Simultaneous Localization and Mapping —Filtering and Optimization Approaches

Amay Saxena
Chih-Yuan (Frank) Chiu
Professor Shankar Sastry

02-10-2021

Outline

- **Motivation, FAQs, Terminology**
- **Front End**
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- **Back End**
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- **Back End—Example**
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- **Global Optimization**
- **Optimization on Manifolds**
- **Future**

Outline

- **Motivation, FAQs, Terminology**
- **Front End**
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- **Back End**
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- **Back End—Example**
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- **Global Optimization**
- **Optimization on Manifolds**
- **Future**

What is SLAM?

- **Motivation:**
  - If you know a robot's location, you can construct a map of its surroundings.
  - If you have a map of a robot's surroundings, you can determine its location.
  - What if you have neither?
- **Simultaneous Localization and Mapping: (SLAM)**
  - Localization — Determine the robot's state in its surroundings. The robot state may include its pose (position + orientation), velocity, etc.
  - Mapping — Construct a metric map of its surroundings. This is represented by objects of interest (landmarks, objects, etc.) in the robot's surroundings.

Leonard et al, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," IEEE Transactions on Robotics, 2016.

Why do we need SLAM?

- **Why SLAM is useful:**
  - SLAM research has produced the visual-inertial odometer algorithms used today (E.g., MSCKF)
  - SLAM allows use of metric information in establishing loop closures, thus helping the robot to construct a robust representation of the environment.
  - SLAM is necessary for many applications that require a globally consistent map (e.g., to construct a map and report back to a human operator).
- **When SLAM is unnecessary:**
  - When sufficient localization can be done without SLAM (e.g., Navigation scenario with access to GPS + LiDAR)
  - When a metric map is unnecessary for the task (e.g., Simple navigation tasks)

Leonard et al, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” IEEE Transactions on Robotics, 2016.

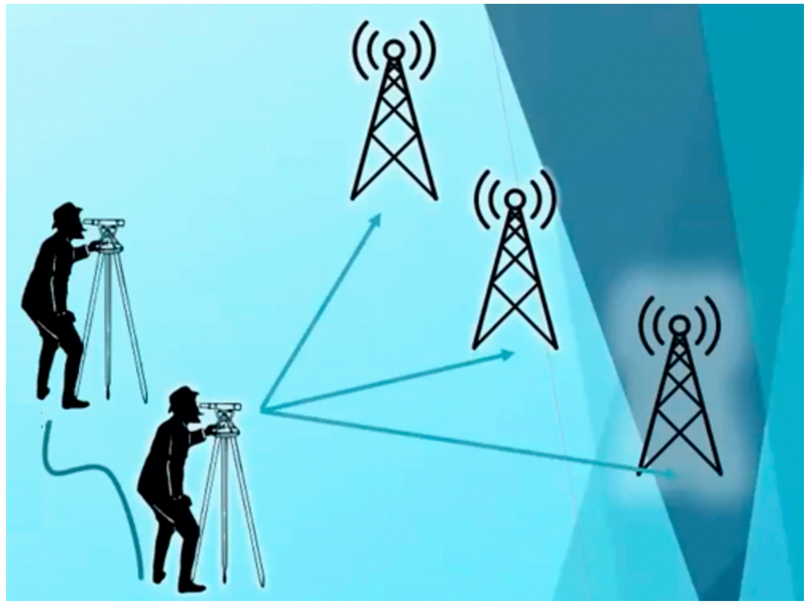
Is SLAM Solved?

- It depends on the robot (sensors), environment, performance requirement in question.
- **SLAM is solved for:**
  - Vision-based SLAM on slow robotic systems.
  - Mapping a 2D indoor environment with a robot equipped with wheel encoders and a laser scanner
- **SLAM is <u>not</u> solved for:**
  - Localization with highly agile robots, mapping rapidly evolving environments
  - Open problems— Robust performance, semantic understanding, resource awareness, task-driven perception.

Leonard et al, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” IEEE Transactions on Robotics, 2016.

SLAM — Problem Setup

- (1) Build a map with reference to the current location.
- (2) Move and estimate the updated location.
- (3) Observe mapped landmarks, and initialize new landmarks.
- (4) Use observations to update the position estimate and landmarks' positions.



Leonard et al, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," IEEE Transactions on Robotics, 2016.

Terminology

- **Full State Vector:**
  - Physical quantities describing the robot, iteratively refined in the SLAM problem
  - May include the IMU state, poses, and / or feature position estimates
  - Speed vs. accuracy tradeoff — Including more quantities in the robot state improves accuracy but lowers computational speed
- **Inertial Measurement Unit (IMU) State:**
  - Dynamic quantities of the robot, e.g. position, velocity, IMU biases.
  - Obtained from gyroscope (angular velocity), accelerometer (acceleration).
- **Pose:**
  - Position and orientation of the robot camera, corresponding to an image.
  - Described by a translational (e.g., vector) and rotational (e.g., SO(3) or quaternion) component.

Leonard et al, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” IEEE Transactions on Robotics, 2016.

Terminology

- **Image Measurement:**
  - 2D measurements of the surroundings, periodically captured by robot sensors (e.g., cameras)
  - Provides features, robot measurements
- **Feature:**
  - Positions (2D or 3D) of notable attributes in a collection of images (e.g., corners)
  - Used to identify correspondences between different images of the same part of the environment (e.g. an image patch of a repeatedly observed of a landmark).

Leonard et al, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” IEEE Transactions on Robotics, 2016.

SLAM—Front End and Back End

- **Front End:**
  - Extracts and processes features, converts signals from sensors into abstracted data (e.g., position, orientation, velocity, etc.).
- **Back End:**
  - Does inference over abstracted data (e.g., MAP Estimation of robot states).

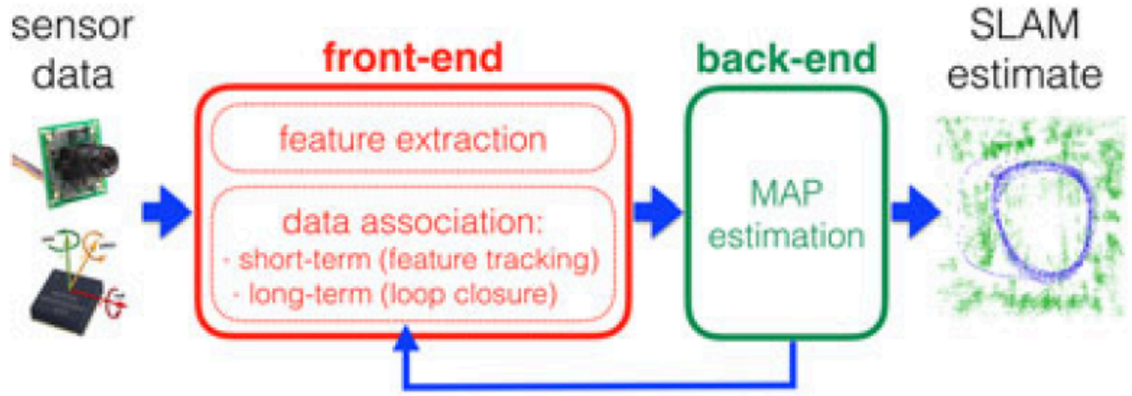


Fig. 2. Front end and back end in a typical SLAM system. The back end can provide feedback to the front end for loop closure detection and verification.

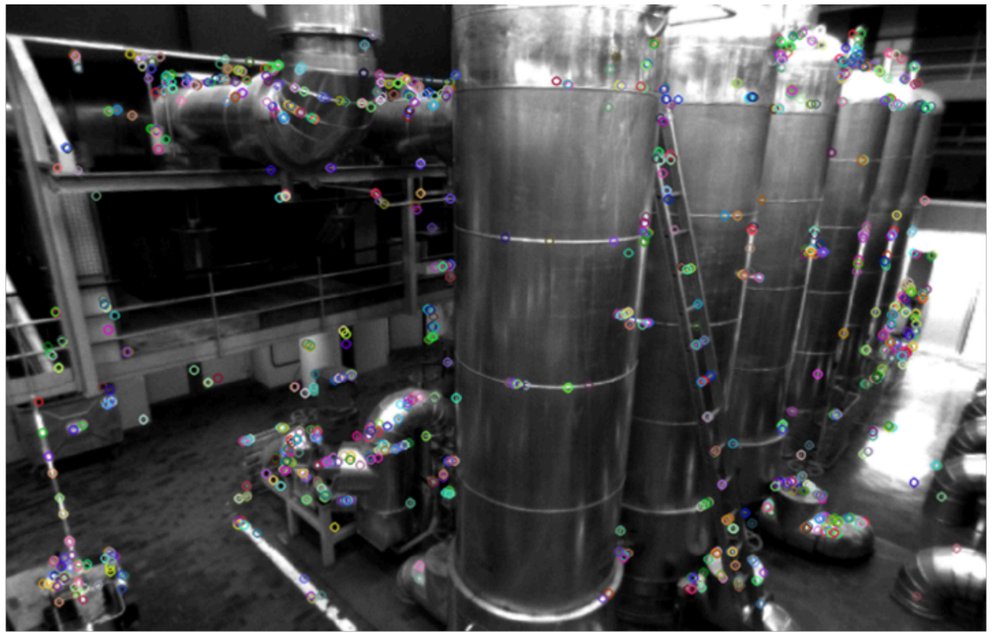
Leonard et al, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” IEEE Transactions on Robotics, 2016.

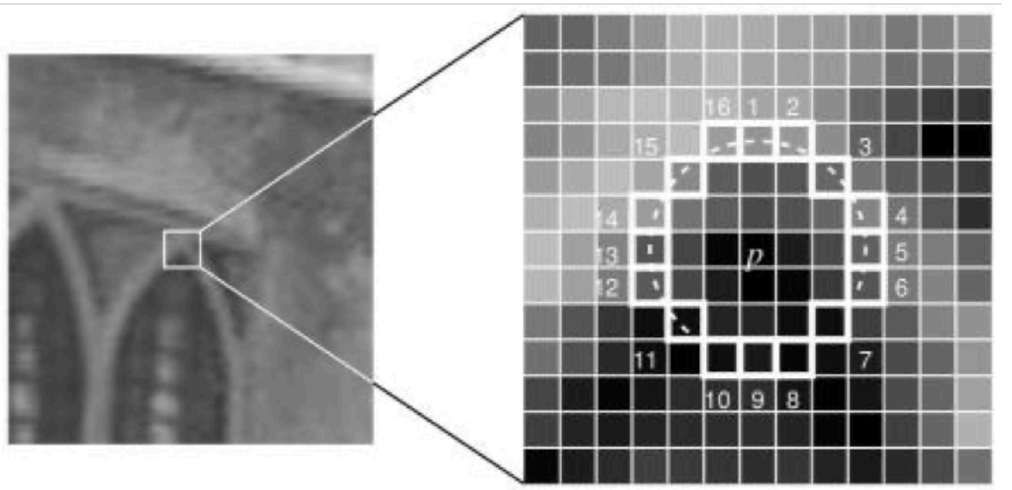
Outline

- Motivation, FAQs, Terminology
- **Front End**
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- Back End
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- Back End—Example
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- Global Optimization
- Optimization on Manifolds
- Future

Front End: Feature Extraction

- We want to extract repeatably detectable landmarks from raw images.
- Do this by looking for distinctive “image patches” that can be detected from multiple views of the scene.
- Corner points work well.
- Many commonly used corner detectors (e.g. FAST, HARRIS, DoG)





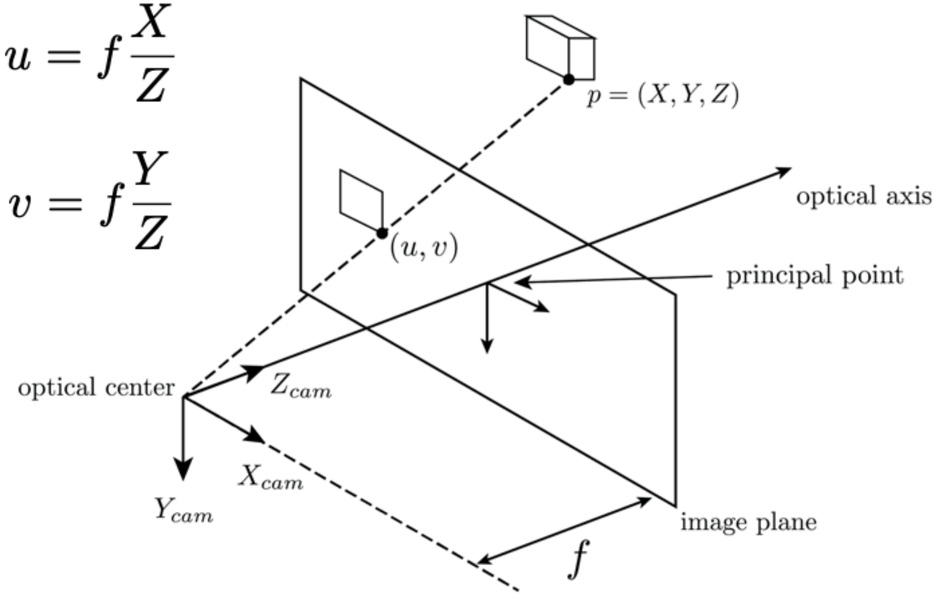
Rosten, Edward, and Tom Drummond. "Machine learning for high-speed corner detection." European conference on computer vision. Springer, Berlin, Heidelberg, 2006.

