniques, such as ORB or SuperPoint, are typically used.
2 Pose Estimation: Using the matched features, the system estimates the camera's pose relative to the map, often leveraging robust optimization techniques to account for outliers.
3 Validation: The estimated pose is validated by checking consistency with the map and potentially re-optimizing if mismatches occur.

7.2.9 Map Representations

Map representations are a critical aspect of visual SLAM systems, as they define how the environment is modeled and stored for navigation, mapping, and localization purposes. A well-designed map representation strikes a balance between accuracy, memory efficiency, and computational cost.

7.3 The Processing Pipeline of a Visual SLAM System

Building a complete Visual SLAM system involves combining various components into a cohesive framework that can handle the demands of real-time operation, scalability, and robustness. Key considerations include deciding when and how each component operates, structuring the compute- and data-flow efficiently, and ensuring adaptability to diverse environments. As in LiDAR SLAM, the problem of visual SLAM can be tackled in several stages. In contrast to LiDAR-based systems, however, the 3D geometry is not directly measured since one rather observes projections of the scene irradiance onto the screen. That makes the overall estimation problem more challenging. Modern complete Visual SLAM systems typically include three core sub-functionalities that complement each other: an odometry front-end, a mapping back-end and a loop closure and re-localization component.

7.3.1 Visual Odometry Front-End

The core element of a Visual SLAM system is visual odometry which aims at estimating relative motion between consecutive camera frames. This stage provides the

initial estimate for the camera's pose. As discussed above, there are two alternative approaches to capture the visual odometry: 1. Feature-based approaches split the challenge into three stages of detecting and extracting feature points, computing pairwise correspondence across images and subsequently determine the relative camera motion by minimizing the re-projection error with respect to camera motion and 3D point coordinates. The last stage is quite analogous to classical bundle adjustment. 2. Direct approaches tackle the problem in one step where a photometric loss function is directly optimized with respect to camera motion and 3D structure. They are therefore quite related to approaches of optical flow and what is sometimes called photometric bundle adjustment.

At least in their naive, first formulations, feature-based methods have demonstrated a larger basin of convergence thanks to explicit data associations. To alleviate this, direct methods thus often employ a corse-to-fine approach, i.e. start with aligning down-sampled images, or even attempt to align dense (learned) features instead of brightness or color.

7.3.2 Mapping Back-End

The back-end optimizes the trajectory and map using global optimization techniques like bundle adjustment or factor graph solvers. This step refines the estimates provided by the front-end and integrates observations into a consistent map. As a consequence, one obtains more long-term consistency, and long-range distortions are reduced.

7.3.3 Visual Place Recognition and Relocalization

Visual odometry is prone to drift because errors in the camera tracking will aggregate over time. In the absence of additional sensors like IMUs, wheel odometry or GPS, one can eliminate drift and enforce global consistency by aligning the current image to previously observed images. To this end, one needs to compute correspondence across a potentially large set of images. This can be done either by an efficient matching of classical feature descriptors like SIFT, SURF or BRIEF–or by means of suitable trained neural networks, an approach that has become increasingly popular in the last years. The resulting component detects when the camera revisits a previously mapped area (loop closure detection), correcting accumulated drift and re-establishing localization when tracking fails.

7.3.4 Compute and Data Flow

Efficient data flow is essential for a well-performing SLAM system:

Pipeline Parallelism: Different components, such as tracking, mapping, and optimization, often run in parallel to maximize efficiency.

Data Sharing: Intermediate outputs, like keypoints or poses, are shared between components to minimize redundant computation.

Adaptive Scheduling: Compute-heavy tasks, like global optimization, are scheduled based on system requirements, prioritizing real-time responsiveness.

7.3.5 Keypoint-based Image Alignment

Although full BA (equation 7.3) is the gold standard in SfM, it is too expensive to run at frame rate, which is typically between 10 and 50 Hz. To operate in real-time, most keypoint-based visual SLAM pipelines use two key ideas:

Parallel Tracking and Mapping: splitting the SLAM process into two threads that run in parallel [591]:

- A tracking thread that finds feature matchings for the current image $i \in \mathsf{C}$ and computes its camera pose, without updating the estimated map points, using *pose-only BA*:

$$\boldsymbol{T}_w^{i\,*} = \arg\min_{\boldsymbol{T}_w^i} \sum_{j \in \mathsf{P}} \rho\left(\left\| \boldsymbol{z}_{ij} - \pi\left(\boldsymbol{R}_w^i \boldsymbol{x}_j^w + \boldsymbol{t}_w^i \right) \right\|_{\boldsymbol{\Sigma}_{ij}} \right). \tag{7.4}$$

- A mapping thread that solves BA only for a subset of images $\mathsf{K} \subset \mathsf{C}$ called *keyframes*, whose poses are the only ones that will be included in the map. In this way, BA only needs to run at keyframe rate, typically between 0.5 and 5 Hz. Keyframes can be inserted at a constant frequency, but a more sensible option is to upgrade to keyframes those frames that contain significantly new information.

