Review on the topic of

Simultaneous localization and mapping (SLAM)

Course: Robotics

Advisor:
prof. Miloš Jovanović

Student:
Vanja Kovinić

Belgrade, 2025.

Contents

1 Introduction to SLAM **1**

1.1 Definition and Problem Statement 1

1.2 Historical Context and Importance in Robotics 3

2 Fundamental Challenges in SLAM **5**

2.1 The Chicken-and-Egg Problem 5

2.2 Sensor Limitations and Environmental Factors 7

2.3 Loop Closure and Data Association 8

3 Classical Approaches to SLAM **11**

3.1 Filter-Based Methods 11

3.2 Graph-Based Optimization 15

3.3 Traditional Visual SLAM Systems 18

4 Deep Learning in Modern SLAM **19**

4.1 Learned Features vs. Handcrafted Features 19

4.2 End-to-End SLAM Architectures 21

4.3 Neural Implicit Representations for Mapping 23

4.4 Depth Estimation Networks 26

1 Introduction to SLAM

1.1 Definition and Problem Statement

Simultaneous Localization and Mapping (SLAM) represents one of the fundamental challenges in robotics. At its core, SLAM addresses a deceptively simple question:
How can a mobile robot build a map of an unknown environment while simultaneously determining its own position within that map?
This seemingly straightforward task embodies a classic *"chicken-and-egg"* problem - to create an accurate map, the robot needs to know its precise location, but to determine its location, it needs an accurate map.
The SLAM problem can be formally defined as the process of constructing or updating a map of an unknown environment while keeping track of an agent's location within it. Mathematically, SLAM estimates both the trajectory of the robot ($X_1 = x_1, x_2, \ldots, x_t$) and the map of the environment (M) given the observations (Z_1) and control inputs (U_1):

$$p(X_1, M \mid Z_1, U_1)$$

This joint probability distribution encapsulates the inherent uncertainty in both the robot's position and the environmental map, recognizing that both must be estimated simultaneously.

The technical challenges of SLAM extend beyond this core problem. Real-world implementations must contend with:

1. **Sensor limitations**: All sensors provide imperfect measurements with noise, limited range, and potential occlusions.
2. **Data association**: Determining whether sensor observations correspond to previously observed landmarks or represent new features.
3. **Loop closure**: Recognizing when the robot has returned to a previously visited location, allowing for correction of accumulated errors.
4. **Computational efficiency**: Balancing accuracy with real-time performance requirements, especially on platforms with limited computing resources.
5. **Environmental dynamics**: Managing changes in the environment, such as moving objects or changing conditions.

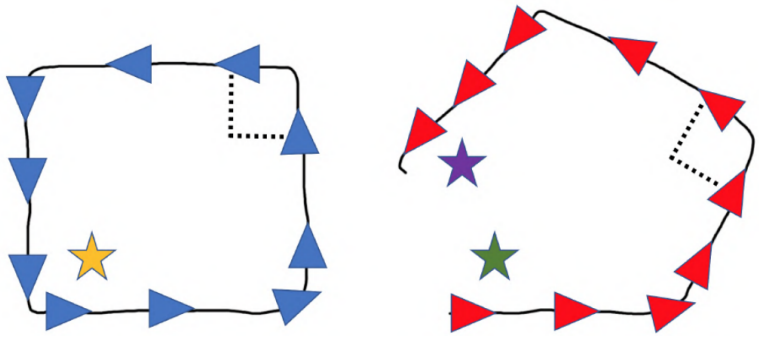

Figure 1: The diagram shows loop closure in robot mapping. When a feature is recognized again (left), a constraint is added to correct the robot's trajectory. If the feature is not recognized (right), errors in the estimated path remain uncorrected. [1]

A successful SLAM system must integrate sensor data over time while accounting for uncertainties and constraints to produce a consistent representation of both the robot's path and the surrounding environment.

1.2 Historical Context and Importance in Robotics

The development of SLAM techniques represents a crucial milestone in the evolution of autonomous robotics. Before SLAM, robots relied on pre-built maps or simple reactive behaviors, severely limiting their autonomy and usefulness in unknown environments. The conceptual foundations of SLAM emerged in the 1980s through seminal work by researchers such as Hugh Durrant-Whyte and John J. Leonard [2], who recognized to the statistical correlations between landmarks in a map. However, the term "SLAM" itself was not coined until the early 1990s.

The historical progression of SLAM research can be divided into several key phases:

1. **Early probabilistic approaches (1990s)**: The initial formulation of SLAM using Extended Kalman Filters (EKF) provided the mathematical framework for handling uncertainty in both robot motion and sensing [3]. These approaches were limited by computational complexity that scaled quadratically with the number of landmarks.

2. **Particle filter methods (early 2000s)**: FastSLAM [4] and other particle filter approaches offered improved performance for nonlinear systems and non-Gaussian noise, enabling more robust implementations.

3. **Graph-based optimization (mid-2000s)**: The formulation of SLAM as a sparse graph optimization problem led to more efficient solutions that could handle larger environments and longer trajectories.

4. **Visual SLAM systems (2007-2015)**: The incorporation of camera data enabled systems like MonoSLAM [5], PTAM, and ORB-SLAM [6], which could operate using only visual information without specialized ranging sensors.

5. **Learning-based approaches (2015-present)**: The integration of deep learning methods has pushed the boundaries of SLAM performance, particularly in feature extraction, depth estimation, and semantic understanding.