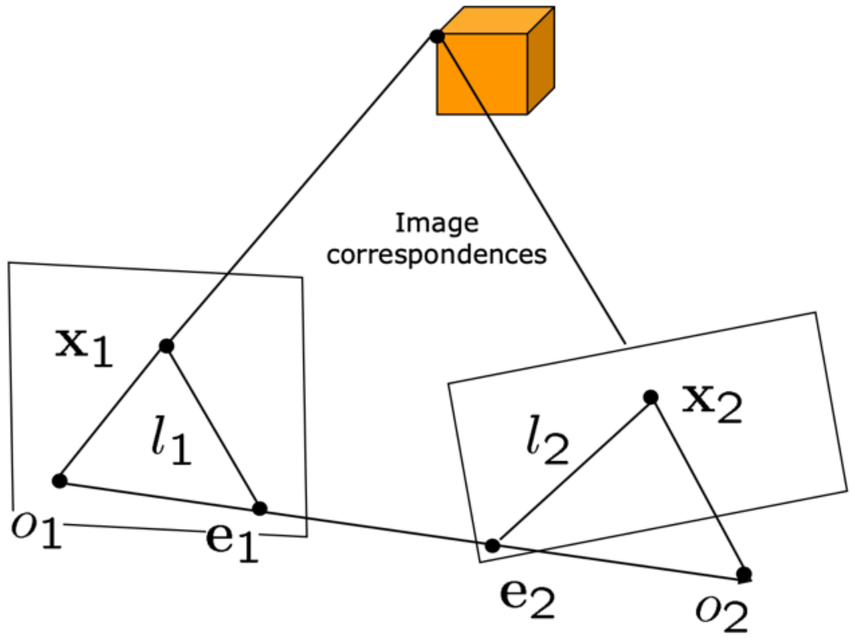
Camera is a bearing sensor

- From one observation of a point feature, we can only infer its bearing with respect to the optical axis.
- Can infer depth by re-observing the point from different angles (triangulation).

$$u = f\frac{X}{Z}$$

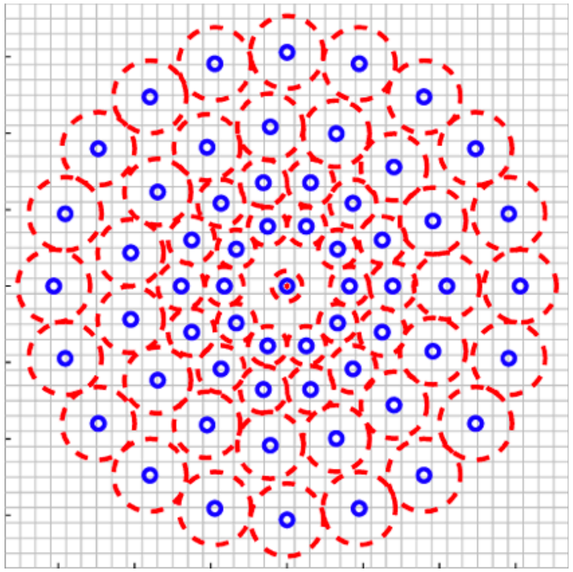
$$v = f\frac{Y}{Z}$$

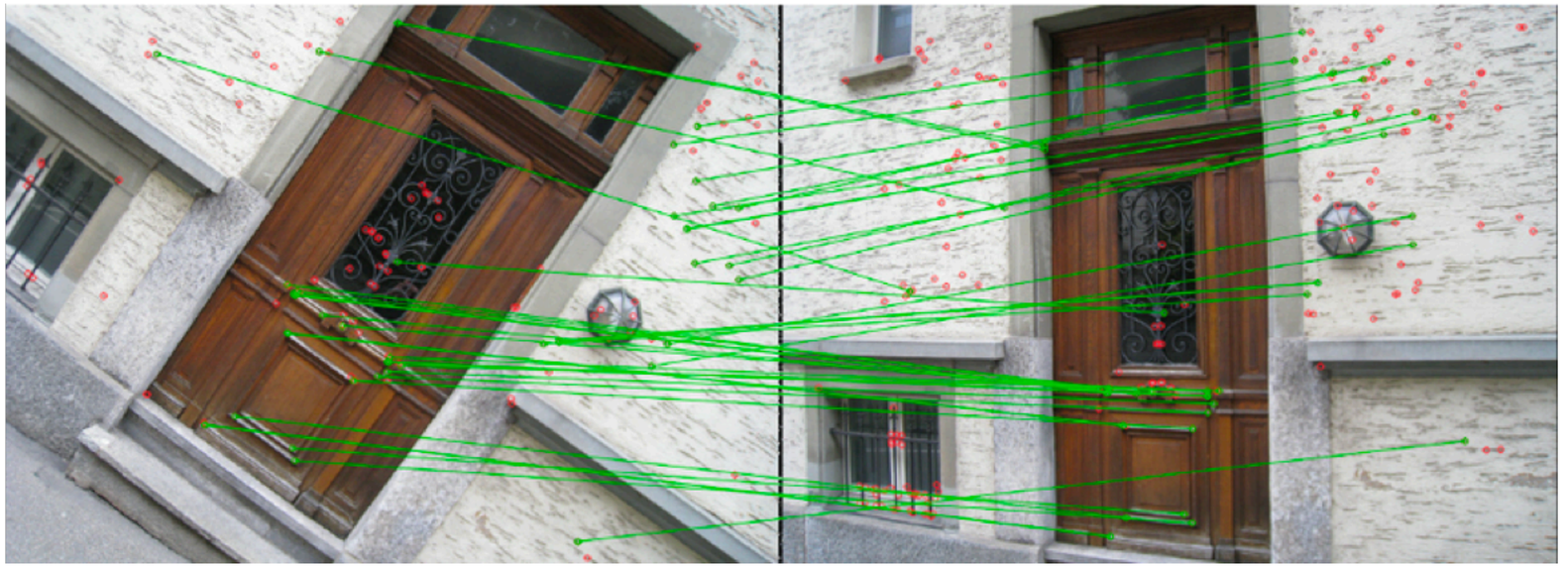




Front End: Data Association

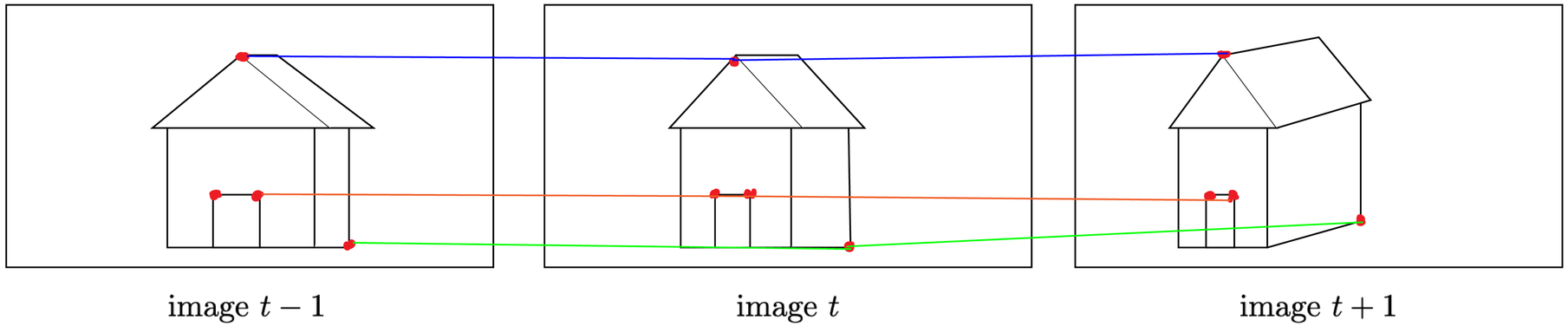
- So, we need a way to reliably detect the same point in multiple views of a scene.
- We need to construct a “descriptor” for the image patch surrounding a feature point in a way that is comparable, informative, and invariant to camera orientation.
- Many methods exist: BRISK, SURF, SIFT, BRIEF, ORB
- Extract features from both images, and then compare descriptors in a nearest-neighbor fashion to establish correspondences.





Front End: Data Association

Feature Tracks

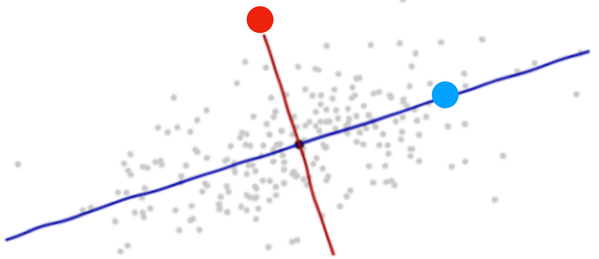


- Keep track of features throughout the image sequence as long as they are visible.
- Extract features from every image, then match descriptors between consecutive frames.
- Assumption is that matched image features correspond to the same point feature in the environment.
- Given estimate of camera pose, can get estimate of feature location by triangulating from multiple views.