Locality: when the camera is operating in a large environment, its observations have a negligible effect on the parts of the map that are far away, except in loop closure events. The usual approach is to relegate loop correction to a third thread that runs quite infrequently, and to run *local BA* in a window of keyframes in the mapping thread. The local window can be defined using a temporal criterion as the last k frames or keyframes, which is the usual choice in visual odometry or visual-inertial SLAM systems. In visual SLAM systems, a better option is to base the local window on a *covisibility* criterion, for example, including in the local window keyframes that have more than θ observed points in common with the current keyframe [1043, 793]

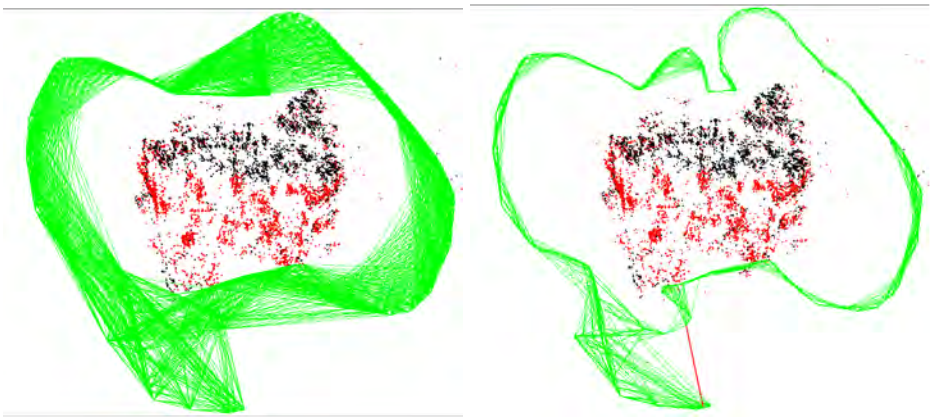

Figure 7.5 Representation of locality in ORB-SLAM [793]. The covisibility graph (left) connects keyframes that have seen at least θ points in common (in this example $\theta = 15$) and is used for local BA. The essential graph (right) is a sparser version, that in this example connects keyframes with at least $\theta = 100$ points in common, and is used for pose-graph optimization during loop correction. ©2015 IEEE.

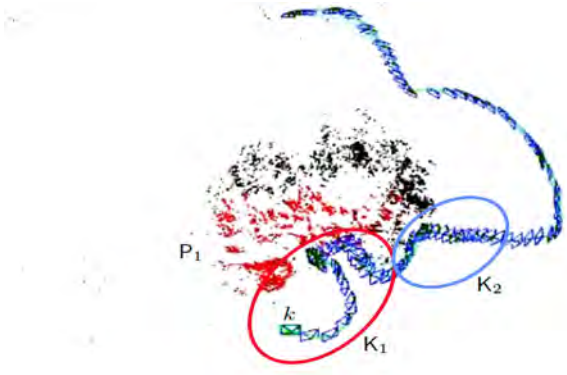

Figure 7.6 Implementation of local BA in ORB-SLAM [793]. The local map is defined by the set of keyframes K_1 that contains the current keyframe k and its neighbors in the covisibility graph, and the set P_1 of points seen by them (in red). K_2 is the rest of keyframes in the map that see some point from P_1.

(see example in figure 7.5). In that way, *local BA* can update just a set of covisible

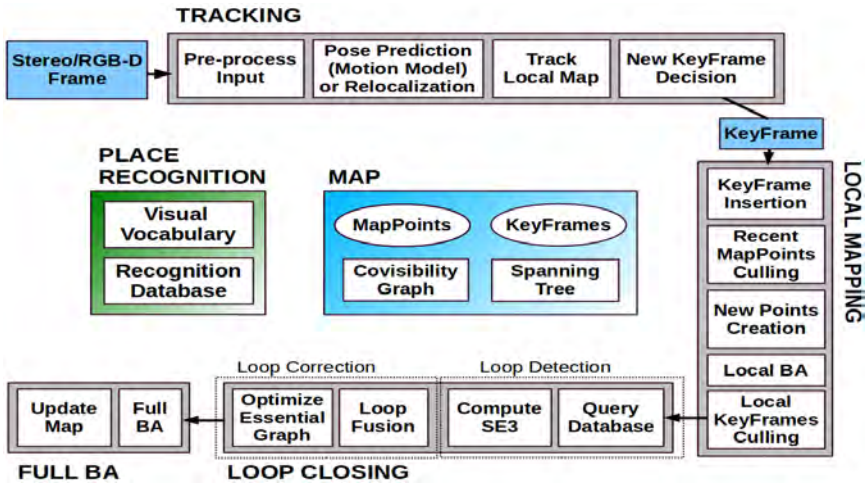

Figure 7.7 Structure of ORB-SLAM2 system [791] showing the map, the place recognition database and its complete processing pipeline composed of four threads: tracking, local mapping, loop closing and full BA. ©2017 IEEE.

keyframes and the points observed by them (figure 7.6):

$$\{\boldsymbol{T}_w^{k*}, \boldsymbol{x}_j^{w*} \mid k \in \mathsf{K}_1,\, j \in \mathsf{P}_1\} = \underset{\boldsymbol{T}_w^k,\, \mathbf{x}_j^w}{\arg\min} \frac{1}{2} \sum_{\substack{i \in \mathsf{K}_1 \cup \mathsf{K}_2, \\ j \in \mathsf{P}_1}} \rho\left(\left\| \boldsymbol{z}_{ij} - \pi\left(\boldsymbol{R}_w^i \boldsymbol{x}_j^w + \boldsymbol{t}_w^i \right) \right\|_{\boldsymbol{\Sigma}_{ij}} \right). \tag{7.5}$$

An example of a complete visual SLAM pipeline can be seen in figure 7.7 with four threads: tracking that runs at frame rate, local mapping that runs at keyframe rate, loop closing that tries to detect loops for every keyframe and corrects them when detected, and full BA that can be run optionally to improve the map after a loop closure.