The introduction of SLAM algorithms effectively gave robots a spatial intelligence - the ability to interpret, map, and navigate previously unseen environments without human guidance. Some of the most notable applications of SLAM include:

- **Autonomous vehicles**: Tesla's FSD system and Waymo's autonomous taxis implement sophisticated SLAM variants that fuse visual data with other sensor modalities. Their ability to navigate complex urban environments relies not just on GPS (which fails in tunnels or urban canyons) but on real-time environmental mapping at centimeter-level precision.
- **Disaster response**: After the Fukushima nuclear disaster, robots equipped with RGBD-SLAM systems entered radiation-contaminated zones inaccessible to humans, creating 3D maps that enabled remote assessment of structural damage without endangering human lives [7].
- **Domestic robotics**: The evolution of cleaning robots from random bouncers like early Roombas to the systematic path-planners seen in Roborock S7 models illustrates how SLAM transformed consumer robotics from novelties into genuinely useful tools that can efficiently cover complex home layouts.
- **Mixed reality**: Apple's ARKit and Microsoft's HoloLens use visual-inertial SLAM to maintain stable positioning of virtual objects in the physical world, enabling applications from surgical training to architectural visualization that rely on precise spatial registration.

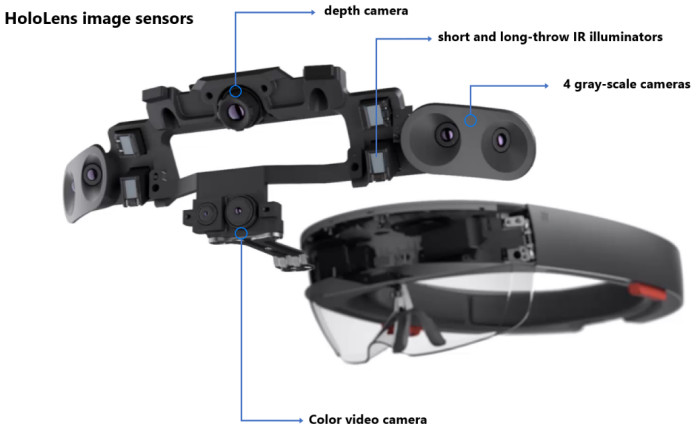

Figure 2: Microsoft HoloLens uses SLAM by fusing data from cameras, depth sensors, and IMUs to simultaneously track the headset's position and map the environment. This enables accurate placement of virtual objects in the real world for stable and immersive augmented reality experiences.

2 Fundamental Challenges in SLAM

2.1 The Chicken-and-Egg Problem

The core paradox of SLAM is deceptively simple: to build an accurate map, a robot needs to know its precise location, but to determine its location accurately, it needs a reliable map. This interdependence creates the *"chicken-and-egg problem"* of SLAM.

Consider a robot entering an unknown room. To map the room accurately, it must know exactly how far it has moved since entering. However, small errors in wheel encoders, IMU drift, or visual tracking cause position uncertainty to grow.

Without external references, these errors accumulate - a phenomenon known as "dead reckoning drift". After moving just a few meters, the robot's position estimate might be off by several centimeters, causing all mapped features to be misplaced. This circular dependency manifests in two critical ways:

First, position estimation errors directly propagate into the map. Imagine a robot that turns a corner and incorrectly estimates its rotation by 2 degrees. This small angular error will cause all subsequent wall measurements to be tilted, creating a distorted map. When the robot later tries to localize using this faulty map, its position estimates become increasingly unreliable.

Second, without correction mechanisms, uncertainty grows unbounded over time. A robot navigating a long corridor might start with centimeter-level position accuracy, but after traveling 20 meters, drift might increase to decimeter-level errors. This uncertainty expansion makes long-term SLAM inherently unstable without additional constraints.

The probabilistic SLAM framework addresses this by maintaining a joint distribution over both the robot's trajectory and map features:

$$p(x_t, m|z_1, u_1, x_0)$$

Where x_t is the robot's pose at time t, m is the map, z_1 are sensor observations, u_1 are control inputs, and x_0 is the initial pose. This formulation acknowledges that robot positions and landmark locations are correlated random variables that must be estimated together, not independently. The correlation is intuitive: if we learn that a landmark is farther to the right than previously estimated, the robot was likely more to the left than we thought. Classic SLAM approaches handle this correlation through:

- Maintaining the full covariance matrix between pose and map variables in EKF-SLAM [8]
- Sampling multiple potential trajectories in particle filter approaches like FastSLAM [4]
- Building a graph of spatial constraints that can be jointly optimized in modern graph-based SLAM

While the chicken-and-egg problem cannot be eliminated completely, these mathematical frameworks manage it by simultaneously optimizing for consistent mapping and localization, effectively distributing errors throughout the system rather than allowing them to accumulate unconstrained.

2.2 Sensor Limitations and Environmental Factors

Every SLAM system must contend with the limitations of its perception hardware and challenging environmental conditions:

LiDAR Sensors struggle with reflective or transparent surfaces, have limited range, and suffer from motion distortion during scanning.

RGB Cameras face scale ambiguity (in monocular setups), illumination sensitivity, difficulty with texture-less regions, and degraded performance from motion blur.

Depth Cameras have shorter range than LiDAR, perform poorly in bright sunlight, and produce noisy measurements at object edges.

Inertial Measurement Units accumulate drift through integration and suffer from changing bias and noise.

Beyond sensor limitations, environmental factors pose additional challenges:

Perceptual Aliasing: Repetitive environments (office corridors, parking structures) create ambiguity as different locations appear identical.

Featureless Environments: Spaces lacking distinctive features provide insufficient information for reliable localization.

Dynamic Objects: Moving elements like people or rearranged furniture violate the static world assumption in many SLAM formulations.