Front End: Outlier Rejection

- Nearest neighbor matching of feature descriptors is a very local operation, and hence can lead to erroneous matches.
- Need to reject “outlier” feature matches.
    1. RANSAC (Random Sampling and Consensus)
        - Estimate fundamental matrix, reject matches that violate epipolar constraint.
        - Estimate Perspective-N-Point solution, reject matches with high reprojection error.
    2. Mahalanobis distance test (“chi-squared rejection test”)
        - When a known feature is detected, accept the match only if the location of the new image feature is within 3 standard deviations of the expected location given the current best estimate.

$$d(x) = \|x - \mu\|_{S^{-1}} = \sqrt{(x - \mu)^\top S^{-1} (x - \mu)}$$



Front End: Outlier Rejection

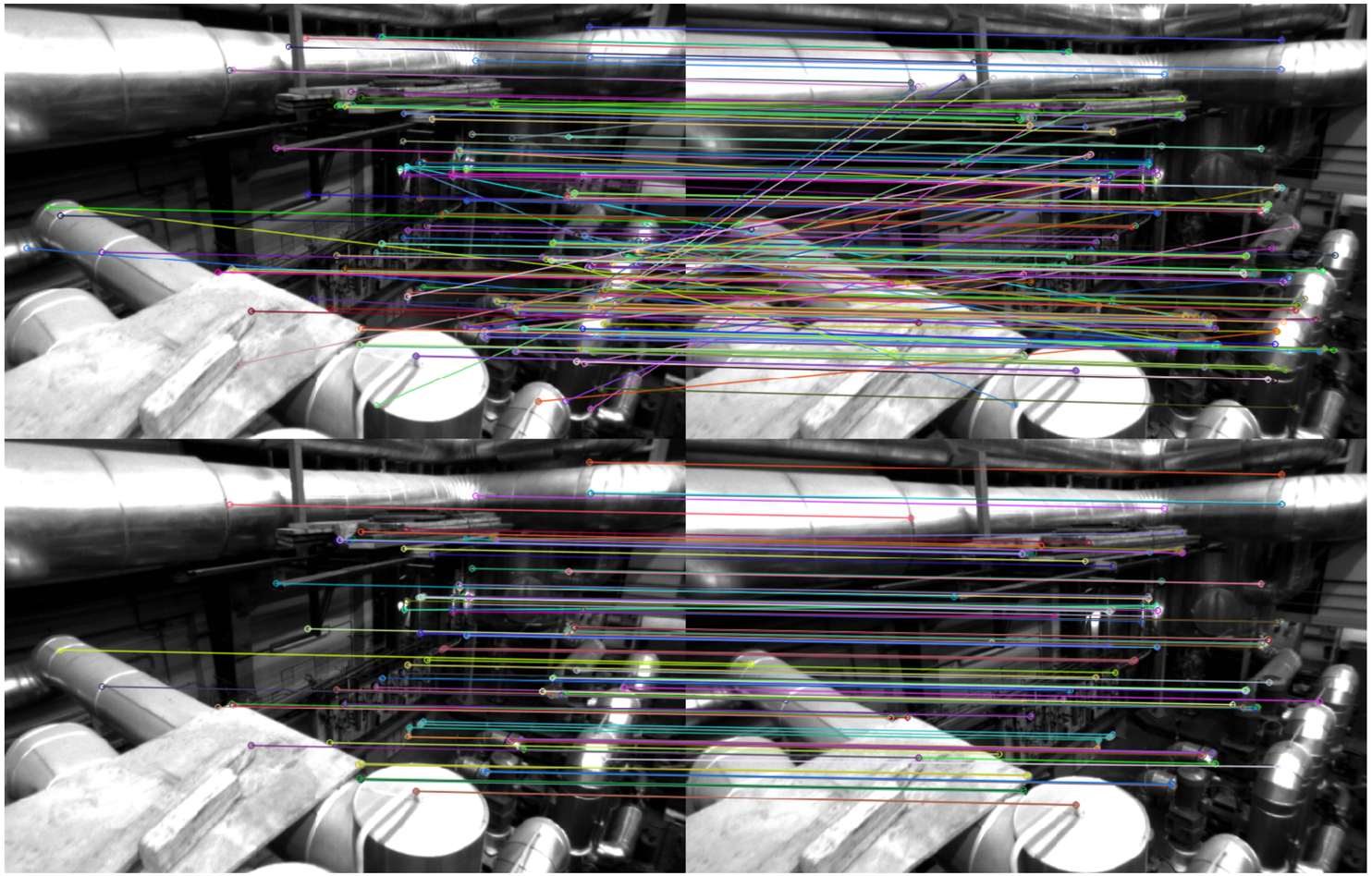


image at time t-1 image at time t

Top: Raw matches

Bottom: After outlier rejection using RANSAC

Front End: Other Tricks

- Use binary descriptors; comparison using hamming distance is faster than using float vector distance (BRISK, ORB, etc.).
- Pre-rotate image patches before describing them to achieve rotation invariance (ORB etc.).
- When IMU is available, can also pre-rotate features to be aligned with gravity vector (OK-Vis).

Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011

Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart and Paul Timothy Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 2015.

Outline

- Motivation, FAQs, Terminology
- Front End
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- **Back End**
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- Back End—Example
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- Global Optimization
- Optimization on Manifolds
- Future

Back End: Optimization Formulation

- **Key Idea:**
  - State-of-the-art SLAM algorithms exhibit different design choices
    - How much information, from our model and measurements, should our SLAM algorithm take as input?
    - How should we process that information?
  - Selecting design choices → Selecting different tradeoffs in computation time, localization accuracy, precision of constructed maps, etc.
  - **Goal**—Recast design choices into a unified optimization framework that allows:
    - (1) Experimentation with different designs,
    - (2) Interpolation between different state-of-the-art SLAM implementations.

Back End: Optimization Formulation

- **Filtering (Recursive) Framework:**
  - Dynamics map—Propagates mean / covariance of the state vector.
  - Measurement map + new images—Update mean / covariance of the state vector.
  - Use statistical tools to iteratively refine the mean and covariance of the state vector's posterior distribution.
- **Optimization (Smoothed) Framework:**
  - Dynamics and measurement maps impose constraints on the state vector, expressed as residuals.
  - These constraints are then encoded into a nonlinear least-squares cost.
  - The cost can then be solved via iterative linearization methods (e.g., Gauss-Newton algorithms).

Back End: Setup and Terminology

- **Full State:** (Variables to estimate)
  - Camera pose, at time t: $x_t \in \mathbb{R}^{d_x}, \forall t \geq 0.$
  - Position of feature j at time t in global frame: $f_{t,j} \in \mathbb{R}^{d_f}, \forall t \geq 0, j \geq 1.$
  - Full state, at time t: $\tilde{x}_t \in \mathbb{R}^d, \forall t \geq 0,$ with prior $N(\mu_t, \Sigma_t)$

    (The full state consists of the concatenation of multiple poses and features)
  -
- **Full State Example—Extended Kalman Filter (EKF):**
  - In EKF SLAM, the state vector consists of the most current pose and all features ever detected (e.g., p features):

$$\tilde{x}_t = (x_t, f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x + p d_f} := \mathbb{R}^d$$

Back End: Setup and Terminology

- **Dynamics map:** $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$
  - General form: $x_{t+1} = g(x_t) + w_t$, with $w_t \sim N(0, \Sigma_w), \forall t \geq 0.$

Pose at time $t+1$ — Pose at time t — Additive noise to dynamics at time t

  - Associated residual: $x_{t+1} - g(x_t)$

- **Examples:**
  - Discrete time robot model with associated input u_t.
  - Integrated IMU readings
  - (See Appendix for a simple, 2D example)

Back End: Setup and Terminology

- **Measurement map:** $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_z}$
  - Image measurement of feature j at time t: $z_{t,j} \in \mathbb{R}^{d_z}, \forall t, j \geq 0.$
  - General form: $z_{t,j} = h(x_t, f_{t,j}) + v_{t,j}$, with $v_{t,j} \sim N(0, \Sigma_v), \forall t \geq 0.$

Image measurement of feature j at time t

Pose at time t

Position estimate of feature j at time t

Additive noise to measurement of feature j at time t

  - Associated residual: $z_{t,j} - h(x_t, f_{t,j})$

Back End: Setup and Terminology

Measurement map Example—Pinhole camera model

$x_t \leftarrow$ pose of camera at time t

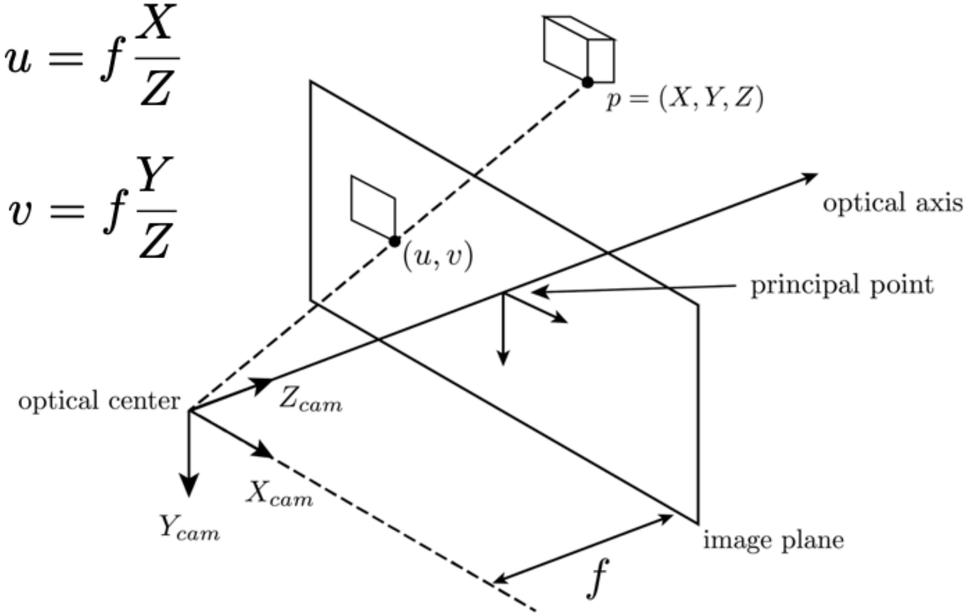
$f_j = (f_j^x, f_j^y, f_j^z) \leftarrow$ location of feature j

$\tilde{f}_j = (\tilde{f}_j^x, \tilde{f}_j^y, \tilde{f}_j^z) \leftarrow$ location of feature in camera frame (transformed using pose x_t)

$$h(x_t, f_j) = \frac{1}{\tilde{f}_j^z} \left(\tilde{f}_j^x, \tilde{f}_j^y \right)^\top$$

$$u = f\frac{X}{Z}$$

$$v = f\frac{Y}{Z}$$



Back End: Main Steps

- Cost Construction
- Gauss-Newton Step(s)
- Marginalization Step(s)

Back End: Cost Construction

- **Constraints imposed by dynamics and measurement maps:**
  - Dynamics map $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$:
  - Measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$:

$$x_{t+1} = g(x_t) + w_t, \text{ with } w_t \sim N(0, \Sigma_w), \forall t \geq 0.$$

Residual: $x_{t+1} - g(x_t)$

$$z_{t,j} = h(x_t, f_j) + v_{t,j}, \text{ with } v_{t,j} \sim N(0, \Sigma_v), \forall t \geq 0.$$

Residual: $z_{t,j} - h(x_t, f_j)$

- **Weighted Least-Squares Cost:**
  - Idea—If the models are accurate, then all residuals should be small.
  - Define cost = sum of norms squared of residuals we care about.
  - Ex: Suppose we wish to constrain residuals of n recent poses and p features:

$$c(\tilde{x}_t) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{i=t-n+1}^{t} \sum_{j=1}^{p} \|z_{i,j} - h(x_i, f_{t,j})\|^2_{\Sigma_v^{-1}} + \sum_{i=t-n+1}^{t-1} \|x_{i+1} - g(x_i)\|^2_{\Sigma_w^{-1}}$$

  - **Goal—Find $\tilde{x}_t$ that minimizes $c(\tilde{x}_t)$.**

(where $\|v\|_\Sigma := v^\top \Sigma v$)

Back End: Gauss-Newton Step

- **Concatenated Cost Vector (Example)**
  - Stacking up the residual terms in the cost, we have:
- $$c(\tilde{x}_t) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{i=t-n+1}^{t} \sum_{j=1}^{p} \|z_{i,j} - h(x_i, f_{t,j})\|^2_{\Sigma_v^{-1}} + \sum_{i=t-n+1}^{t-1} \|x_{i+1} - g(x_i)\|^2_{\Sigma_w^{-1}}$$

$$= \|\Sigma_t^{-1/2}(\tilde{x}_t - \mu_t)\|^2 + \sum_{i=t-n+1}^{t} \sum_{j=1}^{p} \|\Sigma_v^{-1/2}(z_{i,j} - h(x_i, f_{t,j}))\|^2 + \sum_{i=t-n+1}^{t-1} \|\Sigma_w^{-1/2}(x_{i+1} - g(x_i))\|^2$$

Define the full cost vector $C(\tilde{x}_t)$ by:

$$C(\tilde{x}_t) := \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t - \mu_t) \\ \Sigma_v^{-1/2}(z_{t-n+1,1} - h(x_{t-n+1}, f_1)) \quad \cdots \quad \Sigma_v^{-1/2}(z_{t-n+1,p} - h(x_{t-n+1}, f_p)) \\ \vdots \\ \Sigma_v^{-1/2}(z_{t,1} - h(x_t, f_1)) \quad \cdots \quad \Sigma_v^{-1/2}(z_{t,p} - h(x_t, f_p)) \\ \Sigma_w^{-1/2}(x_{t-n+2} - g(x_{t-n+1})) \quad \cdots \quad \Sigma_w^{-1/2}(x_t - g(x_{t-1})) \end{bmatrix}$$