7.3.6 Direct Image Alignment

Direct methods such as LSD SLAM [315] or DSO [316] typically pursue a similar pipeline of first estimating a frame-to-frame tracking and mapping and then assuring some form of global consistency. Rather than first extracting, matching and tracking points and subsequently minimizing a geometric reprojection error, however, they directly use the brightness information from the sensor and aim for computing the maximum a posteriori estimate of 3D structure and camera motion given the raw sensory data. This amounts to solving the correspondence estimation

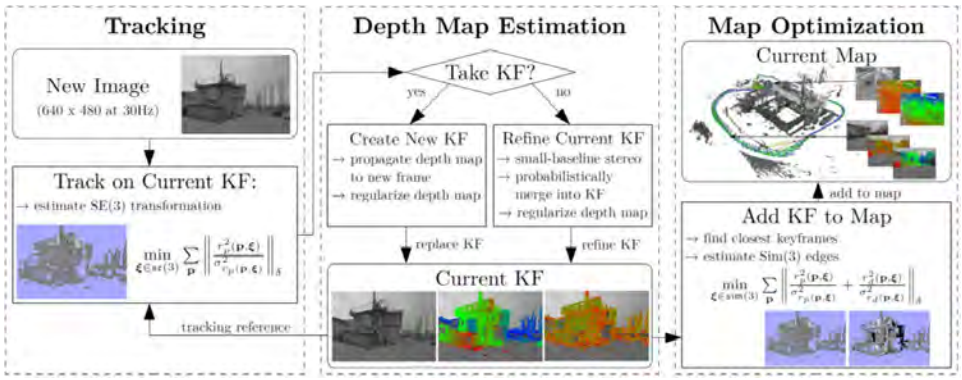

Figure 7.8 Schematic overview of Large Scale Direct (LSD) SLAM [315] showing the three components that perform direct camera tracking, direct mapping and pose graph optimization for global consistency, all running in alternation. In contrast, Direct Sparse Odometry [316] performs the optimization of structure and motion in a single Gauss-Newton optimization in order to achieve even higher precision. Direct methods like DSO were shown to provide higher precision than keypoint-based methods because they do not perform any intermediate abstraction and can determine camera motion from even very subtle brightness variations [316].

and SLAM problem jointly by minimizing a photometric loss of the form:

$$E_{photo} = \sum_{i \in \mathcal{F}} \sum_{z \in \mathcal{P}_i} \sum_{j \in obs(z)} \rho\Big(I_i(z) - I_j(\omega(z, d_z, \boldsymbol{T}_j^i)\Big). \tag{7.6}$$

with respect to all camera parameters $\boldsymbol{T}_j^i \in \mathrm{SO}(3)$ and all depth values $d_z \in \mathbb{R}$. This loss assures that the colors I_i and I_j of corresponding points in all frame pairs i and j are consistent. More specifically, we sum over the set of all keyframes $\mathcal{F}$ and for every point z in keyframe $\mathcal{P}_i$, we assure color consistency for all the frames $obs(z)$ where this point is visible. The warping w takes the point z with its depth value d_z, transforms it from frame i to frame j with the rigid body motion $\boldsymbol{T}_j^i$ and perspectively projects it back into image I_j.

As shown in Figure 7.8, LSD SLAM optimizes for depth maps and camera motion (tracking) in alternation and performs a pose graph optimization to assure global consistency of the estimated camera motion with the computed pairwise image alignments.

In contrast, DSO [316] jointly optimizes for structure and motion in the form of a photometric bundle adjustment. The robust loss ρ is implemented by weighted sum of squared differences over a small patch that includes an automatic exposure time adaptation (in case the exposure time is unknown). The dependency of respective residuals is captured in form of factor graphs. And real-time performance on a CPU is achieved by limiting the optimization to a sliding window of a subset of keyframes while marginalizing out the effect of older frames. This leads to a signifcant boost in

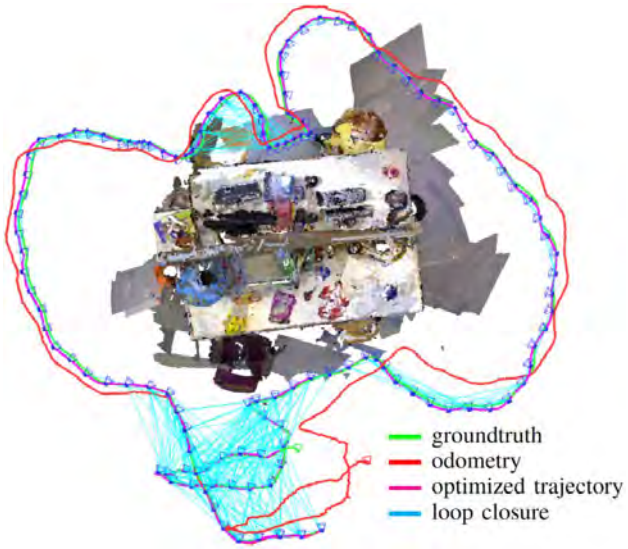

Figure 7.9 In direct visual SLAM methods, one can perform pose graph optimization in order to compute a trajectory that is globally consistent with all previously estimated pairwise image alignments [561].

precision since it relies on a statistically optimal estimate of structure and motion given all the sensory brightness data.

In realtime-capable SLAM methods, global consistency can be achieved in several steps: Firstly, one can jointly optimize over the last k keyframes as done in DSO to assure consistency over a sliding temporal window (sliding window photometric bundle adjustment). Secondly, one can additionally run a Pose Graph Optimization (PGO) [561, 373] – see Figure 7.9 – in order to recompute a camera trajectory that is maximally consistent with all estimated pairwise image alignments. And thirdly, one can perform an adaptive version of pose graph optimization called Pose Graph Bundle Adjustment (PGBA) [1139] which additionally incorporates the full photometric uncertainty of bundle adjustment with the same computational efficiency (because only the camera poses are being updated).