Temporal Changes: Outdoor environments undergo appearance changes with seasons, weather, and time of day.

Successful SLAM implementations address these challenges through sensor fusion, robust feature selection, outlier rejection, and specialized algorithms for dynamic environments.

2.3 Loop Closure and Data Association

Loop closure (recognizing when a robot has returned to a previously visited location) represents perhaps the most powerful constraint in SLAM systems. Without loop closure, errors accumulate unbounded over time, resulting in inconsistent maps and increasingly inaccurate localization. When properly detected and incorporated, loop closures can dramatically improve map consistency by distributing accumulated errors throughout the trajectory.

To understand the significance of loop closure, consider a robot exploring an office building. Starting from the lobby, it navigates through corridors, accumulating small errors in its position estimate. After circling back to the lobby 15 minutes later, its estimated position might be off by several meters from its actual position. Without recognizing the lobby, the robot would produce a map with disconnected segments and potentially overlapping walls. Upon detecting that it has returned to the lobby, the system can "close the loop," triggering a global optimization that realigns the entire trajectory and map.

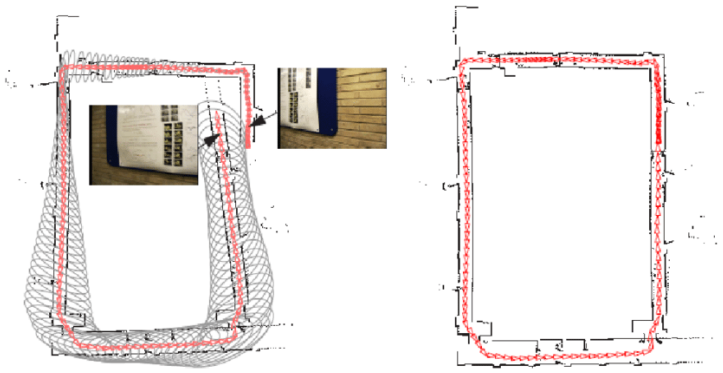

Figure 3: Left image: Robot's path before loop closure, showing growing uncertainty (gray ellipses) and misalignment at the loop point. Camera views used for matching are inset. Right image: Corrected map after loop closure, with aligned positions and reduced errors. [9]

Loop closure detection faces several significant challenges:

Viewpoint variation: The robot may approach the same location from a completely different direction. A corridor observed initially from north to south looks substantially different when viewed from south to north.

Perceptual aliasing: Many environments contain similar-looking areas. In a hospital, multiple identical-looking corridors can trigger false loop closures that catastrophically distort the map.

Accumulated pose error: By the time the robot returns to a known location, its position uncertainty may be so large that the expected and actual observations have minimal overlap.

Appearance changes: Lighting conditions, moved objects, seasonal variations, and human activities can dramatically alter how the same location appears at different times.

Modern loop closure approaches implement multi-stage processing:

- **Place recognition**: Identifying candidate locations, often using appearance-based methods such as visual bag-of-words models (DBoW), sequence-to-sequence matching, or learned embedding spaces that map similar locations close together regardless of viewpoint or illumination.
- **Geometric verification**: Confirming candidates by establishing geometric consistency through feature matching and transformation estimation.
- **Global consistency checking**: Validating that the proposed loop closure is consistent with the existing map and trajectory, sometimes using techniques like RRR (Realizing, Reversing, Recovering) to reject outliers.
- **Graph optimization**: Incorporating verified loop closures as constraints in a pose graph that is globally optimized to distribute errors throughout the trajectory.

Loop closure is intimately connected with data association - the process of matching current observations with previously established map elements. While loop closure operates at the level of places, data association functions across multiple levels. It enables low-level feature matching between consecutive frames for visual odometry, facilitates landmark identification by associating observed features with mapped landmarks, and supports object recognition by identifying and tracking objects across multiple observations.

While structured, static environments are reasonably well-handled by current algorithms, dynamic, changing, or visually ambiguous environments continue to pose significant difficulties that drive ongoing research.

3 Classical Approaches to SLAM

3.1 Filter-Based Methods

Filter-based methods dominated early SLAM research, providing principled approaches to handling the inherent uncertainties in robot motion and sensing. These methods recursively estimate the posterior probability distribution over robot poses and map features as new observations arrive.

Extended Kalman Filter SLAM (EKF-SLAM) was the first widely adopted approach to SLAM. Introduced in the early 1990s, EKF-SLAM represents the joint state of the robot and map features with a multivariate Gaussian distribution, maintaining both mean estimates and their uncertainty through a covariance matrix.

The EKF-SLAM algorithm operates in two phases:

- **Prediction step**: Updates the state estimate based on robot motion, typically increasing uncertainty
- **Correction step**: Incorporates sensor measurements to update both the state estimate and reduce uncertainty

The mathematical formulation involves:

- State vector $\mathbf{x} = [\mathbf{x}_r, \mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n]^T$ combining robot pose $\mathbf{x}_r$ and map features $\mathbf{m}_i$
- Covariance matrix $\mathbf{P}$ representing uncertainty in the state estimate and correlations between variables
- Nonlinear motion model $\mathbf{f}(\mathbf{x}, \mathbf{u})$ with control input $\mathbf{u}$
- Nonlinear observation model $\mathbf{h}(\mathbf{x})$ relating state to expected measurements