$\in \mathbb{R}^d$

$\in \mathbb{R}^{d+npd_z+(n-1)d_x}$

$\in \mathbb{R}^{npd_z}$

$\in \mathbb{R}^{(n-1)d_x}$

We then have: $c(\tilde{x}_t) = \|C(\tilde{x}_t)\|^2$

Back End: Gauss-Newton Step

- **Jacobian Computation:**
  - Jacobian of $C(\tilde{x}_t)$ at linearization point $\tilde{x}_t^\star$:

$$\tilde{x}_t^\star = \mu_t \qquad J := \left.\frac{dC}{d\tilde{x}_t}\right|_{\tilde{x}_t^\star}$$

- **Gauss-Newton Update:**
  - 1st-order Taylor expansion of $c(\tilde{x}_t)$ about $\tilde{x}_t^\star$:

$$\begin{aligned}\min_{\tilde{x}_t} c(\tilde{x}_t) := \min_{\tilde{x}_t} \|C(\tilde{x}_t)\|^2 &= \min_{\tilde{x}_t} \left(\left\| C(\mu_t) + J(\tilde{x}_t - \mu_t) \right\|^2 + o(\tilde{x}_t - \mu_t) \right) \\ &= \min_{\tilde{x}_t} \left((\tilde{x}_t - \overline{\mu}_t)^\top \overline{\Sigma}_t^{-1} (\tilde{x}_t - \overline{\mu}_t) + \text{const} + o(\tilde{x}_t - \mu_t) \right).\end{aligned}$$

where:

$$\begin{aligned}\overline{\Sigma}_t^{-1} &\leftarrow J^\top J, \\ \overline{\mu}_t &\leftarrow \mu_t - (J^\top J)^{-1} J^\top C(\mu_t).\end{aligned}$$

Back End: Marginalization

- **Motivation:**
  - To reduce computational burden, sometimes we wish to remove components of $\tilde{x}_t$ from the optimization problem.
- **Splitting the State and Cost:**
  - Splitting the Concatenated State:

where:

$$\tilde{x}_t := (\tilde{x}_{t,K}, \tilde{x}_{t,M}) \in \mathbb{R}^{d_K} \times \mathbb{R}^{d_M}, \text{ with } d_K + d_M = d :$$
$$\tilde{x}_{t,K} := \text{ component of } \tilde{x}_t \text{ to keep,}$$
$$\tilde{x}_{t,M} := \text{ component of } \tilde{x}_t \text{ to marginalize.}$$

  - Splitting the Cost:

where:

$$c(\tilde{x}_t) = c(\tilde{x}_{t,K}, \tilde{x}_{t,M}) = c_1(\tilde{x}_{t,K}) + c_2(\tilde{x}_{t,K}, \tilde{x}_{t,M}),$$
$$= \|C_1(\tilde{x}_{t,K})\|^2 + \|C_2(\tilde{x}_{t,K}, \tilde{x}_{t,M})\|^2$$
$$c_1(\tilde{x}_{t,K}) := \text{norm squared of residuals in } \tilde{x}_t \text{ that depend only on } \tilde{x}_{t,K}$$
$$c_2(\tilde{x}_{t,K}, \tilde{x}_{t,M}) := \text{norm squared of other residuals in } \tilde{x}_t$$

Back End: Marginalization

- **Splitting the State and Cost:**
  - To remove $\tilde{x}_{t,M}$, minimize $c(\tilde{x}_t) = c(\tilde{x}_{t,K}, \tilde{x}_{t,M})$ w.r.t. $\tilde{x}_{t,M}$:

$$\begin{aligned}\min_{\tilde{x}_t} c(\tilde{x}_t) &= \min_{\tilde{x}_{t,K}, \tilde{x}_{t,M}} \left(c_1(\tilde{x}_{t,K}) + c_2(\tilde{x}_{t,K}, \tilde{x}_{t,M}) \right) \\ &= \min_{\tilde{x}_{t,K}} \left(c_1(\tilde{x}_{t,K}) + \min_{\tilde{x}_{t,M}} c_2(\tilde{x}_{t,K}, \tilde{x}_{t,M}) \right) \\ &= \min_{\tilde{x}_{t,K}} \left(\|C_1(\tilde{x}_{t,K})\|^2 + \min_{\tilde{x}_{t,M}} \|C_2(\tilde{x}_{t,K}, \tilde{x}_{t,M})\|^2 \right)\end{aligned}$$

  - **Goal**—Replace $\min_{\tilde{x}_{t,M}} \|C_2(\tilde{x}_{t,K}, \tilde{x}_{t,M})\|^2$ with a linear least-squares term:

$$(\tilde{x}_{t,K} - \overline{\mu}_{t,K})^\top \overline{\Sigma}_{t,K}^{-1} (\tilde{x}_{t,K} - \overline{\mu}_{t,K})$$

where the mean $\overline{\mu}_{t,K}$ and covariance $\overline{\Sigma}_{t,K}$ encapsulate information encoded in the residuals comprising $C_2(\tilde{x}_{t,K}, \tilde{x}_{t,M})$.

Back End: Marginalization

- **Jacobian Computation:**
  - Jacobian of $c(\tilde{x}_t)$ at linearization point $\tilde{x}_t^\star$:

$$\tilde{x}_t^\star = (\tilde{x}_{t,K}, \tilde{x}_{t,M}) = (\mu_{t,K}, \mu_{t,M}) \qquad J_K := \left.\frac{dC_2}{d\tilde{x}_{t,K}}\right|_{\tilde{x}_{t,K}^\star} \qquad J_M := \left.\frac{dC_2}{d\tilde{x}_{t,M}}\right|_{\tilde{x}_{t,M}^\star}$$

- **Marginalization:**
  - Linearly approximate $c_2(\tilde{x}_{t,K}, \tilde{x}_{t,M}) = \|C_2(\tilde{x}_{t,K}, \tilde{x}_{t,M})\|^2$:

$$\begin{aligned} c_2(\tilde{x}_t) &= \|C_2(\tilde{x}_{t,K}, \tilde{x}_{t,M})\|^2 \\ &= \underbrace{\left\|C_2(\mu_{t,K}, \mu_{t,M}) + J_K(\tilde{x}_{t,K} - \mu_{t,K}) + J_M(\tilde{x}_{t,M} - \mu_{t,M})\right\|^2}_{:= c_2'(\tilde{x}_t)} \\ &\quad + o(\tilde{x}_{t,K} - \mu_{t,K}) + o(\tilde{x}_{t,M} - \mu_{t,M}). \end{aligned}$$

  - **Goal— Minimize $c_2'(\tilde{x}_t)$, which is easier than minimizing $c_2(\tilde{x}_t)$.**

Back End: Marginalization

- **Marginalization:**
  - Minimize $c_2'(\tilde{x}_{t,K}, \tilde{x}_{t,M})$ w.r.t. $\tilde{x}_{t,M}$:

$$\begin{aligned}\min_{\tilde{x}_{t,M}} c_2'(\tilde{x}_t) &= \min_{\tilde{x}_{t,M}} \left\| C_2(\mu_{t,K}, \mu_{t,M}) + J_K(\tilde{x}_{t,K} - \mu_{t,K}) + J_M(\tilde{x}_{t,M} - \mu_{t,M}) \right\|^2 \\ &= \|\tilde{x}_{t,K} - \overline{\mu}_{t,K}\|_{\overline{\Sigma}_{t,K}^{-1}} + \text{const.}\end{aligned}$$

where:
$$\begin{aligned}\overline{\Sigma}_{t,K}^{-1} &\leftarrow J_K^\top \left[I - J_M (J_M^\top J_M)^{-1} J_M^\top \right] J_K, \\ \overline{\mu}_{t,K} &\leftarrow \mu_{t,K} - \overline{\Sigma}_{t,K} J_K^\top \left[I - J_M (J_M^\top J_M)^{-1} J_M^\top \right] C_2(\mu_{t,K}, \mu_{t,M})\end{aligned}$$

  - **Note**—This is related to the Schur complement method in optimization theory (see Appendix).

- $$\min_{\tilde{x}_{t,M}} c_2'(\tilde{x}_t) = \min_{\tilde{x}_{t,M}} \begin{bmatrix} C_2(\mu_{t,K}, \mu_{t,M}) \\ \tilde{x}_{t,K} - \mu_{t,K} \\ \tilde{x}_{t,M} - \mu_{t,M} \end{bmatrix}^\top \begin{bmatrix} I & J_K & J_M \\ J_K^\top & J_K^\top J_K & J_K^\top J_M \\ J_M^\top & J_M^\top J_K & J_M^\top J_M \end{bmatrix} \begin{bmatrix} C_2(\mu_{t,K}, \mu_{t,M}) \\ \tilde{x}_{t,K} - \mu_{t,K} \\ \tilde{x}_{t,M} - \mu_{t,M} \end{bmatrix}.$$

Outline

- Motivation, FAQs, Terminology
- Front End
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- Back End
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- **Back End—Example**
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- Global Optimization
- Optimization on Manifolds
- Future

Extended Kalman Filter (EKF)

- **Introduction:**
  - Extended Kalman Filter — Recursive, filtering-based algorithm for updating a full state vector, using dynamics and measurement maps.
- **Brief Outline of Upcoming Slides:**
  - Present Extended Kalman Filter (EKF) in its standard formulation
  - Present algorithm modules for the key steps of the EKF algorithm— **Feature augmentation, feature update, and state propagation**
  - Define cost functions, and apply the optimization framework on previous slides.
  - Conclusion — Descent steps on the cost functions give the same update and propagation equations, for the full state, as the EKF algorithm modules

Extended Kalman Filter (EKF)

- **Setup:**
  - Full State $\tilde{x}_t$—In EKF SLAM, the state vector consists of the most current pose and all features ever detected (e.g., p features):

$$\tilde{x}_t = (x_t, f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x + p d_f} := \mathbb{R}^d$$

  - Dynamics, Measurement Maps:

Dynamics map $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$: $x_{t+1} = g(x_t) + w_t$, with $w_t \sim N(0, \Sigma_w), \forall t \geq 0.$

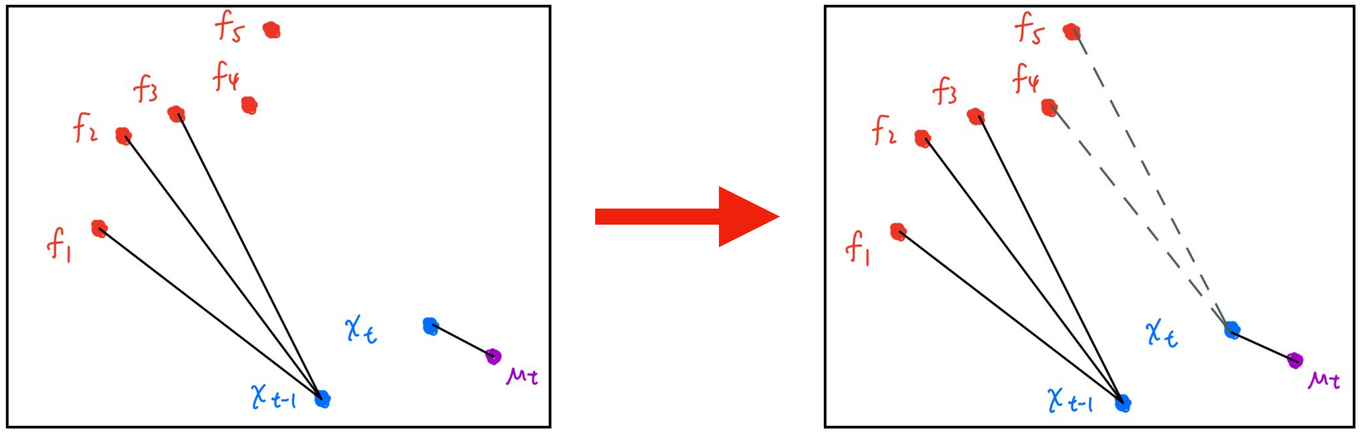
Measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$: $z_{t,j} = h(x_t, f_j) + v_{t,j}$, with $v_{t,j} \sim N(0, \Sigma_v), \forall t \geq 0.$