7.3.7 Solving BA

Although BA could be solved using general variable elimination techniques as discussed in Chapter 2, there are specific sparsity solutions able to profit from its structure. Figure 7.10 shows a toy example with 4 cameras and 9 points observed from them. The observation Jacobian is very sparse, as each observation $\boldsymbol{z}_{ij}$ depends only on the camera i and the point j. As a result, each observation introduces in the Hessian a diagonal block for the camera, a diagonal block for the point, and an

off-diagonal camera-point block. As the number of points is typically several orders of magnitude larger than the number of cameras, a good solution is to eliminate first the points, and then solve for the cameras.

This can be done using the Schur complement. If we have a linear system where D is invertible, we can transform it by multiplying both sides on the left by a matrix:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix},$$
$$\begin{pmatrix} \mathbf{I} & -\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix},$$
$$\begin{pmatrix} \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C} & \mathbf{0} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 - \mathbf{B}\mathbf{D}^{-1}\mathbf{b}_2 \\ \mathbf{b}_2 \end{pmatrix},$$

to get a system where we can solve first $\boldsymbol{x}_1$ and then $\boldsymbol{x}_2$:

$$\begin{aligned} \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)\mathbf{x}_1 &= \mathbf{b}_1 - \mathbf{B}\mathbf{D}^{-1}\mathbf{b}_2, \\ \mathbf{D}\,\mathbf{x}_2 &= \mathbf{b}_2 - \mathbf{C}\mathbf{x}_1. \end{aligned}$$

The BA problem needs to solve in each iteration an equation of the form:

$$\begin{pmatrix} \mathbf{H}_{cc} & \mathbf{H}_{cp} \\ \mathbf{H}_{cp}^{\top} & \mathbf{H}_{pp} \end{pmatrix} \begin{pmatrix} \mathbf{d}_c \\ \mathbf{d}_p \end{pmatrix} = \begin{pmatrix} \mathbf{b}_c \\ \mathbf{b}_p \end{pmatrix}.$$

This can be solved in tree steps: computing the Schur complement of the points to obtain the reduced camera system, solving it for the cameras, and finally solving for the points:

$$\begin{aligned} \mathbf{H}_{cc}^{red} &= \mathbf{H}_{cc} - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{cp}^{\top}, \\ \mathbf{H}_{cc}^{red}\,\mathbf{d}_c &= \mathbf{b}_c - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{b}_p, \\ \mathbf{H}_{pp}\,\mathbf{d}_p &= \mathbf{b}_p - \mathbf{H}_{cp}^{\top}\mathbf{d}_c. \end{aligned} \tag{7.7}$$

As $\mathbf{H}_{pp}$ is block diagonal, the Schur complement and the final point solution can be done very efficiently point by point. As shown in Figure 7.10 the reduced camera system is less sparse, as it contains blocks that relate pairs of cameras that have seen some point in common. In the example, there is a block between cameras 1 and 2, but not between cameras 1 and 3. In local BA the reduced camera system will be almost full and can use dense matrix solvers. In contrast, full BA has a much larger number of keyframes but there is less covisibility between them, so it can profit from a sparse solver for the reduced camera system.

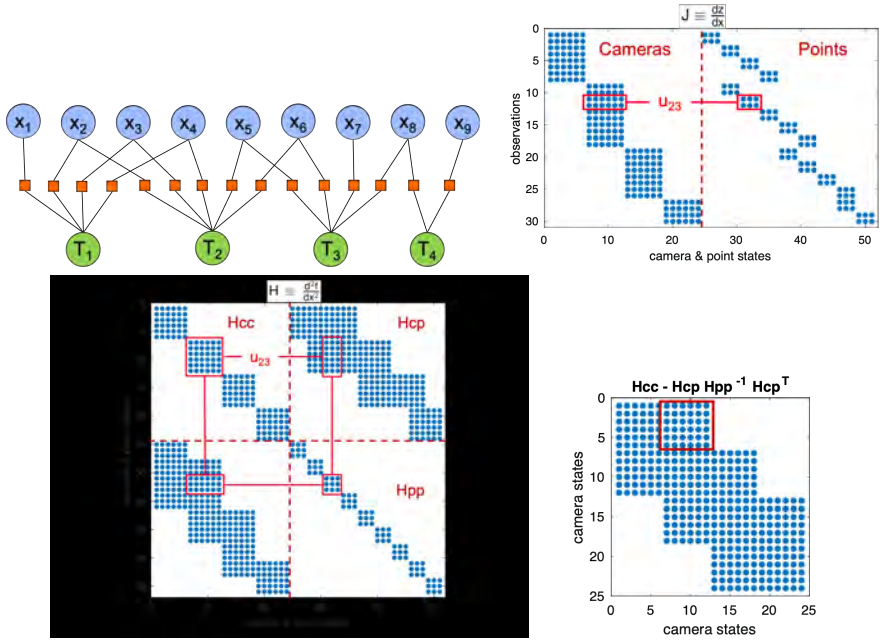

Figure 7.10 Toy example with 4 cameras and 9 points showing in the first row the factor graph and the Jacobian of point observations. The second row shows complete Hessian, and the Hessian of the reduced camera system.

7.3.8 Examples of Full Visual SLAM Systems

LSD-SLAM (Large-Scale Direct SLAM) [315] is a direct SLAM system that focuses on dense tracking and semi-dense mapping. It relies on photometric error minimization rather than keypoint-based methods, making it particularly effective in low-texture environments. The system operates in real-time, providing a semi-dense reconstruction of the scene and is well-suited for monocular cameras in indoor and small-scale outdoor environments.