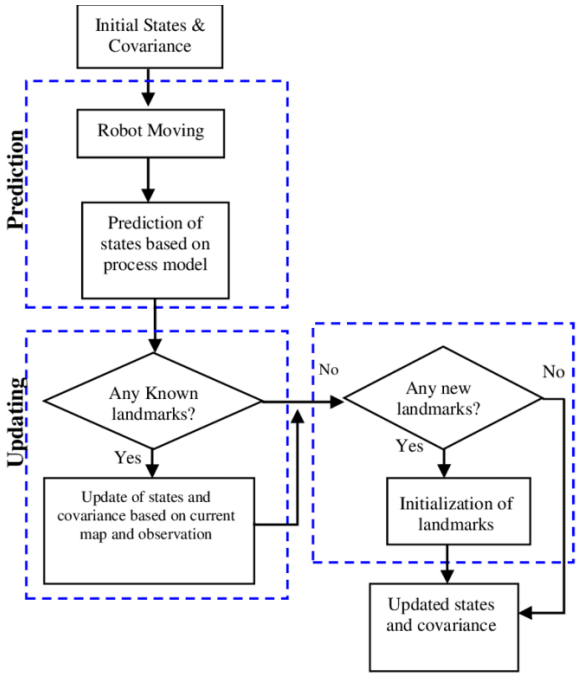

Figure 4: This flowchart shows the EKF-SLAM algorithm process. Starting with initial states and covariance, it follows a prediction phase when the robot moves, then branches into an update phase that either updates existing landmarks or initializes new ones, ultimately producing updated states and covariance. [10]

EKF-SLAM handles the chicken-and-egg problem by explicitly maintaining correlations between the robot pose and map features. When the robot observes a landmark, the uncertainty in both the robot's position and the landmark's position becomes correlated—improving the estimate of one improves the estimate of the other.

However, EKF-SLAM suffers from significant limitations:

- $O(n^2)$ computational complexity with n landmarks, making it unsuitable for large-scale environments
- Linearization errors from the extended Kalman filter that can cause inconsistency and divergence

- Difficulty handling multi-modal distributions and ambiguous data associations
- Sensitivity to outliers in sensor measurements

Particle Filter SLAM addresses some EKF-SLAM limitations by representing the posterior distribution with a set of weighted samples (particles) rather than a single Gaussian. The most influential particle filter approach, FastSLAM, introduced by Montemerlo et al. in 2002 [4], uses a Rao-Blackwellized particle filter that:

- Factors the SLAM posterior into robot trajectory and map features conditioned on trajectory
- Represents the robot trajectory distribution with particles
- Represents map feature estimates with compact EKFs, one per particle

This factorization leverages a key insight: conditioned on the robot trajectory, map features are independent. The FastSLAM algorithm:

- Maintains multiple trajectory hypotheses through particles
- Updates individual map features efficiently with separate EKFs
- Naturally handles multi-modal distributions and ambiguous data associations
- Resamples particles based on observation likelihood, removing unlikely trajectories

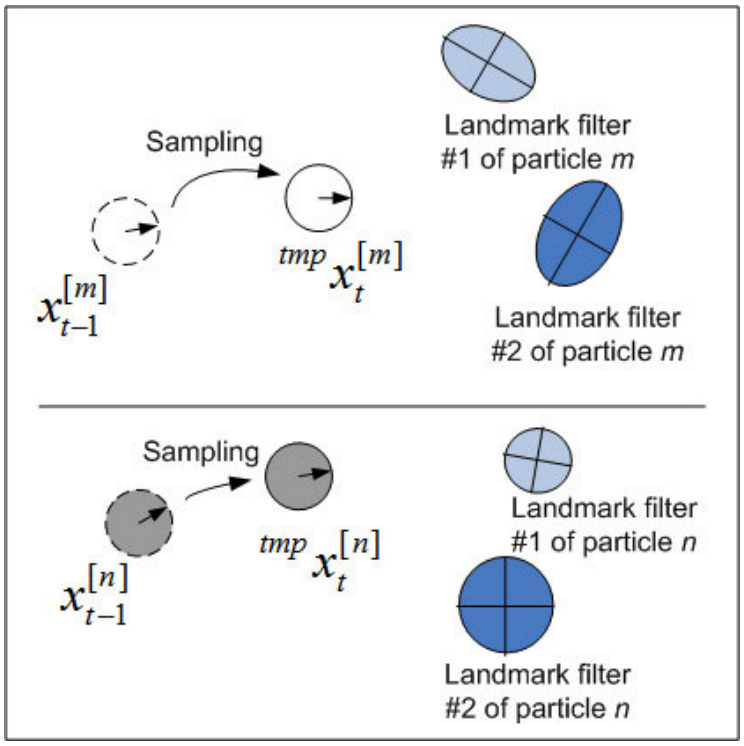

Figure 5: This diagram shows the FastSLAM particle filter process. Two particles (m and n) are sampled from previous to temporary states. Particles represent different possible robot pose hypotheses. Each particle maintains separate uncertainty estimates for two landmarks, shown as ellipses - larger ellipses indicate greater positional uncertainty. Light blue ellipses represent landmark filter #1, while darker blue ellipses represent landmark filter #2, with the uncertainty patterns varying between particles. [11]

FastSLAM achieves $O(M \log N)$ complexity for M particles and N landmarks, allowing it to scale better than EKF-SLAM to larger environments.

Despite these advantages, particle filter approaches have their own limitations:

- Particle depletion in high-dimensional spaces
- Difficulty representing long trajectories efficiently
- Computational constraints on the number of particles

- Need for resampling, which can cause loss of diversity

Filter-based methods laid the groundwork for understanding SLAM as a probabilistic estimation problem, but their limitations in handling large-scale environments eventually led to the development of more efficient optimization-based approaches.