(For detailed descriptions of g, h, see previous slides)

  - Steps for iteratively refining $\tilde{x}_t$ — **Feature augmentation, feature update, and state propagation.**

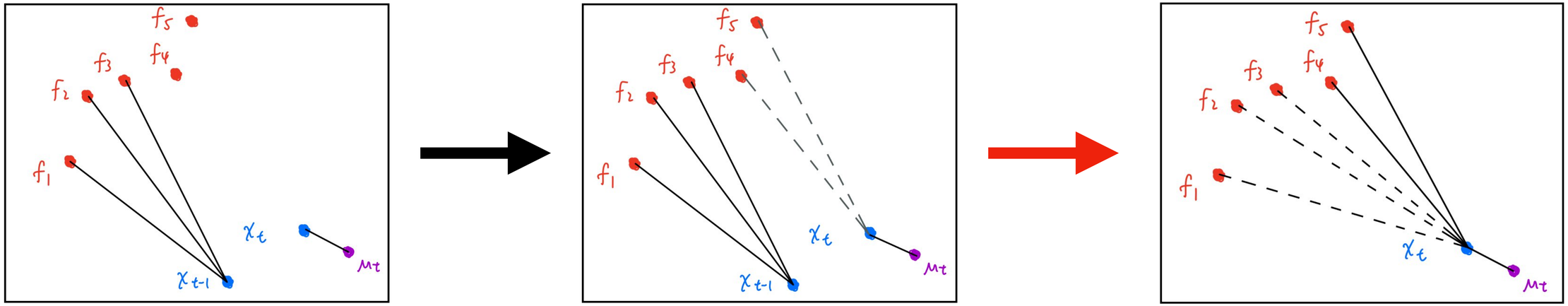
Extended Kalman Filter (EKF)

- **Step 1—Feature Augmentation:**
  - Augment $\tilde{x}_t$ with position estimates of newly detected features



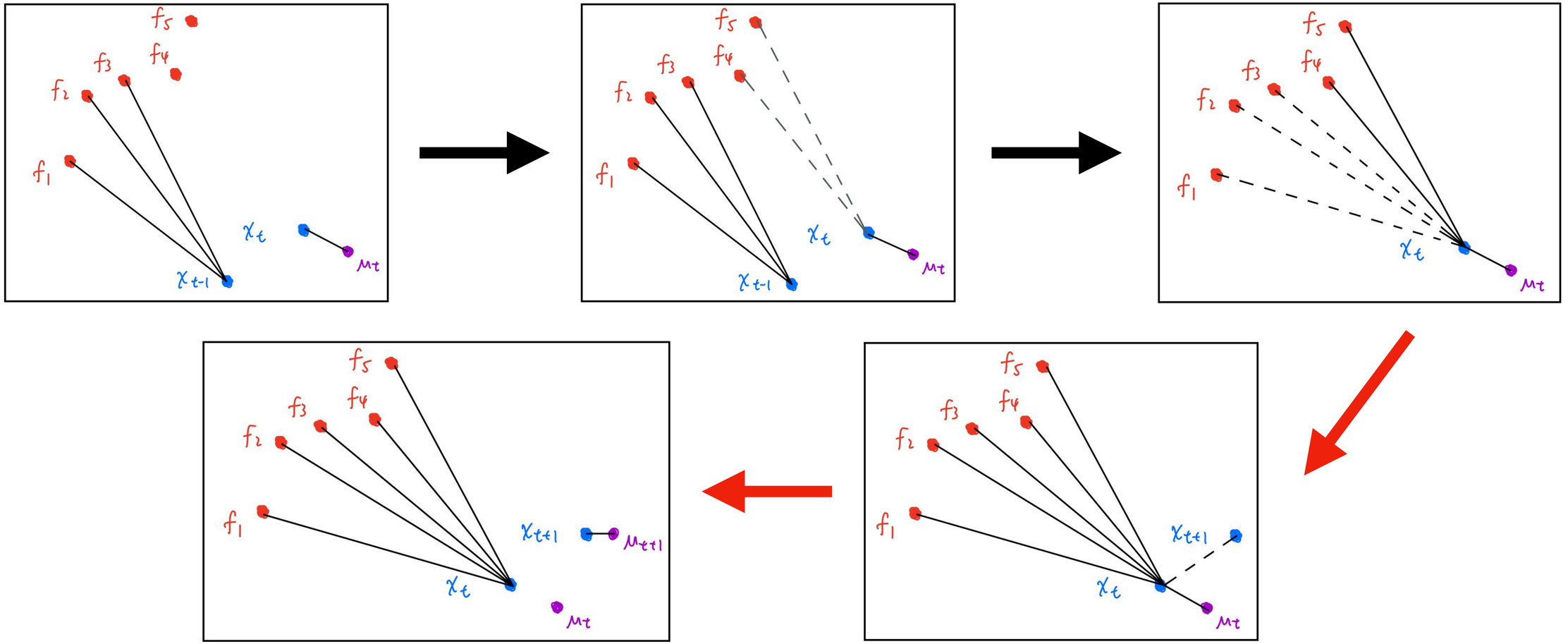
Extended Kalman Filter (EKF)

- **Step 2—Feature Update:**
  - Update $\tilde{x}_t$ with position estimates of features already described in $\tilde{x}_t$



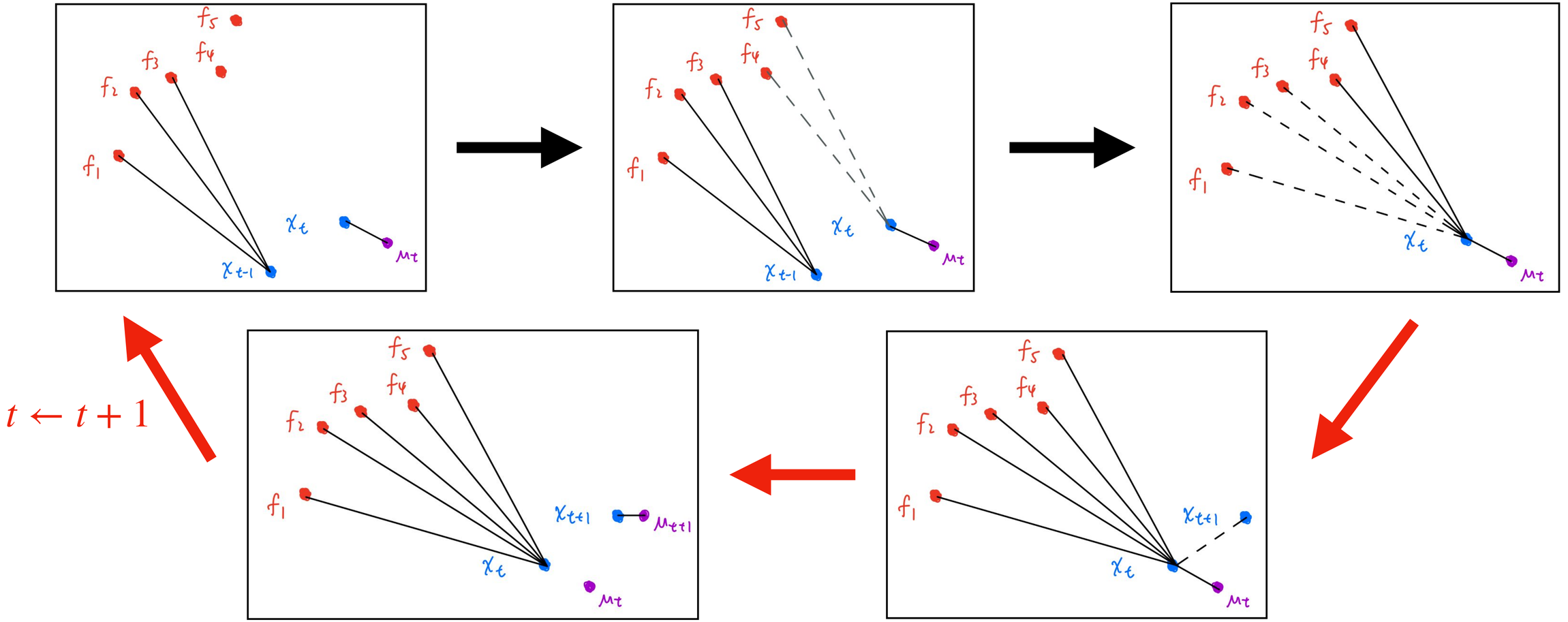
Extended Kalman Filter (EKF)

- **Step 3—State Propagation:**
  - In $\tilde{x}_t$, replace the current pose x_t with the new pose x_{t+1}



Extended Kalman Filter (EKF)

- **Repeat Steps 1 to 3:**
  - Increment t by 1, and repeat.



Extended Kalman Filter (EKF)

- **EKF, Standard Formulation:** **(Algorithm 1)**

Algorithm 1: Extended Kalman Filter SLAM, Standard Formulation.

Data: Prior distribution on $x_0 \in \mathbb{R}^{d_x}$: $\mathcal{N}(\mu_0, \Sigma_0)$, dynamics and measurement noise covariances $\Sigma_w \in \mathbb{R}^{d_x \times d_x}, \Sigma_v \in \mathbb{R}^{d_z \times d_z}$, (discrete-time) dynamics map $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{pd_f} \to \mathbb{R}^{d_z}$, time horizon $T \in \mathbb{N}$.

Result: Estimates $\hat{x}_t$ for all desired timesteps $t \leq T$.

For loop running through the finite time horizon →

1 **for** $t = 0, \cdots, T$ **do**

2 **if detect new feature measurements** $z_{t,p+1:p+p'} := (z_{t,p+1}, \cdots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}$ **then**

Feature Augmentation Step →

3 $\mu_t, \Sigma_t, p \leftarrow$ Alg. 2, EKF feature augmentation $\left(\mu_t, \Sigma_t, p, z_{t,p+1:p+p'}, h(\cdot)\right)$

4 **end**

5 $z_{t,1:p} := (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{pd_z} \leftarrow$ New measurements of existing features.

Feature Update Step →

6 $\overline{\mu_t}, \overline{\Sigma_t} \leftarrow$ Alg. 3, EKF feature update $\left(\overline{\mu_t}, \overline{\Sigma_t}, z_{t,1:p}, h(\cdot)\right)$.

7 **if** $t < T$ **then**

State Propagation Step →

8 $\mu_{t+1}, \Sigma_{t+1} \leftarrow$ Alg. 4, EKF state propagation $\left(\mu_t, \Sigma_t, g(\cdot)\right)$

9 **end**

10 **end**

11 **return** $\hat{x}_0, \cdots \hat{x}_T \in \mathbb{R}^{d_x}$.

Optimization-Based Framework

- **Feature Augmentation = Gauss-Newton Step:**

Theorem 6.1. *The feature augmentation step of the EKF SLAM algorithm is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1}, \cdots, f_{t,p+p'}) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\tilde{\Sigma}_v^{-1}}.$$

- **Idea of Theorem:**
  - Filtering approach—Linearize inverse measurement map (image → feature position), then perform a MAP estimate update using this linearized function
  - Optimization approach—Use measurement map to form a nonlinear least-squares cost, then perform one Gauss-Newton step on this cost
  - This theorem shows that these two approaches are equivalent.

Optimization-Based Framework

- **Feature Update = Gauss-Newton Step:**

Theorem 6.2. *The feature update step of the EKF SLAM algorithm (Alg. 3) is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,3}(\tilde{x}_t) := \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_k)\|^2_{\Sigma_v^{-1}}.$$

- **Idea of Theorem:**
  - Filtering approach—Linearize measurement map, then perform a MAP estimate update using this linearized function
  - Optimization approach—Use measurement map to form a nonlinear least-squares cost, then perform one Gauss-Newton step on this cost
  - This theorem shows that these two approaches are equivalent.