ORB-SLAM [793, 791] is one of the most widely adopted SLAM systems due to its robustness and flexibility. It integrates keypoint-based tracking using ORB (Oriented FAST and Rotated BRIEF) descriptors, effective loop closure detection, and sparse map representation. ORB-SLAM supports monocular, stereo, and RGB-D cameras, making it highly versatile across different setups. It excels in scenarios requiring high accuracy and robust relocalization capabilities. In ORB-SLAM3 [148] it was extended to fisheye cameras, multimap, and visual-inertial SLAM. While originally conceived as a visual-inertial odometry system, OKVIS [652], the newest version, OKVIS2 [651], a visual-inertial SLAM system, may also be run in vision

only (multi-camera) mode. Similar to the various ORB-SLAM versions, it uses keypoints and descriptors (BRISK).

Direct Sparse Odometry (DSO) [316] is a direct method for estimating 3D point cloud and camera trajectory. In contrast to LSD SLAM camera motion and landmark points are estimated jointly in a single Gauss-Newton optimization. In order to achieve real-time performance, only the last k key frames are being updated resulting in a sliding-window photometric bundle adjustment. The number k of considered keyframes provides a trade-off between speed and accuracy. Moreover, DSO makes use of a full photometric calibration with camera response function and vignette. The traditional brightness-constancy assumption is thus replaced by an irradiance-constancy assumption, i.e. the assumption that respective points in the 3D world emit the same irradiance over time. Extensions of this approach to stereo systems [1158] and omni-directional cameras [746] have been proposed. Loop closure detection has also been added to reduce drift in longer sequences [373].

While among real-time capable approaches direct methods like DSO were shown to provide more accuracy and robustness than keypoint-based methods [316], for optimal performance they typically require a good photometric calibration and a global shutter camera. The rolling shutter effect leads to geometric distortions. While these can be modeled in direct SLAM methods [985, 986], the resulting approaches are often no longer real-time capable. As a result they are better handled in keypoint-based approaches which by design minimize geometric distortions. Nevertheless, direct methods often do better on low-resolution videos where due to blurring and down-sampling it may be harder to identify reliable feature points [1222]. Furthermore, feature points are valuable for efficient re-localization and loop closuring. As a consequence, practical visual SLAM systems will often revert to hybrid approaches that combine the best of both worlds. An example of a hybrid approach is [393] where feature-based re-localization information is tightly integrated in a direct visual SLAM approach to further boost its robustness and precision.

7.4 Realtime Dense Reconstruction

While traditionally bundle adjustment and SLAM tackle the problem of reconstructing camera motion and a sparse set of landmark points for numerous applications ranging from augmented reality to autonomous robots, one would prefer having a dense reconstruction of the observed world. To this end, a number of algorithms for realtime dense reconstruction from a monocular camera have been advocated over the years [1051, 806, 1177, 882]. Traditionally they revert to variational methods to estimate a continuous 3D structure by minimizing a loss function [1051]:

$$\min_{h:\Omega\to\mathbb{R}} \frac{1}{2} \sum_{i\in\mathcal{I}(x)} \int_{\Omega} \rho_i(x,h)dx \; + \; \lambda \int |\nabla h| \, d^2x, \tag{7.8}$$

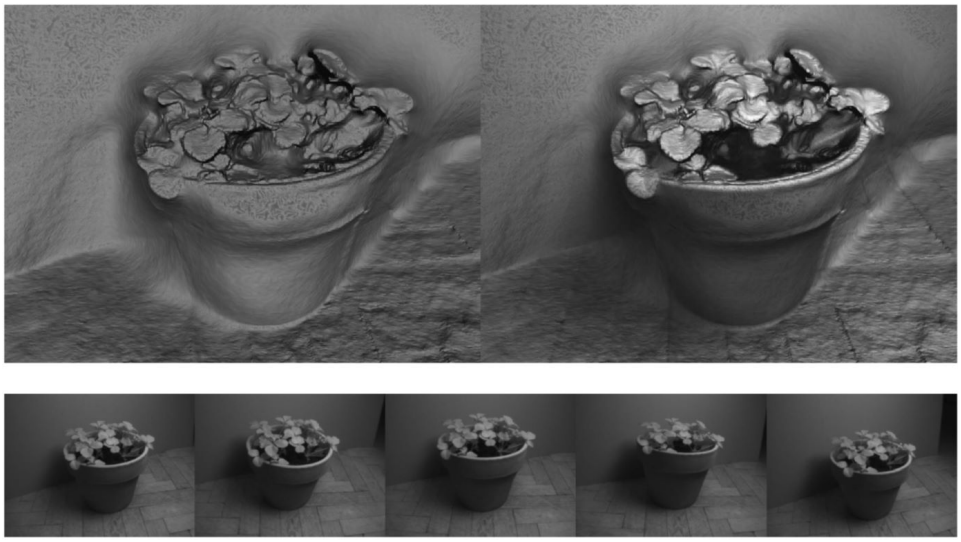

Figure 7.11 Dense reconstructions of the 3D world (above) can be computed in realtime from a handheld monocular camera (below) using variational methods that are efficiently parallelized on a GPU [1051]. Related approaches were proposed in [806, 1177, 882].

with respect to a dense map h that assigns a depth value to every pixel x in the image plane $\Omega \subset \mathbb{R}^2$. The residual

$$\rho_i(x,h) = \Big| I_i\left(\pi T_w^i X(x,h)\right) - I_0(x) \Big|, \tag{7.9}$$

enforces consistency of the brightness at pixel x in the reference image I_0 to the corresponding pixels in a set of adjacent images I_i. The corresponding pixel is obtained by taking the $3D$ point $X(x,h)$, performing the transformation $T_w^i \in SE(3)$ to camera i, followed by a perspective projection π into the image I_i. Here $\mathcal{I}(x)$ denotes the index set of the images for which the projected point lies inside the image plane. The total variation regularizer (weighted by λ) enforces spatial smoothness of the computed depth map and induces a soap-film-like fill-in for unobserved areas as shown in Figure 7.11.