3.2 Graph-Based Optimization

Graph-based optimization has become the dominant paradigm in modern SLAM systems, offering superior scalability and accuracy compared to filter-based methods. Rather than recursively updating a state estimate, graph-based SLAM formulates the problem as finding the configuration of robot poses and map features that best explains all observations.

The core idea is to represent SLAM as a sparse graph where:

- **Nodes** represent robot poses and map features
- **Edges** represent constraints from odometry measurements and landmark observations

This approach was pioneered by Lu and Milios in the 1990s but became practical only in the 2000s with the development of efficient optimization techniques. The graph-based formulation transforms SLAM into a nonlinear least squares problem, finding the configuration of nodes that minimizes the sum of squared errors introduced by the constraints.

Mathematically, we aim to find the state vector $\mathbf{x}$ that minimizes:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i,j} \rho(\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij})$$

Where:

- $\mathbf{e}_{ij}$ is the error between predicted and measured relative poses or observations
- $\mathbf{\Omega}_{ij}$ is the information matrix (inverse covariance) representing constraint certainty
- ρ is a robust cost function that reduces the influence of outliers

Key advantages of graph-based SLAM include:

- **Scalability**: Sparse graph structure enables efficient optimization even for large environments
- **Global consistency**: Optimizes all constraints simultaneously instead of sequential filtering
- **Loop closure integration**: Naturally incorporates loop closures as additional constraints
- **Incremental operation**: Can update the solution efficiently as new information arrives

Graph-based SLAM also integrates well with both traditional and learning-based front-ends that extract constraints from sensor data. The separation of front-end (data association and constraint extraction) from back-end (optimization) has enabled modular system design and focused innovation in each component.

While graph-based optimization overcomes many limitations of filter-based methods, challenges remain:

- **Initialization sensitivity**: Performance depends on good initial values for optimization
- **Outlier robustness**: False constraints can significantly distort the solution
- **Parameter tuning**: Requires careful selection of robust cost functions and optimization parameters

Nevertheless, graph-based optimization has proven remarkably successful, becoming the foundation for most state-of-the-art SLAM systems in the last decade.

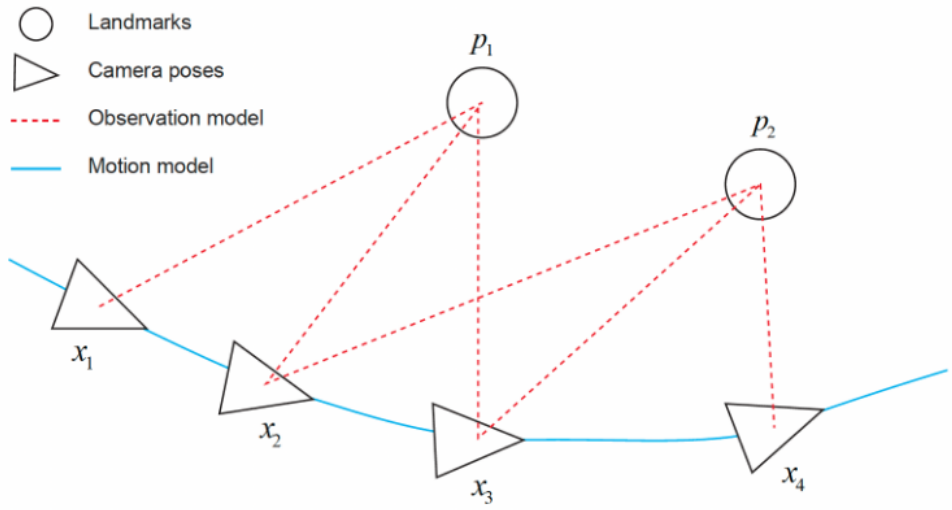

Figure 6: An example of the graph optimization. [12]

3.3 Traditional Visual SLAM Systems

Visual SLAM systems use cameras as their primary sensing modality, offering rich environmental information at relatively low cost. Three main approaches have emerged in traditional visual SLAM:

1. Feature-based methods detect and track distinctive points across frames, using them as landmarks. These systems typically operate through feature detection, description, matching, motion estimation, and bundle adjustment to optimize camera poses and feature positions. Key innovations include parallel tracking and mapping threads and keyframe-based representations.

2. Direct methods operate on image intensity values without extracting features, potentially using all pixels rather than sparse points. These approaches minimize photometric error rather than geometric reprojection error and can work in less textured environments where feature detection might fail.

3. Hybrid approaches combine elements of both feature-based and direct methods for increased robustness and efficiency. Traditional visual SLAM systems face several challenges: scale ambiguity in monocular setups, difficult initialization, sensitivity to dynamic objects and illumination changes, and poor performance in textureless regions. Many systems address these limitations by incorporating additional sensing modalities like stereo cameras, RGB-D sensors, or IMUs. Despite impressive performance in controlled environments, traditional visual SLAM approaches still struggle with perceptual aliasing, long-term operation in changing environments, and semantic understanding—limitations that have motivated the integration of learning-based methods.

4 Deep Learning in Modern SLAM

Deep learning has revolutionized numerous aspects of SLAM systems, addressing many limitations of traditional approaches while introducing new capabilities. Rather than completely replacing geometric methods, learning-based components typically enhance SLAM pipelines by leveraging data-driven insights where classical algorithms struggle.

4.1 Learned Features vs. Handcrafted Features

Traditional SLAM systems rely on handcrafted feature detectors and descriptors like SIFT [13], SURF [14], and ORB [6], which were designed based on human intuition about what makes points distinctive and matchable. While effective in many scenarios, these features have inherent limitations:

- Poor performance under challenging conditions (illumination changes, motion blur)
- Limited robustness to viewpoint variations
- Difficulty in textureless or repetitive environments
- Inability to adapt to specific domains without manual tuning

Deep learning has transformed feature extraction through learned alternatives that outperform traditional methods in robustness and accuracy. These approaches typically use convolutional neural networks (CNNs) trained on large datasets of image pairs with known correspondences.