Optimization-Based Framework

- **State Propagation = Marginalization Step:**

Theorem 6.3. *The state propagation step of the EKF SLAM algorithm (Alg. 4) is equivalent to one Marginalization step on the cost function* $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) = \|\tilde{x}_t - \overline{\mu}_t\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}$$

- **Idea of Theorem:**
  - Filtering approach—Linearize dynamics map, then perform a MAP estimate update using this linearized function
  - Optimization approach—Use dynamics map to form a nonlinear least-squares cost, then perform one marginalization step on this cost
  - This theorem shows that these two approaches are equivalent.

State-of-the-art SLAM Algorithms:

- Below, we cast state-of-the-art SLAM algorithms into our optimization framework, and compare / contrast them
- One iterate of each algorithm is described as follows:
  - **(1) Extended Kalman Filter (EKF)** — Performs 1 Gauss-Newton step, marginalizes all poses but current one, does not marginalize features
  - **(2) Iterative Extended Kalman Filter (iEKF)** — Same as EKF SLAM, but performs multiple Gauss-Newton steps

J. Sola. Simulataneous localization and mapping with the Extended Kalman Filter. arXiv, 2014.
S. Tully, Hyungpil Moon, G. Kantor, and H. Choset. Iterated filters for bearing-only SLAM, ICRA, 2008, pages 1442-1448.

State-of-the-art SLAM Algorithms:

- Below, we cast state-of-the-art SLAM algorithms into our optimization framework, and compare / contrast them
- One iterate of each algorithm is described as follows:
  - **(3) Multi-State Constrained Kalman Filter (MSCKF)** — Performs 1 Gauss-Newton step, marginalizes all poses except the most recent k poses, marginalize features as soon as they are no longer seen by the current pose
  - **(4) Fixed Lag Smoother —** Same as MSCKF, but performs multiple Gauss-Newton steps
  - **(5) Open Keyframe Visual-Inertial SLAM (OK-Vis) —** Same as MSCKF, but performs multiple Gauss-Newton steps and drops some intermediate poses (used in OpenARK)

Mourikis and S. I. Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation.Proceedings, ICRA, pages 3565–3572, 2007.

S. Tully, Hyungpil Moon, G. Kantor, and H. Choset. Iterated filters for bearing-only SLAM, ICRA, pages 1442–1448, 2008.

S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization. The International Journal of Robotics Research, 34:314 – 334, 2015.

State-of-the-art SLAM Algorithms:

- Below, we cast state-of-the-art SLAM algorithms into our optimization framework, and compare / contrast them
- One iterate of each algorithm is described as follows:
  - **(6) Graph SLAM —** Performs multiple Gauss-Newton steps when all information has been collected, does not marginalize any variable.
  - **(7) Bundle Adjustment —** Same as Graph SLAM, but without motion constraints, may add more constraints to account for scale ambiguity
  - **(8) Pose Graph SLAM —** Same as Graph SLAM, but measurements are relative pose constraints.

S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005.
G. Grisetti, R. K~A 14mmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. IEEE Intelligent Transportation Systems Magazine, 2(4):31{43, 2010.
F. Daellert, “Visual SLAM Tutorial: Bundle Adjustment,” CVPR 2014.

Outline

- **Motivation, FAQs, Terminology**
- **Front End**
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- **Back End**
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- **Back End—Example**
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- **Global Optimization**
- **Optimization on Manifolds**
- **Future**

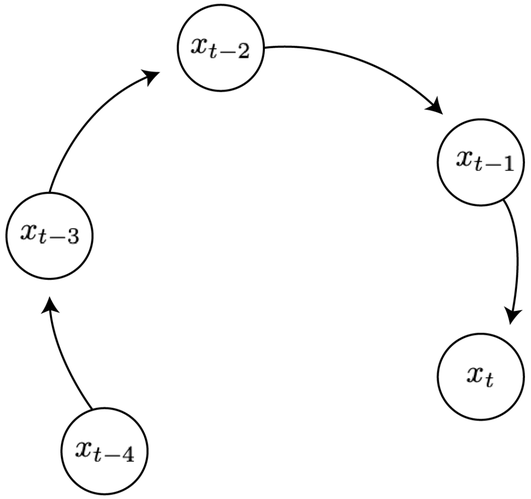
Global Optimization

- In addition to real-time tracking, we also maintain a "global" optimization problem over all robot poses.
- Every time a pose is marginalized out of the local problem, it is introduced into the global problem.
- The global optimization maintains global consistency between the poses, and is what separates SLAM from odometry.

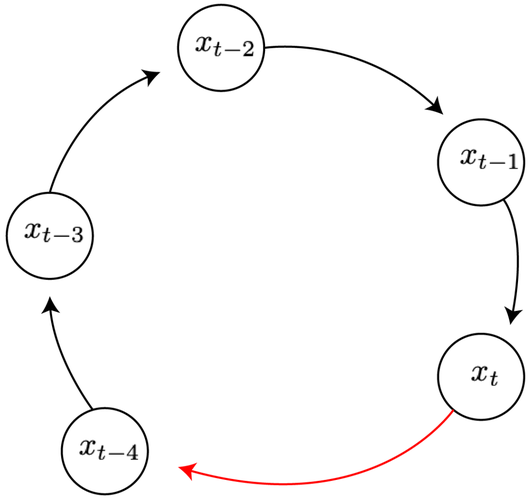
- In addition to incremental pose constraints, we also introduce "loop closure" constraints which are activated when the robot re-visits a part of the map it has seen before.

Global Optimization

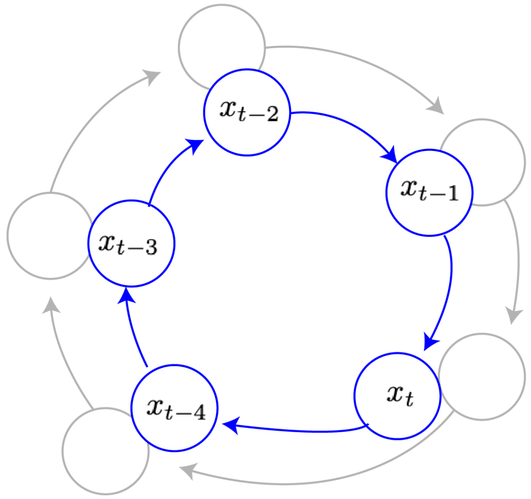
- In addition to incremental pose constraints, we also introduce "loop closure" constraints which are activated when the robot re-visits a part of the map it has seen before.
- Inference is done each time a new loop closure is registered.



1) Original global problem



2) Introduce loop closure constraint



3) Run inference

Global Optimization: Loop Closure Detection

- To establish loop closures, we need to detect when the camera is looking at the same place as some time in the past.
- Naive way: compare every detected feature to every feature we have seen so far. Terrible.

- Instead, we seek to compute a global descriptor of the entire image, which can be compared to descriptors of past images.
  - Bag-of-words approach: assigns a binary descriptor to each image encoding the presence or absence of certain features.

Gálvez-López, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." IEEE Transactions on Robotics 28.5 (2012)

Outline

- **Motivation, FAQs, Terminology**
- **Front End**
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- **Back End**
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- **Back End—Example**
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- **Global Optimization**
- **Optimization on Manifolds**
- **Future**

The elephant in the room...

- So far, we have freely been speaking of performing a Taylor expansion on an objective function f.

$$f(x+\delta) \approx f(x) + Df(x)\delta$$

- In the 3D SLAM case, the variable x consists of *poses*:

$(R,T) \in SE(3)$ with rotation matrix R and translation vector T.

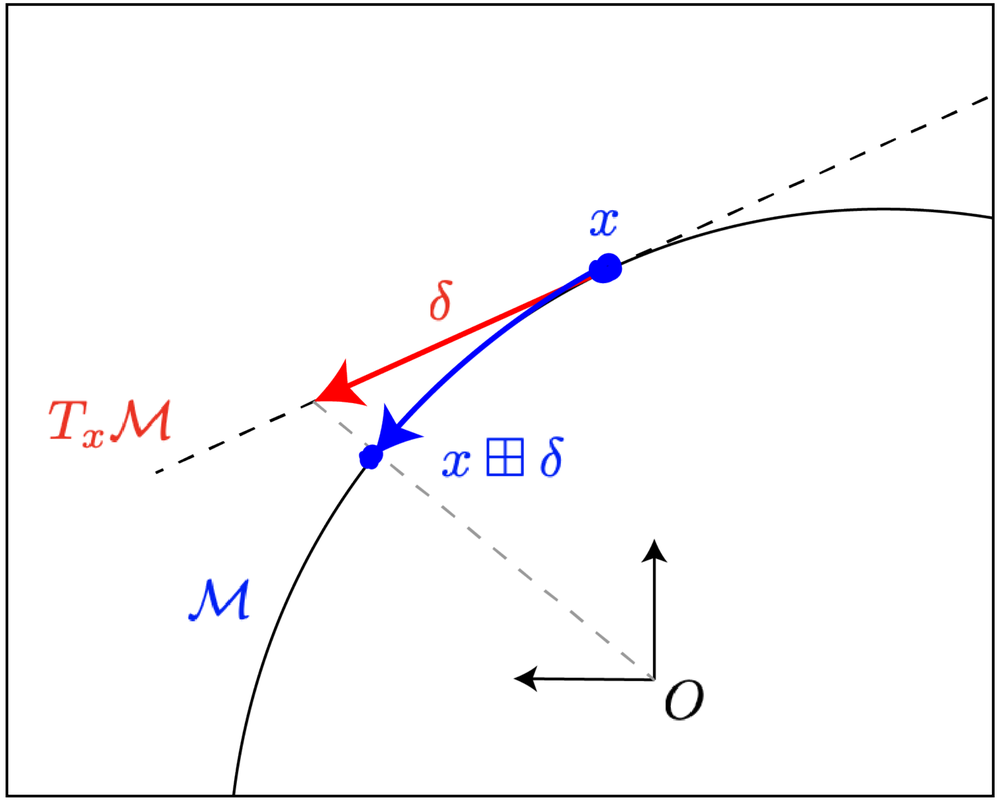
$$R + \delta?$$

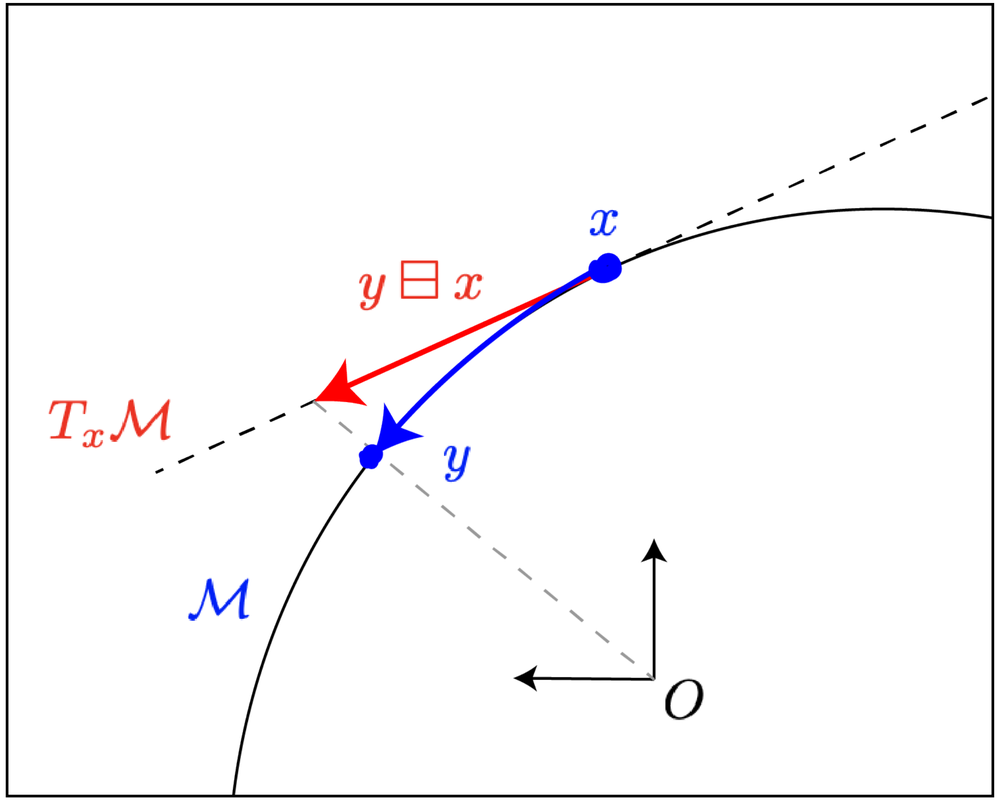
Optimization on Manifolds

- In reality, our optimization variables lie on some smooth manifold M, not in regular Euclidean space. So we need a different notion of a "differential change" than simple addition.
- Idea: Use the tangent space T_xM at a given point x to locally parameterize small changes from that point.