7.5 SLAM with Depth-sensing Cameras

With the introduction of Microsoft Kinect, depth-sensing cameras became a commodity. These so-called RGB-D cameras are typically either structured-light or time-of-flight-based and provide a stream of depth images, often in conjunction with a color image stream. As such, they are conceptually between standard cameras and LiDAR sensors. Yet, in contrast to LiDAR sensors they provide an instant 2D array of depth values. Equipped with suitable algorithms, RGB-D cameras are

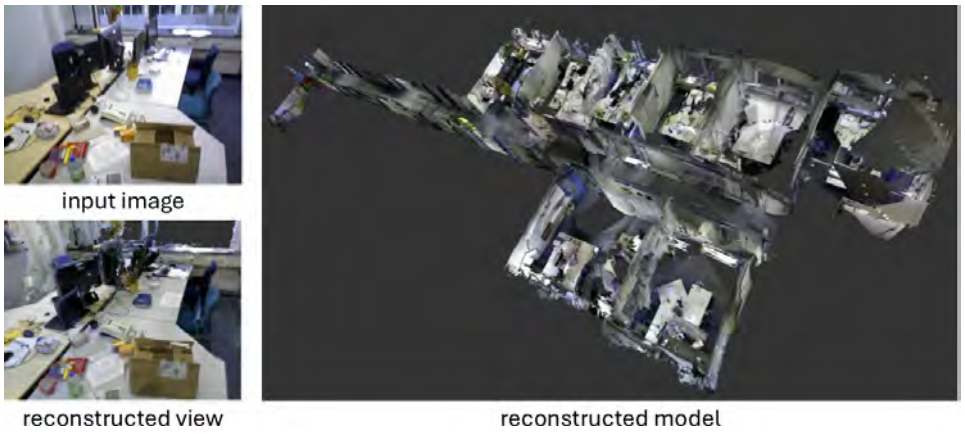

Figure 7.12 Dense reconstructions of a large-scale corridor scene with multiple offices computed from a moving RGB-D camera [1036]. Using octrees [1036] or voxel hashing [814] and direct camera tracking [562], such reconstructions were demonstrated to run in real-time on a GPU [1036] and even on a tablet CPU [1037]

very powerful for 3D sensing, albeit being limited to indoor applications (because the infrared-based sensing clashes with sunlight) and a certain range (typically up to around 5 meters).

Kinect Fusion [807] built upon previous work for range image fusion [242] to advocate a method for reconstructing camera motion and 3D structure from a moving RGB-D camera. The basic idea for fusing the various depth images into a coherent 3D reconstruction is to encode each depth image as a (projective) signed distance function $d_i(x)$ which for every voxel $x \in V$ encodes the (signed) distance to the nearest surface point (along the viewing ray). Subsequently one can compute an aggregated distance function $D(x)$ for all voxels as a weighted average of the individual distance functions:

$$D(x) = \frac{\sum_i \omega_i(x) d_i(x)}{\sum_i \omega_i(x)}. \tag{7.10}$$

Under the assumption of Gaussian noise in the depth direction, this weighted average is nothing but the maximum-likelihood estimate of the distance function. For more robustness, one typically averages *truncated* signed distance functions so that each sensed surface point merely has a local impact on the reconstruction. The weights $\omega_i(x)$ encode the certainty of the respective surface measurements (that is sensor dependent and typically decays with distance from the object). In order to fill holes in the reconstruction, one can revert to post-processing techniques [807], or modify the above weighting scheme to ensure watertight-ness of the reconstructions [1049].

While earlier RGB-D SLAM approaches track the camera by means of aligning

respective depth point clouds with the ICP algorithm [85], subsequent works advocated a direct minimization of residuals that measure color and depth consistency of two consecutive RGB-D frames (I_1, d_1) and (I_2, d_2) [562]:

$$r_I(\xi) = I_2(\tau_g(x)) - I_1(x), \quad r_d = d_2(\tau_g(x)) - \left[g\pi^{-1}(x, d_1(x))\right]_z, \tag{7.11}$$

where π denotes the perspective projection, $g \in \mathrm{SE}(3)$ denotes the desired rigid body motion, $\tau_g(x) = \pi g \pi^{-1}(x, d_1(x))$ denotes the induced warping between corresponding pixels, and $[\cdot]_z$ returns the z-component of a point. One can then fit the distribution of all residuals $r = (r_I, r_d)$ with a suitable distribution and determine a maximum a posteriori estimate of the camera transformation $g = \exp(\xi)$ by means of a coarse-to-fine Gauss-Newton optimization in $\xi \in \mathrm{se}(3)$ [1035, 561]. Compared to the classical ICP approach, this direct tracking approach reduces the root mean square tracking error by nearly an order of magnitude on established benchmarks while being significantly faster – see [562] for details.

For mapping at larger scale, the uniform voxel representation is too memory-intense. Therefore one typically reverts to adaptive representations using voxel hashing [814] or octrees [1036] – see Figure 7.12.