Learned feature detectors identify keypoints that are:

- Repeatable across different views and conditions
- Distinctive enough for reliable matching
- Accurately localizable in the image

Key advantages of learned features include:

- **Adaptability**: Can be trained for specific environments or conditions
- **Context awareness**: Consider wider image context beyond local patches
- **Task optimization**: Can be trained specifically for SLAM performance metrics
- **Robustness**: Better handling of challenging conditions than handcrafted features

Many modern systems adopt a hybrid approach, using learned features for challenging situations while maintaining handcrafted features for computational efficiency in standard conditions. This combination leverages the strengths of both paradigms: the proven reliability of geometric methods and the adaptability of learning-based approaches.

4.2 End-to-End SLAM Architectures

End-to-end SLAM architectures represent the most ambitious application of deep learning to SLAM, aiming to replace traditional modular pipelines with neural networks that directly map sensor inputs to pose estimates and map representations. These systems attempt to learn the entire SLAM process through data rather than explicit geometric modeling.

Unlike traditional SLAM systems that consist of clearly defined components (feature extraction, matching, pose estimation, mapping, loop closure), end-to-end approaches blur these boundaries, allowing the network to discover its own internal representations and processing steps.

The spectrum of end-to-end SLAM includes:

Pose Regression Networks predict camera poses directly from images or image sequences. While conceptually simple, these approaches often struggle with generalization to new environments and accumulate drift over time, limiting their applicability to short trajectories.

Deep Visual Odometry systems use CNNs or recurrent neural networks (RNNs) to estimate frame-to-frame motion, sometimes incorporating geometric constraints. DeepVO [15] represents early example that demonstrated the potential of learned odometry while highlighting the challenge of preventing drift.

Learning-based SLAM Frameworks integrate deep networks while maintaining some classical components. MapNet [16] combines absolute pose regression with relative pose constraints, while CNN-SLAM [17] and CodeSLAM [18] use learned components within a traditional SLAM framework.

Memory-Augmented Networks address the mapping aspect by incorporating explicit memory mechanisms that store and retrieve spatial information. Neural SLAM architectures use differentiable memory structures that function similarly to occupancy grids or feature maps, allowing end-to-end training while maintaining interpretable map representations.

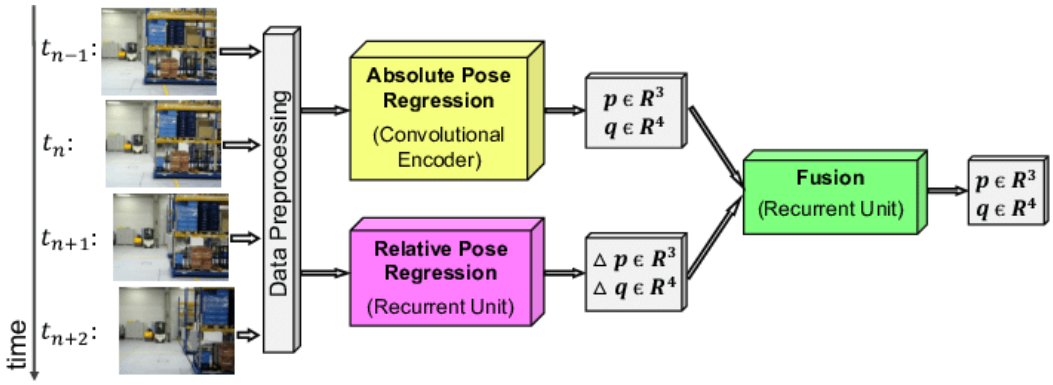

Figure 7: An example of pose regression neural network. [19]

Key challenges in end-to-end SLAM include:

- **Generalization**: Networks trained in specific environments often perform poorly in new settings
- **Data Requirements**: Acquiring sufficient training data with ground truth poses and maps is difficult
- **Interpretability**: Black-box models make debugging and performance guarantees challenging
- **Computational Efficiency**: Deep networks typically require significant computational resources
- **Consistency**: Ensuring global map consistency without explicit loop closure is difficult

Despite these challenges, end-to-end approaches offer intriguing benefits:

- **Adaptability**: Can potentially learn to handle conditions that challenge geometric methods
- **Integration of Semantic Information**: Naturally incorporate object recognition and scene understanding

- **Multimodal Fusion**: Can learn optimal ways to combine different sensor inputs
- **Uncertainty Representation**: Some architectures naturally represent uncertain estimates

Currently, end-to-end SLAM systems have not yet surpassed the best modular approaches in general settings. However, they show particular promise in specific scenarios where geometric assumptions break down, such as highly dynamic environments or scenes with significant non-rigid elements.

4.3 Neural Implicit Representations for Mapping

Neural implicit representations have emerged as a powerful alternative to traditional mapping techniques in SLAM. Unlike explicit representations such as point clouds, voxel grids, or mesh surfaces, implicit representations define the environment as a continuous function that maps spatial coordinates to properties like occupancy, signed distance to surfaces, or appearance.