Optimization on Manifolds

- Define a new "plus" operator: $\boxplus : \mathcal{M} \times \mathbb{R}^n \to M$
- Define a new "minus" operator: $\boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^n$





Optimization on Manifolds

- For Lie groups, such as SO(3) (the rotation group) and SE(3) (the group of rigid body transformations), the exponential and logarithmic maps give us straightforward definitions for these operations.

$$x \boxplus \delta = x \cdot \exp(\delta)$$

$$x \boxminus y = \log(y^{-1} x)$$

Optimization on Manifolds

- With these in hand, we define our optimization problem as

$$c\left(\tilde{x}_t\right) = \left\| \tilde{x}_t \boxminus \mu_t \right\|^2_{\Sigma_t^{-1}} + \sum_{i=t-n+1}^{t} \sum_{j=1}^{p} \left\| z_{i,j} \boxminus h\left(x_i, f_{t,j}\right) \right\|^2_{\Sigma_v^{-1}} + \sum_{i=t-n+1}^{t-1} \left\| x_{i+1} \boxminus g\left(x_i\right) \right\|^2_{\Sigma_w^{-1}}$$

- Note that the covariances are now defined over the tangent-space deviation from the mean.
- The correct Taylor expansion is now

$$f(x \boxplus \delta) \approx f(x) + J\delta$$

where

$$J = \left. \frac{\partial f(x \boxplus \delta)}{\partial \delta} \right|_{\delta=0}$$

Optimization on Manifolds

- And our update rules generalize as:
  - Gauss-Newton descent:

$$\overline{\mu}^{(k+1)} \leftarrow \overline{\mu}^{(k)} \boxplus \left(-(J^T J)^{-1} J^T C(\overline{\mu}^{(k)}) \right)$$

  - Marginalization:

$$\bar{\mu}_{t,K} \leftarrow \mu_{t,K} \boxplus \left(-\bar{\Sigma}_{t,K} J_K^\top \left[I - J_M \left(J_M^\top J_M \right)^{-1} J_M^\top \right] C_2 \left(\mu_{t,K}, \mu_{t,M} \right) \right)$$

  - And the new prior is introduced into the optimization problem as

$$(x_K \boxminus \bar{\mu}_K)^\top \bar{\Sigma}_K^{-1} (x_K \boxminus \mu_K)$$

Outline

- **Motivation, FAQs, Terminology**
- **Front End**
  - Feature Extraction
  - Data Association
  - Outlier Rejection
- **Back End**
  - Optimization-Based Formulation
  - Setup, Terminology
  - 3 Steps—Cost Construction, Gauss-Newton Step, Marginalization
- **Back End—Example**
  - EKF—Standard Formulation vs. Optimization Framework
  - State-of-the-art SLAM algorithms
- **Global Optimization**
- **Optimization on Manifolds**
- **Future**

Deep Learning Interventions

- Deep pose estimation:
  - Kendall, Alex, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization." Proceedings of the IEEE international conference on computer vision. 2015.
  - Mohanty, Vikram, et al. "Deepvo: A deep learning approach for monocular visual odometry." arXiv preprint arXiv:1611.06069 (2016).
  - Wang, Sen, et al. "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks." 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.

- Deep data association:
  - Hou, Yi, Hong Zhang, and Shilin Zhou. "Convolutional neural network-based image representation for visual loop closure detection." 2015 IEEE international conference on information and automation. IEEE, 2015.
  - DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "Toward geometric deep slam." arXiv preprint arXiv:1707.07410 (2017).

Deep Learning Interventions

- Tight fusion into the SLAM pipeline:
  - Li, Ruihao, Sen Wang, and Dongbing Gu. "Deepslam: A robust monocular slam system with unsupervised deep learning." IEEE Transactions on Industrial Electronics 68.4 (2020): 3577-3587.
  - Li, Yang, Yoshitaka Ushiku, and Tatsuya Harada. "Pose graph optimization for unsupervised monocular visual odometry." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.

- Deep map enhancement:
  - Salas-Moreno, Renato F., et al. "Slam++: Simultaneous localisation and mapping at the level of objects." Proceedings of the IEEE conference on computer vision and pattern recognition. 2013.
  - McCormac, John, et al. "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks." 2017 IEEE International Conference on Robotics and automation (ICRA). IEEE, 2017.
  - Nicholson, Lachlan, Michael Milford, and Niko Sünderhauf. "Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam." IEEE Robotics and Automation Letters 4.1 (2018): 1-8.
  - Yu, Chao, et al. "DS-SLAM: A semantic visual SLAM towards dynamic environments." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

Dynamic SLAM

- In motion planning, some features often belong to moving objects that may be other robotic agents, dynamic obstacles, etc.
- Extend the SLAM pipeline to track features on a collection of moving rigid bodies in the scene.
- In these scenarios, the above optimization framework must be adapted to account for feature motion.
  - Additional optimization variables for moving features.
  - Motion constraints between moving features.
  - Motion model?

Wang, Chieh-Chih, et al. "Simultaneous localization, mapping and moving object tracking." The International Journal of Robotics Research 26.9 (2007): 889-916.

Questions?

References

- C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J.Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age.IEEE Transactions on Robotics, 32(6):1309–1332, 2016.
- F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing.The International Journal of Robotics Research,25(12):1181–1203, 2006.
- G. Grisetti, R. K˜A14mmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. IEEE Intelligent Transportation Systems Magazine, 2(4):31–43, 2010.
- R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, USA, 2 edition, 2003.
- S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization.The International Journal of Robotics Research, 34:314 – 334, 2015.
- M. Li and A. I. Mourikis. Improving the accuracy of EKF-based visual-inertial odometry. 2012 IEEE International Conference on Robotics and Automation, pages 828–835, 2012.
- M. Li and A. I. Mourikis. Optimization-based estimator design for vision-aided inertial navigation: Supplemental materials. Robotics: Science and Systems, 2012.

References

- P. S. Maybeck, T. In, A. Form, O. B. Means, O. Mechanical, I. Photocopy, and M. P.S. Stochastic models, estimation, and control: Introduction, 1979.
- C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. Rslam: A system for large-scale mapping in constant-time using stereo. International Journal of Computer Vision, 94:198–214, 09 2011.
- A. I. Mourikis and S. I. Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 3565–3572, 2007.
- E. Nerurkar, K. Wu, and S. Roumeliotis. C-klam: Constrained keyframe-based localization and mapping. Proceedings - IEEE International Conference on Robotics and Automation, pages 3638–3643, 05 2014.
- G. Sibley, L. H. Matthies, and G. S. Sukhatme. Sliding window filter with application to planetary landing.J. Field Robotics, 27(5):587–608, 2010.
- J. Sola. Simultaneous localization and mapping with the extended kalman filter. arXiv, 2014.
- J. Sola, J. Deray, and D. Atchuthan. A micro lie theory for state estimation in robotics. ArXiv, abs/1812.01537, 2018.

References

- S. Thrun, W. Burgard, and D. Fox.Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005.
- S. Tully, Hyungpil Moon, G. Kantor, and H. Choset. Iterated filters for bearing-only SLAM. In2008 IEEE International Conference on Robotics and Automation, pages 1442–1448, 2008.
- Z. Zhang. Parameter estimation techniques: a tutorial with application to conic fitting. Image and Vision Computing, 15(1):59 – 76, 1997.
- Murray, Richard M., et al. A mathematical introduction to robotic manipulation. CRC press, 1994.
- Ma, Yi, et al. An invitation to 3-d vision: from images to geometric models. Vol. 26. Springer Science & Business Media, 2012.
- Gálvez-López, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." IEEE Transactions on Robotics 28.5 (2012)
- Kendall, Alex, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization." Proceedings of the IEEE international conference on computer vision. 2015.
- Mohanty, Vikram, et al. "Deepvo: A deep learning approach for monocular visual odometry." arXiv preprint arXiv:1611.06069 (2016).
- Wang, Sen, et al. "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks." 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.

References

- Hou, Yi, Hong Zhang, and Shilin Zhou. "Convolutional neural network-based image representation for visual loop closure detection." 2015 IEEE international conference on information and automation. IEEE, 2015.
- DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich. "Toward geometric deep slam." arXiv preprint arXiv:1707.07410 (2017).
- Li, Ruihao, Sen Wang, and Dongbing Gu. "DeepSLAM: A robust monocular slam system with unsupervised deep learning." IEEE Transactions on Industrial Electronics 68.4 (2020): 3577-3587.
- Li, Yang, Yoshitaka Ushiku, and Tatsuya Harada. "Pose graph optimization for unsupervised monocular visual odometry." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.
- Salas-Moreno, Renato F., et al. "Slam++: Simultaneous localisation and mapping at the level of objects." Proceedings of the IEEE conference on computer vision and pattern recognition. 2013.
- McCormac, John, et al. "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks." 2017 IEEE International Conference on Robotics and automation (ICRA). IEEE, 2017.
- Nicholson, Lachlan, Michael Milford, and Niko Sünderhauf. "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented slam." IEEE Robotics and Automation Letters 4.1 (2018): 1-8.
- Yu, Chao, et al. "DS-SLAM: A semantic visual SLAM towards dynamic environments." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.
- Wang, Chieh-Chih, et al. "Simultaneous localization, mapping and moving object tracking." The International Journal of Robotics Research 26.9 (2007): 889-916

Appendix

Back End: Setup and Terminology

- **Dynamics map with an example:**
  - General form: $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$

    $x_{t+1} = g(x_t) + w_t$, with $w_t \sim N(0, \Sigma_w), \forall t \geq 0.$
  - Associated residual: $x_{t+1} - g(x_t)$
- **Example—2D Extended Kalman Filter (EKF):**
  - Pose: $x_t := (x_{t,1}, x_{t,2}, \theta_t) \in \mathbb{R}^3$
  - Noise: $w_t := (w_t^1, w_t^2, w_t^3) \in \mathbb{R}^3$
  - Dynamics:

$$\dot{x}_t^1 = v \cos\theta_t + w_t^1$$
$$\dot{x}_t^2 = v \sin\theta_t + w_t^2$$
$$\dot{\theta}_t = \omega + w_t^3$$

Back End: Setup and Terminology

- **Measurement map with an example:**
  - General form:

$$h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$$

$$z_{t,j} = h(x_t, f_{t,j}) + v_{t,j}, \text{ with } v_{t,j} \sim N(0, \Sigma_v), \forall t \geq 0.$$