Later approaches have furthermore demonstrated scalability to larger scenes by employing a moving fusion window as in kintinuous [1180], or by including loop-closure as ElasticFusion [1181] that deforms the entire dense map representation through a deformation graph.

7.6 Combining Vision with Other Modalities

For numerous reasons, it is advisable to combine vision with other modalities. This can provide additional robustness and precision, but it can also provide absolute scale information. Specifically, inertial sensors provide metric scale and highly reliable and robust local, relative, motion measurements. GPS/WiFi, however, may be leveraged for global (re-)localization and geo-positioning. These complementary inputs address limitations inherent to vision-only systems, making them indispensable in practical applications.

7.6.1 Inertial Measurement Units (IMU)

IMUs provide high-frequency measurements of angular velocity and linear acceleration, enabling precise estimation of local motion. A simple way to fuse sensory information is a loosely coupled approach where the sensory information from each sensor is independently aggregated into a pose estimate and subsequently, respective estimates are fused with a Kalman filter. While this often works fairly well in practice, for example for autonomous navigation of quadrotors [314] or nano-copters [299], it is less precise than a tighly-coupled integration of sensory information.

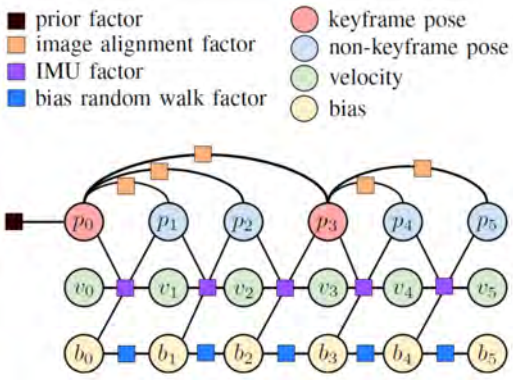

Figure 7.13 The integration of multiple sensors can be elegantly performed in tightly coupled manner using factor graphs. This is an example factor graph for tightly coupled fusion of vision and IMU [1120]. It significantly increases precision and robustness of camera motion and 3D structure – see Figure 7.14.

Figure 7.14 Tight fusion of the IMU measurements with direct image alignment results in more accurate position tracking (left) compared to the purely visual odometry system that only relies on image alignment (right). The fusion was achieved with the factor graph shown in Figure 7.13. The reconstructed pointclouds come from pure odometry, no loop closures were enforced [1120].

First tightly-coupled approaches leverage an EKF scheme, whereby the IMU kinematics are integrated in the prediction step, and visual keypoint measurements serve as upadates–as in the seminal work of the MSCKF [783]. Alternatively, sliding-window and batch optimizers may be formulated adopting the factor graph approach discussed in detail in part 1 of this book. Figure 7.13 shows a factor graph proposed for stereo-inertial odometry [1120] that combines stereo LSD SLAM with IMU information. Figure 7.14 shows how the tightly coupled IMU information reduces the drift leading to more crisp and precise reconstructions. This tight coupling by factor graphs is also employed by state-of-the-art keypoint-based approaches, such as ORB-SLAM3 [148], OKVIS2 [651], and many more.

In practice, some of the most accurate visual-inertial odometry systems start with inertial integration as basis, and use vision primarily to correct for the fast-

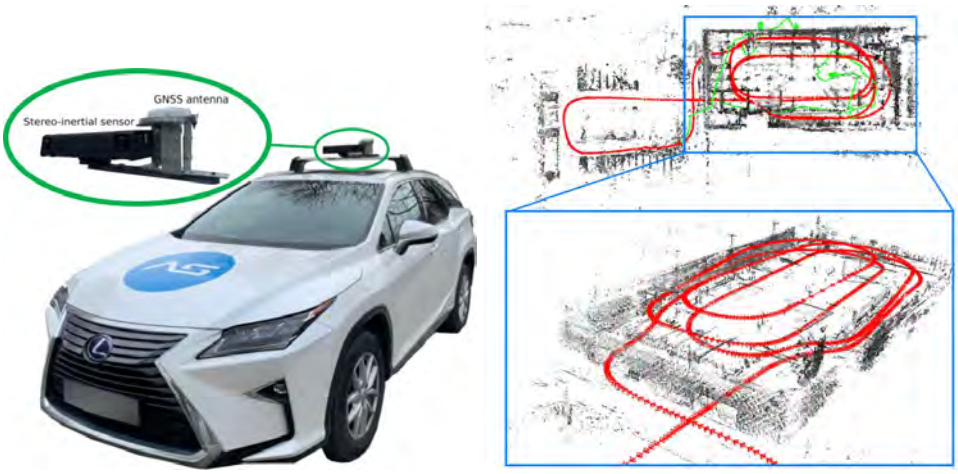

Figure 7.15 By combining multiple sensors including stereo cameras, IMU and RTK-GPS information (left), fused in a tighly coupled manner using factor graphs, one can obtain highly accurate and robust trajectories and pointclouds, both indoor and outdoor as shown in the reconstruction of a car driving on multiple levels of a parking garage (right) [1178].

growing drift owing to (doubly) integrated noise and biases over time. In addition, IMUs allow to observe metric scale of motion, as well as the sensor's orientation with respect to gravity–thus reducing the gauge freedom of the system to only 4 unknowns (global x,y,z position, as well as yaw)–compared to 7 unknowns (global x,y,z, roll, pitch, yaw, as well as scale) for vision-only systems. In many ways, IMUs are complementary to vision as a modality, and thus are ideal to combine in visual-inertial SLAM or Odometry systems.