In the context of SLAM, neural networks parameterize these functions, typically as multi-layer perceptrons (MLPs) that take 3D coordinates as input and output properties at those locations. This approach offers several compelling advantages:

- **Continuous Representation**: No discretization artifacts or resolution limitations
- **Memory Efficiency**: Complex scenes represented by network parameters rather than explicit storage
- **Differentiability**: Enables gradient-based optimization and integration with learning pipelines
- **Regularization**: Networks naturally interpolate between observations and regularize noisy data

Several types of neural implicit representations have been applied to SLAM:

Neural Radiance Fields (NeRF) encode both geometry and appearance, representing scenes as continuous volumetric fields. Originally developed for novel view synthesis from static images [20], NeRF has been adapted for SLAM in systems like iNeRF [21], which inverts the rendering process to estimate camera poses, and NICE-SLAM [22], which builds NeRF representations incrementally from RGB-D sequences.

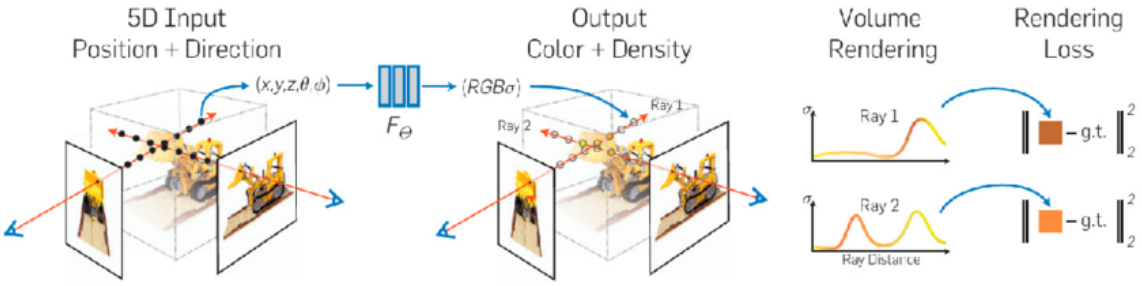

Figure 8: This figure illustrates the NeRF pipeline. The left shows 5D input coordinates (x, y, z, θ, ϕ) feeding into a neural network F_θ. The network outputs color and density values (RGB, σ) shown in the center. The right demonstrates volume rendering with two rays, showing accumulated color values and the L_2 rendering loss comparing predictions with ground truth. The scene reconstruction is visualized using an excavator model rendered from multiple viewpoints. . [23]

Signed Distance Functions (SDFs) represent surfaces as the zero level-set of a function that outputs the signed distance to the nearest surface. Neural SDFs encode this function as a MLP, providing a compact and detailed surface representation. Systems like iSDF [24] demonstrate how these representations can be built incrementally during SLAM operation.

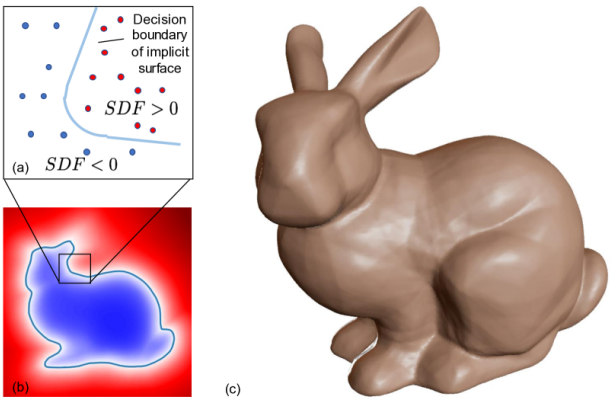

Figure 9: (a) depiction of the underlying implicit surface SDF $= 0$ trained on sampled points inside (SDF < 0) and outside (SDF > 0) the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from SDF $= 0$. [25]

Occupancy Networks learn a function that maps 3D points to occupancy probability, effectively encoding which parts of space are free or occupied.

4.4 Depth Estimation Networks

Depth estimation is a fundamental component of many SLAM systems, providing crucial 3D structure information from 2D images. Traditional approaches rely on multiple views (stereo or structure from motion) with explicit geometric constraints. Deep learning has transformed this area by enabling accurate depth estimation from limited inputs—even from single images—by leveraging learned priors about the world.

Monocular depth estimation networks address a fundamentally ill-posed problem: recovering scale-ambiguous 3D information from a single 2D projection. These networks succeed by learning statistical relationships between image appearance and geometric structure from large datasets. They effectively encode priors such as:

- Typical scales of common objects
- Relationship between texture gradients and depth
- Perspective effects on parallel lines
- Occlusion patterns and their relationship to relative depth

Modern architectures for depth estimation typically use encoder-decoder structures with skip connections, similar to U-Net designs. These networks process images at multiple scales to capture both fine details and broader context. For SLAM applications, depth networks are used in several ways:

Initialization and Bootstrapping: Monocular SLAM systems traditionally struggle with initialization and scale estimation. Depth networks provide initial depth maps that establish scale and bootstrap mapping without requiring specific initialization motions.

Filling Gaps: In semi-dense or sparse SLAM systems, learned depth can fill regions where geometric methods fail, particularly in textureless areas or regions with repetitive patterns.

Scale Recovery: Monocular SLAM inherently suffers from scale drift. Depth networks that provide absolute scale estimates can correct this drift throughout operation.

Multi-View Consistency: Modern systems like CNN-SLAM and DeepFactors [26] use depth predictions as priors in multi-view optimization, combining learning-based initial estimates with geometric refinement.

Self-supervised training has proven particularly valuable, allowing networks to learn from unlabeled video sequences by using photometric consistency loss between frames. This approach enables training on vast amounts of unlabeled data and adaptation to specific environments.