  - Associated residual:

$$z_{t,j} - h(x_t, f_{t,j})$$

- **Example—2D Extended Kalman Filter (EKF):**
  - Feature positions:

$$f_{t,j} := (f^1_{t,j}, f^2_{t,j}) \in \mathbb{R}^2$$

  - Image measurements:

$$z_{t,j} := (z^1_{t,j}, z^2_{t,j}) \in \mathbb{R}^2$$

  - Noise:

$$v_t := (v^1_t, v^2_t) \in \mathbb{R}^2$$

  - Measurement map:

$$z^1_{t,j} = f^1_{t,j} - x^1_t + v^1_t$$

$$z^2_{t,j} = f^2_{t,j} - x^2_t + v^2_t$$

Back End: Marginalization

- **Marginalization:**
  - Equivalence to the Schur complement method:
-

$$\min_{\tilde{x}_{t,M}} c_2'(\tilde{x}_t) = \min_{\tilde{x}_{t,M}} \begin{bmatrix} C_2(\mu_{t,K}, \mu_{t,M}) \\ \tilde{x}_{t,K} - \mu_{t,K} \\ \tilde{x}_{t,M} - \mu_{t,M} \end{bmatrix}^\top \begin{bmatrix} I & J_K & J_M \\ J_K^\top & J_K^\top J_K & J_K^\top J_M \\ J_M^\top & J_M^\top J_K & J_M^\top J_M \end{bmatrix} \begin{bmatrix} C_2(\mu_{t,K}, \mu_{t,M}) \\ \tilde{x}_{t,K} - \mu_{t,K} \\ \tilde{x}_{t,M} - \mu_{t,M} \end{bmatrix}.$$

By the theory of Schur complements:

$$\begin{aligned} &\min_{\tilde{x}_{t,M}} c_2'(\tilde{x}_t) \\ &= \begin{bmatrix} C_2(\mu_{t,K}, \mu_{t,M}) \\ \tilde{x}_{t,K} - \mu_{t,K} \end{bmatrix}^\top \begin{bmatrix} \big(I - J_M(J_m^\top J_M)J_M^\top\big) & \big(I - J_M(J_m^\top J_M)J_M^\top\big)J_K \\ J_K^\top\big(I - J_M(J_m^\top J_M)J_M^\top\big) & J_K^\top\big(I - J_M(J_m^\top J_M)J_M^\top\big)J_K \end{bmatrix} \begin{bmatrix} C_2(\mu_{t,K}, \mu_{t,M}) \\ \tilde{x}_{t,K} - \mu_{t,K} \end{bmatrix} \\ &= (\tilde{x}_{t,K} - \overline{\mu}_{t,K})^\top \overline{\Sigma}_{t,K}^{-1} (\tilde{x}_{t,K} - \overline{\mu}_{t,K}) \end{aligned}$$

where:

$$\begin{aligned} \overline{\Sigma}_{t,K}^{-1} &\leftarrow J_K^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top\big] J_K, \\ \overline{\mu}_{t,K} &\leftarrow \mu_{t,K} - \overline{\Sigma}_{t,K} J_K^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top\big] C_2(\mu_{t,K}, \mu_{t,M}) \end{aligned}$$

Optimization-Based Framework — Proof

- **Feature Augmentation = Gauss-Newton Step:**

Theorem 6.1. *The feature augmentation step of the EKF SLAM algorithm is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1}, \cdots, f_{t,p+p'}) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\tilde{\Sigma}_v^{-1}}.$$

- **Proof (Sketch):**
  - Concatenate terms:

$$\begin{aligned} z_{t,p+1:p+p'} &= (z_{t,p+1}, \cdots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}, \\ f_{t,p+1:p+p'} &= (f_{t,p+1}, \cdots, f_{t,p+p'}) \in \mathbb{R}^{p'd_f}, \\ \tilde{h}(x_t, f_{t,p+1:p+p'}) &:= \big(h(x_t, f_{t,p+1}), \cdots, h(x_t, f_{t,p+p'})\big) \in \mathbb{R}^{p'd_z}, \\ \tilde{\Sigma}_v &= \mathrm{diag}\{\Sigma_v, \cdots, \Sigma_v\} \in \mathbb{R}^{p'd_z \times p'd_z}. \end{aligned}$$

  - Rewrite cost:

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1:p+p'}) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \|z_{t,p+1:p+p'} - \tilde{h}(x_t, f_{t,p+1:p+p'})\|^2_{\tilde{\Sigma}_v^{-1}}.$$

Optimization-Based Framework — Proof

- **Feature Augmentation = Gauss-Newton Step:**

Theorem 6.1. *The feature augmentation step of the EKF SLAM algorithm is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1}, \cdots, f_{t,p+p'}) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\tilde{\Sigma}_v^{-1}}.$$

- **Proof (Sketch):**
  - Compute $C(\tilde{x}_t)$ and J:

$$C(\tilde{x}_t, f_{t,p+1:p+p'}) := \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t - \mu_t) \\ \Sigma_v^{-1/2}\left(z_{t,p+1:p+p'} - \tilde{h}(x_t, f_{t,p+1:p+p'})\right) \end{bmatrix}.$$

$$J = \begin{bmatrix} \Sigma_t^{-1/2} & O \\ -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,x}\begin{bmatrix} I & O \end{bmatrix} & -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f} \end{bmatrix}$$

  - Apply Gauss-Newton Equations:

$$\overline{\Sigma}_t^{-1} \leftarrow J^\top J,$$
$$\overline{\mu}_t \leftarrow \mu_t - (J^\top J)^{-1} J^\top C(\mu_t).$$

Optimization-Based Framework — Proof

- **Feature Augmentation = Gauss-Newton Step:**

Theorem 6.1. *The feature augmentation step of the EKF SLAM algorithm is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1}, \cdots, f_{t,p+p'}) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\tilde{\Sigma}_v^{-1}}.$$

- **Proof (Sketch):**
  - Result — The Gauss-Newton Equations above yield Alg. 2, Lines 4, 9:

$$4 \quad \mu_t \leftarrow \left(\mu_t, \ell(\mu_{t,x}, z_{t,p+1}, \cdots, z_{t,p+p'})\right) \in \mathbb{R}^{d_x+(p+p')d_f}$$

$$9 \quad \Sigma_t \leftarrow \begin{bmatrix} \Sigma_{t,xx} & \Sigma_{t,xf} & \Sigma_{t,xx}L_x^\top \\ \Sigma_{t,fx} & \Sigma_{t,ff} & \Sigma_{t,fx}L_x^\top \\ L_x\Sigma_{t,xx} & L_x\Sigma_{t,xf} & L_x\Sigma_{t,xx}L_x^\top + L_z\tilde{\Sigma}_v L_z^\top \end{bmatrix} \in \mathbb{R}^{(d_x+(p+p')d_f)\times(d_x+(p+p')d_f)}$$

Optimization-Based Framework — Proofs

- **Feature Update = Gauss-Newton Step:**

Theorem 6.2. *The feature update step of the EKF SLAM algorithm (Alg. 3) is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,3}(\tilde{x}_t) := \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_k)\|^2_{\Sigma_v^{-1}}.$$

- **Proof (Sketch):**
  - Concatenate terms:

$$\begin{aligned} z_{t,1:p} &:= (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{pd_z}, \\ f_{t,1:p} &:= (f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{pd_f}, \\ \tilde{h}(x_t, f_{t,1:p}) &:= \big(h(x_t, f_{t,1}), \cdots, h(x_t, f_{t,p})\big) \in \mathbb{R}^{pd_z}, \\ \tilde{\Sigma}_v &:= \mathrm{diag}\{\Sigma_v, \cdots, \Sigma_v\} \in \mathbb{R}^{pd_z \times pd_z}. \end{aligned}$$

  - Rewrite cost:

$$c_{EKF,t,1}(\tilde{x}_t) = \|\tilde{x}_t^\star - \mu_t\|^2_{\Sigma_t^{-1}} + \|z_{t,1:p} - \tilde{h}(\tilde{x}_t^\star)\|^2_{\tilde{\Sigma}_v^{-1}}.$$

Optimization-Based Framework — Proofs

- **Feature Update = Gauss-Newton Step:**

Theorem 6.2. *The feature update step of the EKF SLAM algorithm (Alg. 3) is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,3}(\tilde{x}_t) := \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_k)\|^2_{\Sigma_v^{-1}}.$$

- **Proof (Sketch):**
  - Compute $C(\tilde{x}_t)$ and J:

$$C(\tilde{x}_t) := \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t - \mu_t) \\ \tilde{\Sigma}_v^{-1/2}(z_{t,1:p} - \tilde{h}(\tilde{x}_t)) \end{bmatrix} \qquad J = \begin{bmatrix} \Sigma_t^{-1/2} \\ -\tilde{\Sigma}_v^{-1/2} H_t \end{bmatrix}$$

  - Apply Gauss-Newton Equations:

$$\overline{\Sigma}_t^{-1} \leftarrow J^\top J,$$
$$\overline{\mu}_t \leftarrow \mu_t - (J^\top J)^{-1} J^\top C(\mu_t).$$

Optimization-Based Framework — Proofs

- **Feature Update = Gauss-Newton Step:**

Theorem 6.2. *The feature update step of the EKF SLAM algorithm (Alg. 3) is equivalent to one Gauss-Newton step on the cost function* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,3}(\tilde{x}_t) := \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_k)\|^2_{\Sigma_v^{-1}}.$$

- **Proof (Sketch):**
  - Result — The Gauss-Newton Equations above yield Alg. 3, Lines 5-6:

5 $\overline{\mu_t} \leftarrow \mu_t + \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1} \big(z_{t,1:p} - h(\mu_t, f_{t,1:p})\big) \in \mathbb{R}^{d_x+pd_f}$.

6 $\overline{\Sigma_t} \leftarrow \Sigma_t - \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1} H_t \Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$.

Optimization-Based Framework — Proofs

- **State Propagation = Marginalization Step:**

Theorem 6.3. *The state propagation step of the EKF SLAM algorithm (Alg. 4) is equivalent to one Marginalization step on the cost function* $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) = \|\tilde{x}_t - \overline{\mu}_t\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}$$

- **Proof (Sketch):**
  - Identify c_K, c_M, C_K, C_M:
  -

$$c_K(x_{t+1}) = 0$$

$$c_M(\tilde{x}_t, x_{t+1}) = \|\tilde{x}_t - \overline{\mu_t}\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}.$$

$$C_K(\tilde{x}_{t,K}) = 0 \in \mathbb{R}$$

$$C_M(\tilde{x}_{t,K}, \tilde{x}_{t,M}) = \begin{bmatrix} \bar{\Sigma}_t^{-1/2}(\tilde{x}_t - \overline{\mu_t}) \\ \Sigma_w^{-1/2}\big(x_{t+1} - g(x_t)\big) \end{bmatrix} \in \mathbb{R}^{2d_x+pd_f}.$$

Optimization-Based Framework — Proofs

- **State Propagation = Marginalization Step:**

Theorem 6.3. *The state propagation step of the EKF SLAM algorithm (Alg. 4) is equivalent to one Marginalization step on the cost function* $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) = \|\tilde{x}_t - \overline{\mu}_t\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}$$

- **Proof (Sketch):**
  - Compute J_K, J_M, and apply Marginalization equations:

$$\Sigma^{-1}_{t+1,K} \leftarrow J_K^\top \left[I - J_M (J_M^\top J_M)^{-1} J_M^\top\right] J_K,$$

$$\mu_{t+1,K} \leftarrow \overline{\mu}_{t,K} - \Sigma_{t+1,K} J_K^\top \left[I - J_M (J_M^\top J_M)^{-1} J_M^\top\right] C_2(\overline{\mu}_{t,K}, \overline{\mu}_{t,M})$$

Optimization-Based Framework — Proofs

- **State Propagation = Marginalization Step:**

Theorem 6.3. *The state propagation step of the EKF SLAM algorithm (Alg. 4) is equivalent to one Marginalization step on the cost function* $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \to \mathbb{R}$, *given by:*

$$c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) = \|\tilde{x}_t - \overline{\mu}_t\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}$$

- **Proof (Sketch):**
  - Result — The Marginalization Equations above yield Alg. 4, Lines 5-6:

4 $\mu_{t+1} \leftarrow \left(g(\overline{\mu}_t), \overline{\mu}_{t,f,1:p}\right) \in \mathbb{R}^{d_x+pd_f}$.

5 $\Sigma_{t+1} \leftarrow \begin{bmatrix} G_t\overline{\Sigma}_{t,xx}G_t^\top + \Sigma_w & G_t\overline{\Sigma}_{t,xf} \\ \overline{\Sigma}_{t,fx}G_t^\top & \overline{\Sigma}_{t,ff} \end{bmatrix} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$.