A multitude of visual-inertial odometry/SLAM methods have been proposed over the years, often as extensions of existing visual SLAM systems. Popular approaches include a visual-inertial version of ORB SLAM [792], Direct Sparse Visual-Inertial Odometry [1140], VINS-Mono [903], BASALT [1122] and DM-VIO [1139]. The latter approach is a mono-inertial formulation that makes use of the concept of *delayed marginalization* to better capture the observability of motion in the respective sensors.

7.6.2 GPS and WiFi for Global Localization

While IMUs improve local motion estimation, GPS and WiFi are essential for global localization, especially in large-scale environments. In outdoor environments, GPS provides absolute position data, enabling the system to anchor the SLAM map to global coordinates. This is critical for outdoor applications like autonomous vehicles [1178, 200]–see Figure 7.15. In indoor scenarios, WiFi signals enable coarse

localization where GPS signals are unavailable, complementing visual map-based localization.

7.7 Bundle Adjustment Revisited

At the historical origin and at the heart of visual SLAM is the classical problem of bundle adjustment, namely, the joint estimation of all camera positions and all landmark positions. It has been studied for over a century and the classical approach detailed in Section 7.3.7 has been established and shown to work well in a multitude of seminal papers [1111, 21, 983]. Yet, this pipeline has two important shortcomings: Firstly, respective solutions require a suitable initialization of landmarks and camera poses. And secondly, for large-scale problems the computational and memory requirements can grow prohibitively large. In recent years, there have been a series of papers that address these shortcomings and challenge the traditional computational pipeline [477, 478, 266, 267, 1168, 1169, 1170].

The key computational bottleneck is the solution to the reduced camera system (7.7). Instead of solving this by means of an iterative conjugate gradient algorithm, Power Bundle Adjustment [1169] approximates the inverse of the Schur matrix

$$\mathbf{H}_{cc}^{red} = \mathbf{H}_{cc} - \mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{cp}^{\top}, \tag{7.12}$$

by means of a matrix power series [1169]:

$$(\mathbf{H}_{cc}^{red})^{-1} \approx \sum_{i=0}^{m} (\mathbf{H}_{cc}^{-1}\mathbf{H}_{cp}\mathbf{H}_{pp}^{-1}\mathbf{H}_{cp}^{\top})^{i}\, \mathbf{H}_{cc}^{-1}. \tag{7.13}$$

This power series provably converges to the true inverse for increasing cut off parameter m [1169]. The main advantage is that tedious matrix inversion is replaced by simple matrix multiplications, which can be done much faster and more memory-efficiently.

The dependency on initialization is alleviated in [477, 478] by reverting to the concept of variable projection–see Figure 7.16. To this end, the bundle adjustment problem is split into two stages. In the first stage, the complicated perspective projection is replaced by a generic projective matrix such that the resulting optimization can be solved analytically for the landmarks as a function of the camera parameters. This removes the chicken-and-egg dependency between landmarks and camera poses, leading to a larger basin of attraction when optimizing the camera poses. In the second stage of projective refinement, one uses the computed solution as an initialization for the original (perspective) reconstruction. Although this strategy is computationally too demanding for more than 100 cameras, its combination with the power series approach as proposed in [1170] offers a scalable solution for large-scale bundle adjustment problems without initialization.

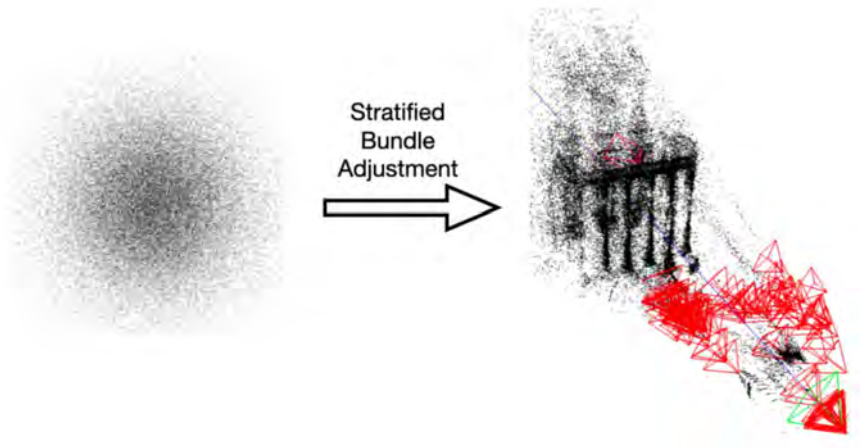

Figure 7.16 In a series of papers [477, 478, 1169, 1170], researchers advocate the use of variable projection methods and matrix power series in order to solve large scale bundle adjustment problems without initialization in a runtime- and memory efficient way. As shown above, camera poses and landmarks can thus be computed starting from a random initialization.

7.8 Recent Developments

Visual SLAM is an extremely active and dynamic field of research. While we tried to provide a comprehensive overview of classical visual SLAM methods, we invariably only covered a fraction of relevant work in this chapter, and we hope the reader may look into some of the many cited works for a more in-depth coverage. Moreover, one should emphasize that there have been many exciting developments in the recent past. Among other developments, there are generalizations of visual SLAM to dynamic environments with moving and potentially deformable objects. In addition, learning-based approaches to visual SLAM are becoming increasingly popular: In a multitude of publications, more and more components of the classical visual SLAM pipeline (feature extraction, correspondence estimation, image alignment, camera tracking, bundle adjustment, dense reconstruction, etc.) are being enhanced or replaced by learning-based formulations. All these ideas will be discussed in more detail in Part 3 of this book.