Depth estimation networks have become integral components in state-of-the-art SLAM systems, effectively complementing geometric methods. They are particularly valuable in challenging scenarios such as dynamic scenes, textureless regions, and monocular setups where traditional approaches struggle. As these networks continue to improve in accuracy, efficiency, and generalization capability, their role in SLAM systems is likely to expand further.

References

[1] Samer B. Nashed. "A Brief Survey of Loop Closure Detection: A Case for Rethinking Evaluation of Intelligent Systems". In: (2020). URL: https://ml-retrospectives.github.io/neurips2020/camera_ready/21.pdf.

[2] J.J. Leonard and H.F. Durrant-Whyte. "Simultaneous map building and localization for an autonomous mobile robot". In: (1991), 1442–1447 vol.3. DOI: 10.1109/IROS.1991.174711.

[3] Randall Smith, Matthew Self, and Peter Cheeseman. "Estimating Uncertain Spatial Relationships in Robotics". In: (2013). arXiv: 1304.3111 [cs.AI]. URL: https://arxiv.org/abs/1304.3111.

[4] Michael Montemerlo et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". In: (2002). URL: https://cdn.aaai.org/AAAI/2002/AAAI02-089.pdf.

[5] Andrew J. Davison et al. "MonoSLAM: Real-Time Single Camera SLAM". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067. DOI: 10.1109/TPAMI.2007.1049.

[6] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163. ISSN: 1941-0468. DOI: 10.1109/tro.2015.2463671. URL: http://dx.doi.org/10.1109/TRO.2015.2463671.

[7] Keiji Nagatani et al. "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots". In: *Journal of Field Robotics* 30 (Feb. 2013), pp. 44–63. DOI: 10.1002/rob.21439.

[8] Axel Barrau and Silvere Bonnabel. *An EKF-SLAM algorithm with consistency properties*. 2016. arXiv: 1510.06263 [cs.RO]. URL: https://arxiv.org/abs/1510.06263.

[9] Sherine Rady. "INFORMATION-THEORETIC ENVIRONMENT MODELING FOR MOBILE ROBOT LOCALIZATION". PhD thesis. Jan. 2012. DOI: 10.13140/RG.2.2.11156.30081.

[10] Umme Hani and Lubna Moin. "Realtime autonomous navigation in V-Rep based static and dynamic environment using EKF-SLAM". In: *IAES International Journal of Robotics and Automation (IJRA)* 10 (Dec. 2021), p. 296. DOI: 10.11591/ijra.v10i4.pp296-307.

[11] Nosan Kwak et al. "Adaptive prior boosting technique for the efficient sample size in FastSLAM". In: Oct. 2007, pp. 630–635. ISBN: 978-1-4244-0912-9. DOI: 10.1109/IROS.2007.4399039.

[12] Steven Gong. *Graph Optimization (Graph-Based SLAM)*. 2025. URL: https://stevengong.co/notes/Graph-Optimization.

[13] Tony Lindeberg. "Scale Invariant Feature Transform". In: vol. 7. May 2012. DOI: 10.4249/scholarpedia.10491.

[14] Yin-Tien Wang, Duan-Yan Hung, and Chung-Hsun Sun. "Improving Data Association in Robot SLAM with Monocular Vision". In: *Journal of Information Science and Engineering* 27 (Nov. 2011), pp. 1823–1837.

[15] Sen Wang et al. "DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017, pp. 2043–2050. DOI: 10.1109/icra.2017.7989236. URL: http://dx.doi.org/10.1109/ICRA.2017.7989236.

[16] Samarth Brahmbhatt et al. *Geometry-Aware Learning of Maps for Camera Localization*. 2018. arXiv: 1712.03342 [cs.CV]. URL: https://arxiv.org/abs/1712.03342.

[17] Keisuke Tateno et al. *CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction*. 2017. arXiv: 1704.03489 [cs.CV]. URL: https://arxiv.org/abs/1704.03489.

[18] Michael Bloesch et al. *CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM*. 2019. arXiv: 1804.00874 [cs.CV]. URL: https://arxiv.org/abs/1804.00874.

[19] Felix Ott et al. "ViPR: Visual-Odometry-aided Pose Regression for 6DoF Camera Localization". In: June 2020. DOI: 10.1109/CVPRW50498.2020.00029.

[20] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV]. URL: https://arxiv.org/abs/2003.08934.

[21] Lin Yen-Chen et al. *INeRF: Inverting Neural Radiance Fields for Pose Estimation*. 2021. arXiv: 2012.05877 [cs.CV]. URL: https://arxiv.org/abs/2012.05877.

[22] Zihan Zhu et al. *NICE-SLAM: Neural Implicit Scalable Encoding for SLAM*. 2022. arXiv: 2112.12130 [cs.CV]. URL: https://arxiv.org/abs/2112.12130.

[23] Dongtai Liang et al. "Scene Measurement Method Based on Fusion of Image Sequence and Improved LiDAR SLAM". In: *Electronics* 13 (Oct. 2024), p. 4250. DOI: 10.3390/electronics13214250.

[24] Joseph Ortiz et al. *iSDF: Real-Time Neural Signed Distance Fields for Robot Perception*. 2022. arXiv: 2204.02296 [cs.RO]. URL: https://arxiv.org/abs/2204.02296.

[25] Jeong Joon Park et al. *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*. 2019. arXiv: 1901.05103 [cs.CV]. URL: https://arxiv.org/abs/1901.05103.

[26] Jan Czarnowski et al. "DeepFactors: Real-Time Probabilistic Dense Monocular SLAM". In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 721–728. ISSN: 2377-3774. DOI: 10.1109/lra.2020.2965415. URL: http://dx.doi.org/10.1109/LRA.2020.2965415.